**Введение**

Одной из наиболее актуальных проблем при разработке web-приложений и их работе являются так называемые боты.

Бот – это специальная программа, выполняющая некоторые рутинные действия, такие, как регистрация нового пользователя, отправка комментариев и сообщений, сбор данных и т.д.

К основным вредоносным действиям относят:

Сбор личных данных, электронных адресов и мобильных телефонов из контактных форм;

DoS- и DDoS-атаки;

Некоторые вирусы;

Перегрузка интернет-каналов потоками ненужной информации;

Одним из наиболее популярных методов противодействия интернет-ботам является методология CAPTCHA.

CAPTCHA – автоматизированный публичный тест Тьюринга для различения компьютеров и людей. Это аудиовизуальный инструмент, с помощью которого администраторы защищают сайты от автоматических регистраций на сайтах интернет-ботов и выполнения ими поставленных недоброжелателями целей.

Актуальность проблемы противодействия интернет-ботам с развитием и популяризацией интернет ресурсов становится все острее, так как технологии активно развиваются и злоумышленники придумывают новые способы кражи информации и нанесения различного рода вреда.

Целью работы является написание одного из вариантов алгоритма CAPTCHA.

Для выполнения проекта были поставлены следующие задачи:

* Рассмотреть понятие CAPTCHA;
* Рассмотреть основные виды технологии CAPTCHA;
* Изучить инструменты редактирования кода для веб-разработки;
* Реализовать действующее веб-приложение.

**Глава 1. Теоретическая часть**

* 1. Основные технологии создания веб-приложения

При создании любого веб-приложения обязательно используют HTML, CSS, JavaScript и PHP. При помощи HTML, CSS и JavaScript (или их аналогов) структуру, внешний вид и динамику интерфейса, а также взаимосвязь между его элементами, если она должна присутствовать в приложении.

На языке PHP реализуют взаимосвязь между клиентом и сервером, на котором расположен интернет-ресурс, связь приложения с базой данных и проводят вычислительные операции.

**HTML/CSS/JS**

HTML – язык гипертекстовой разметки, который позволяет создавать и структурировать разделы, ссылки и блоки в веб-приложении или на сайте.

CSS – язык описания внешнего вида приложения. В настоящее время ни одни сайт не обходится без применения дополнительных стилей, поэтому CSS принято считать второй из базовых технологий.

JavaScript – язык программирования, с помощью которого приложениям добавляют интерактивность. Язык активно развивается и поддерживает такие отрасли программирования, как функциональный и объектно-ориентированный подходы. JS обычно используют для проверки правильности заполнения полей форм и ограничения вводимых в них данных, создания лоадеров, осуществления запросов на сервера и различных динамических манипуляций со структурой сайта.

* 1. Анализ основных технологий CAPTCHA

**Распознавание защищенного текста**

Распознавание делится на 2 основных этапа:

Определение местоположения и границ каждого символа;

Собственно распознавание символа.

Если символы постоянно расположены на одних и тех же позициях, из написанных выше этапов остается только второй. Следовательно, координаты символов нужно изменять динамически.

Также, для повышения уровня защищенности, символы не должны по цвету сильно отличаться от цвета фона. По возможности, символы должны быть разных цветов.

Таким образом, необходимо создать визуальный шум, трудноотделимый от символов, либо размещать символы близко друг к другу или накладывая их друг на друга.

Для того, чтобы алгоритмам автоматического распознавания было труднее расшифровывать текст, можно также применять для генерации надписи различные шрифты, так как автоматизированные системы распознавания нередко используют наложение маски и эталонных шрифтов.

**Аудио-CAPTCHA**

Аудио-капча предлагает пользователю прослушать некоторую фразу, набор букв или цифр, после чего ее ввести. Фраза обычно произносится с изменяемой тональностью, паузами и фоновым шумом. На такую капчу сможет ответить человек с нарушениями зрения. Но пользователь должен иметь оборудование для воспроизведения звука.

Создание звуковых шумов требовательна к ресурсам сервера. Также, ее трудно реализовать недостаточно квалифицированному программисту.

Из-за трудностей реализации и требований к ресурсам сервера данную капчу применяют редко.

