

Создание CAPTCHA на языке программирования PHP

Акентьев Данила Денисович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данной работе предложен метод создания динамической системы CAPTCHA с использованием языка программирования PHP, целью которой является защита от автоматических ботов и обеспечение надежности веб-приложений. Особенностью предлагаемого подхода является индивидуальная генерация координат для каждого символа текста, что позволяет точно контролировать его размещение на изображении и предотвратить его выход за пределы заданных границ. Метод основывается на генерации случайного текста с использованием алгоритма перетасовки символов, который затем отображается на изображении с применением случайного наклона и положения каждого символа. Для повышения сложности CAPTCHA добавлены случайные линии и пиксели, создающие визуальные помехи. Этот подход способствует повышению эффективности распознавания CAPTCHA и делает её более устойчивой к атакам ботов. Для вычисления точных координат каждого символа используется функция `imagettfbbox()`, которая позволяет вычислить габариты каждого символа с учетом угла наклона. Координаты символов генерируются таким образом, что они строго укладываются в пределах изображения, исключая возможность их выхода за пределы. Этот метод также позволяет легко адаптировать параметры шрифта, углы наклона и отступы между символами для дальнейшего улучшения защиты. Предложенная система CAPTCHA имеет значительное преимущество в плане гибкости и настроек, а также в плане возможности создания нестандартных и сложных изображений, что значительно усложняет работу автоматических систем и улучшает безопасность веб-ресурсов.

Ключевые слова: CAPTCHA, защита от ботов, генерация случайного текста, PHP, динамическая система CAPTCHA, изображение, наклон текста, вычисление координат, обработка изображений, безопасность веб-приложений, случайные помехи, визуальные помехи, шрифт TTF, программирование на PHP.

Creating a CAPTCHA in the PHP programming language

Akentev Danila Denisovich

Sholom-Aleichem Priamursky State University

Student

Abstract

In this paper, we propose a method for creating a dynamic CAPTCHA system using the PHP programming language, which aims to protect against automated bots and ensure the reliability of web applications. A special feature of the proposed approach is the individual generation of coordinates for each character of the text, which allows you to accurately control its placement on the image and prevent it from going beyond the specified boundaries. The method is based on generating random text using a character shuffling algorithm, which is then displayed on the image using a random tilt and position of each character. Random lines and pixels that create visual interference have been added to increase the complexity of the CAPTCHA. This approach helps improve the efficiency of CAPTCHA recognition and makes it more resistant to bot attacks. To calculate the exact coordinates of each character, the `imagettfbbox()` function is used, which allows you to calculate the dimensions of each character based on the angle of inclination. The coordinates of the symbols are generated in such a way that they strictly fit within the image, excluding the possibility of their going beyond. This method also makes it easy to adapt font settings, tilt angles, and character margins to further improve protection.

The proposed CAPTCHA system has a significant advantage in terms of flexibility and settings, as well as the ability to create non-standard and complex images, which significantly complicates the operation of automated systems and improves the security of web resources.

Keywords: CAPTCHA, bot protection, random text generation, PHP, dynamic CAPTCHA system, image, text tilt, coordinate calculation, image processing, web application security, random interference, visual interference, TTF font, PHP programming.

1 Введение**1.1 Актуальность**

С каждым годом растет число автоматических систем (ботов), предназначенных для обхода механизмов защиты на веб-ресурсах. Веб-приложения и онлайн-сервисы сталкиваются с все более сложными попытками автоматических атак, включая спам, фальсификацию регистраций, краже учетных данных и другие виды мошенничества. Одним из наиболее эффективных методов защиты от таких атак является использование CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart), системы, которая позволяет различать действия людей и ботов.

Однако с развитием технологий, таких как машинное обучение и оптическое распознавание символов (OCR), традиционные методы CAPTCHA становятся уязвимыми для автоматических систем. Чтобы повысить устойчивость к таким атакам, важно разрабатывать новые подходы, которые бы эффективно усложняли задачу распознавания CAPTCHA для ботов и при этом оставались бы удобными для пользователей.

Предложенная в статье система CAPTCHA, использующая индивидуальную генерацию координат для каждого символа, значительно усложняет задачу распознавания текста для автоматических систем. Такой

подход позволяет эффективно контролировать размещение символов на изображении, исключая их выход за пределы, а также добавлять случайные элементы помех для создания дополнительной сложности. Это делает систему более защищенной от современных атак, повышая её эффективность в реальных условиях эксплуатации.