Одним из видов аудио-капчи являются задаваемые пользователю задачи и загадки. Ответ чаще всего предлагают выбрать из списка, или ввести в поле. Но выбор ответа из предложенного списка увеличивает вероятность правильного ответа, так как правильный ответ – обязательно есть в списке. Поэтому пользователю задают не один вопрос, а несколько.

Еще одним минусом данного метода является необходимость посетителю сайта быть ознакомленным с алфавитом и языком, на котором задаются вопросы.

**Распознавание предметов**

При использовании данного метода пользователя просят узнать показанные на картинке (или картинках) предметы.

Данный метод делят на два вида:

Показ на картинке предмета и просьба ввести в поле его название;

Указание пользователю названия предмета и просьба выбрать несколько изображений, на которых он изображен, из списка.

Пользователю легче отличить один предмет от другого, чем сильно искаженные, похожие друг на друга буквы. Боту – наоборот. Для автоматического распознавания злоумышленники чаще всего должны прибегать к использованию нейронных сетей. Нейросети сильно ограничены в функциональности и требовательны к ресурсам компьютера злоумышленника, что затрудняет их применение, поэтому данный метод можно считать наиболее безопасным из рассмотренных.

**Методы, позволяющие усложнить распознавание**

Одним из дополнительных способов усложнения разгадывания капчи автоматически является ограничение количества запросов с одного IP-адреса. Данный метод прост и малоэффективен из-за развития прокси и подставных IP-адресов. Ограничение через cookie тоже неэффективно, так как многие пользователи их игнорируют или удаляют и получают возможность новых попыток.

Благодаря тому, что пользователь при заполнении форм заполняет только те поля, которые видит и смотрит на визуальную часть сайта, а не на разметку, был разработан еще один метод – создание невидимых пользователь полей ввода. Боты обычно смотрят в разметку страницы и обнаруживают поля, которые автоматически заполняют. Это позволяет идентифицировать бота и ограничить ему доступ к сайту.

Также можно ограничить ввод данных по времени, так как компьютер всегда тратит на работу и вывод данных меньше времени, чем человек, этим себя выдавая.

**Глава 2. Практическая часть**

2.1. Выбор редактора кода

Для разработки данного веб-приложения был выбран Visual Studio Code. Данный редактор полностью бесплатен. Также, он наделен большей частью необходимых функций:

* Поддержка большинства популярных языков программирования;
* Подсветка синтаксиса;
* Возможность одновременной работы с несколькими проектами;
* Интегрирована система подсказок;
* Обширная библиотека плагинов, позволяющих расширить функционал редактора (Emmet – значительно ускоряет написание текста на языках HTML, CSS, JS и другие).

Существует множество других редакторов кода и сред разработки для разработки приложений:

* WebStorm – полноценная среда разработки веб-приложений. Множество полезных функций, таких как работа с git через удобный интерфейс, умный анализатор кода и плагины идут «из коробки». Однако, все это не бесплатно. Пользователь должен оформить подписку, либо получить студенческую лицензию;
* Atom – приятный и простой в использовании редактор. Поддерживает все основные языки программирования. Главные недостатки – потребляет много ресурсов системы и не имеет большого количества плагинов, сложнее настроить под себя;
* Sublime Text 3 – некогда популярная программа, высокая производительность и огромное количество уроков по настройке под различные проекты. Однако сейчас его функционал не настолько широк, как у того же VS Code. Подойдет для создания простых сайтов.
* PHPStorm – среда разработки от компании JetBrains, имеющая значительно расширенный функционал среды WebStorm: поддержку языка PHP и большинства его библиотек и фреймворков. Платный.

2.2. Структура и функционал приложения

После того, как была определена среда разработки, необходимо определить структуру и функционал приложения. Ранее упоминалось о том, что структура сайта должна быть простой и дружелюбной в использовании.

В нашем случае структура приложения состоит из одной веб-страницы с формой и текстовым блоком CAPTCHA, которая при правильно заполненном поле капчи сообщает, что все введено верно. В противном случае выводится сообщение об ошибке с просьбой повторить попытку ввода.

Далее определим функционал приложения:

* Повторная генерация изображения с текстом капчи с целью избежания повтора задачи и снижения риска прохождения ее ботом.

2.3. Проектирование интерфейса

Определим состав страницы нашего приложения:

Чтобы показать визуально цель применения капчи, страница будет состоять из формы, включающей в себя такие поля, как:

* Поле ввода логина пользователя;
* Поле ввода электронной почты;
* Поле ввода пароля;
* Генерируемую картинку с капчей;
* Поле ввода для самой капчи.

2.4. Процесс разработки

После проектирования функционала и структуры приложения, а также его интерфейса, следует процесс написания кода.

Для начала создаем HTML-разметку. Необходимо верстать пустую страницу, в которую поместить форму.

<!Doctype html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Captcha</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<header>

</header>

<main>

<form action="go.php" method="post" class="login\_\_box">

<h1>LOGIN</h1>

<input name="name" placeholder="Username:" type="text" autocomplete="off"><br>

<input name="email" placeholder="Email:" type="email" autocomplete="off"><br>

<input type="password" placeholder="Password:"><br>

<!-- Тут будет обитать сама капча. -->

<img title="Щёлкните для нового кода" alt="Код" src="jcaptcha.php" style="border: 1px solid #000000" onclick="this.src='jcaptcha.php?id=' + (+new Date());"><br>

<input name="captcha" placeholder="Код с картинки:" type="text" autocomplete="off"><br>

<input type="submit" value="Login">

<!-- Тут она закончится. -->

</form>

</main>

<footer>

</footer>

</body>

</html>

Данный код представляет собой четыре поля формы, описанные ранее и кнопку, при нажатии на которую начинается проверка введенных данных, а также контентные блоки header, main и footer, не несущие в данном случае особой нагрузки, но предусмотренных HTML5 как основные.

Чтобы наша верстка приобрела привычный для пользователя вид, нужно добавить стили для страницы и ее элементов.

html {

height: 100%;

}

body {

font-family: sans-serif;

background: linear-gradient(-50deg, #970d1a, #4e204d);

height: 100%;

display: flex;

justify-content: center;

align-items: center;

overflow-y: hidden;

overflow-x: hidden;

}

.login\_\_box {

height: auto;

width: 300px;

padding: 40px;

/\* transform: translate(-50%, -50%); \*/

background: #191919;

text-align: center;

background: linear-gradient(-50deg, #970d1a, #4e204d);

box-shadow: 0 0 10px rgba(0,0,0,0.5);

border-radius: 5px;

}

.login\_\_box h1 {

color: #ffffff;

text-transform: uppercase;

font-weight: 500;

}

.login\_\_box input[type = "text"],

.login\_\_box input[type = "password"],

.login\_\_box input[type = "email"] {

border: 0;

background: none;

display: block;

margin: 8px auto;

text-align: center;

border: 2px solid #3498db;

padding: 14px 10px;

width: 200px;

outline: none;

color: #ffffff;

border-radius: 24px;

transition: 0.25s;

}

.login\_\_box input[type = "text"]:focus,

.login\_\_box input[type = "password"]:focus,

.login\_\_box input[type = "email"]:focus {

width: 280px;

border-color: #2ecc71;

}

.login\_\_box input[type = "submit"] {

border: 0;

background: none;

display: block;

margin: 15px auto;

text-align: center;

border: 2px solid #2ecc71;

padding: 14px 40px;

width: 200px;

outline: none;

color: #ffffff;

border-radius: 24px;

transition: 0.25s;

cursor: pointer;

}

.login\_\_box input[type = "submit"]:hover {

background: #2ecc71;

}

Далее нужно предусмотреть открытие пользователем страницы с мобильного устройства, то есть добавить на страницу адаптив.

@media (max-width: 425px) {

main {

height: 100%;

width: 100%;

margin: 0;

padding: 0;

}

body {

margin: 0;

}

.login\_\_box {

width: 100%;

height: 100%;

padding: 0;

margin: 0;

box-shadow: none;

display: flex;

flex-direction: column;

justify-content: center;

align-items: center;

}

}

После завершения базовой разметки и стилизации, необходимо написать алгоритмы для генерации изображения капчи и ее проверки после заполнения поля пользователем при помощи языка PHP.