Таким образом, актуальность работы заключается в разработке и внедрении инновационной системы CAPTCHA, которая отвечает современным требованиям безопасности и предоставляет дополнительные возможности для борьбы с автоматическими атаками.

1.2 Обзор исследований

Ранее вопросами противодействия атакам типа brute force и использования графических объектов CAPTCHA занимались ряд авторов. Так, А.В. Яковлев, А.М. Шуваева и М.А. Пеливан в своей работе «Противодействие атаке brute force на основе графических объектов типа CAPTCHA» [1] рассмотрели различные виды CAPTCHA, оценили их сложность для прохождения с помощью автоматизированных систем и предложили методы использования CAPTCHA для повышения устойчивости парольной аутентификации к атакам грубой силы. О.Н. Жданов в своей исследовательской работе «Распознавание современных CAPTCHA» [2] подробно изучил проблему автоматизированного распознавания современных типов CAPTCHA. В этом контексте автор рассматривает три ключевые задачи: предварительную обработку изображений, сегментацию и распознавание отдельных символов или элементов CAPTCHA. Кроме того, М.П. Явич и И.З. Пирцхалава в статье «Проблемы развития CAPTCHA» [3] предложили различные варианты CAPTCHA и обсудили пути усложнения таких тестов в ответ на развитие более мощных ботов, способных обходить традиционные системы защиты. В работе М. Кумар, М.К. Джиндал и М. Кумар (2018) представлен систематический анализ различных типов CAPTCHA, включая текстовые, изображенные, аудио- и видео CAPTCHA, с оценкой их безопасности и удобства использования. Рассматриваются 16 методов генерации текстовых CAPTCHA, использующих различные языки и техники деформации, которые были успешно взломаны с использованием алгоритмов машинного обучения, таких как CNN, KNN и SVM. Анализируются также изображения и аудио CAPTCHA, которые, несмотря на высокий уровень безопасности, поддаются атаке с использованием аналогичных методов. В работе также обсуждаются видео CAPTCHA, которые, несмотря на высокие показатели безопасности (98-100%), имеют уязвимости, позволяющие успешно взломать системы с применением SIFT, NN и простого OCR. Этот обзор подчеркивает необходимость развития более сложных и устойчивых CAPTCHA-систем для защиты от современных автоматических атак [4]. В статье Н. Тарик, Ф.А. Хан, С.А. Мокурраб и Г. Шривастава (2021) представлен обзор механизма CAPTCHA, который используется для аутентификации пользователей и защиты от вредоносных ботов. Рассматриваются различные типы CAPTCHA, их преимущества, недостатки и рекомендации по

проектированию. Также подробно анализируются методы обхода CAPTCHA с использованием алгоритмов машинного обучения и нейронных сетей, таких как CNN, KNN и SVM. Обзор, основанный на более чем 200 исследованиях, подчеркивает необходимость разработки более устойчивых и удобных систем CAPTCHA, а также освещает нерешённые проблемы в их проектировании [5].

1.3 Цель исследования

Целью исследования является разработка и реализация эффективной системы CAPTCHA на языке программирования PHP, которая обеспечивает надежную защиту веб-приложений от автоматических атак ботов. Особое внимание уделяется индивидуальной генерации координат для каждого символа текста с учетом наклона и случайных помех, что повышает сложность распознавания и делает систему более устойчивой к современным методам обхода защиты.

2 Материалы и методы

Для разработки системы CAPTCHA использовался язык программирования PHP с библиотекой GD для работы с изображениями. В качестве шрифта применялся TrueType Font (TTF), что обеспечивало гибкость в настройке текста на изображении, включая возможность наклона и изменения размера.

Генерация текста CAPTCHA основывалась на случайной выборке символов из заданного набора букв и цифр. Для создания изображения использовалась функция библиотеки GD, которая позволяла создавать фон, задавать цвета и добавлять визуальные помехи (случайные линии и пиксели) для усложнения распознавания.

Для каждого символа текста рассчитывались координаты с учётом его габаритов и наклона, что позволило точно размещать символы на изображении, предотвращая их выход за пределы. Позиции символов вычислялись индивидуально, что повышало сложность распознавания для ботов.

Кроме того, для улучшения защиты и усложнения автоматического распознавания текста использовались случайные помехи в виде линий и пикселей, которые затрудняли работу алгоритмов OCR. После формирования изображения оно выводилось в формате PNG с помощью библиотеки GD.

Методы, использованные в исследовании, позволили создать гибкую и устойчивую CAPTCHA-систему, которая эффективно защищает веб-приложения от автоматических атак.

3 Результаты и обсуждения

Начнем исследование с создания файла `index.php`, в котором пропишем код (рис.1).

```
<?php
session_start();

// Проверяем, был ли отправлен правильный код CAPTCHA
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    if (isset($_POST['captcha_input']) && $_POST['captcha_input'] == $_SESSION['captcha_text']) {
        echo "CAPTCHA пройдена!";
    } else {
        echo "Неверный код CAPTCHA. Попробуйте снова.";
    }
}
?>

<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Тест CAPTCHA</title>
</head>
<body>
    <h1>Введите текст с изображения</h1>

    <form action="index.php" method="POST">
        <p></p>
        <p><input type="text" name="captcha_input" required></p>
        <p><input type="submit" value="Проверить"></p>
    </form>
</body>
</html>
```

Рисунок 1. Index.php

PHP-часть — Обработка запроса и проверка CAPTCHA

Запуск сессии: В начале скрипта запускается сессия с помощью функции `session_start()`. Это необходимо для хранения значений между запросами, например, для сохранения текста CAPTCHA в сессии и его последующей проверки.

Проверка типа запроса: далее идет проверка, был ли запрос отправлен методом `POST`. Это делается для того, чтобы обработать данные только в случае, если форма была отправлена.

Проверка введенного текста CAPTCHA: если форма была отправлена, проверяется, был ли введен текст (из поля `captcha_input`). Далее сравнивается введенный пользователем текст с тем, что хранится в сессии (сгенерированный текст CAPTCHA). Если они совпадают, выводится сообщение о том, что CAPTCHA пройдена. В противном случае пользователю сообщается, что код неверен.

HTML-форма — Вывод CAPTCHA и ввод пользователем

Структура страницы: В HTML-документе задается базовая структура страницы, включая метатеги для правильного отображения символов и адаптивности.

Заголовок страницы: Заголовок страницы сообщает пользователю, что он должен ввести текст с изображения CAPTCHA.

Форма для ввода CAPTCHA: На странице отображается форма, в которой:

Вставляется изображение CAPTCHA с помощью тега `<img>`. Это изображение будет отображать случайно сгенерированный текст CAPTCHA.

Поле для ввода текста CAPTCHA, где пользователь должен ввести символы с изображения. Это поле обязательно для заполнения.

Кнопка для отправки формы, которая проверит введенный код.

Введите текст с изображения



Проверить

Рисунок 2. Внешний вид страницы

CAPTCHA пройдена!

Введите текст с изображения

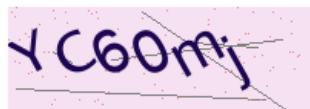


Проверить

Рисунок 3. Успешное прохождение CAPTCHA

Неверный код CAPTCHA. Попробуйте снова.

Введите текст с изображения



Проверить

Рисунок 4. Не успешное прохождение CAPTCHA

Взаимодействие между PHP и HTML:

В файле captcha.php генерируется изображение с текстом CAPTCHA, которое показывается пользователю.

Когда пользователь вводит текст и отправляет форму, данные отправляются обратно на текущую страницу (index.php), где PHP проверяет введенный текст и выводит результат: либо сообщение о том, что CAPTCHA пройдена, либо о том, что введенный код неверен.

Заключение:

Этот код представляет собой простую форму CAPTCHA, которая помогает защитить веб-страницу от автоматических ботов, проверяя правильность ввода текста с изображения (рис.5.1, 5.2).

```

<?php
session_start();

// Устанавливаем заголовок для изображения
header('Content-Type: image/png');

// Размер изображения
$width = 200;
$height = 70;
$image = imagecreatetruecolor($width, $height);

// Генерация случайной палитры цветов
$bg_color = imagecolorallocate($image, rand(200, 255), rand(200, 255), rand(200, 255)); // светлый фон
$text_color = imagecolorallocate($image, rand(0, 100), rand(0, 100), rand(0, 100)); // темный текст
$line_color = imagecolorallocate($image, rand(100, 200), rand(100, 200), rand(100, 200)); // случайные линии
$pixel_color = imagecolorallocate($image, rand(100, 255), rand(100, 255), rand(100, 255)); // случайные пиксели

// Заполняем фон
imagefilledrectangle($image, 0, 0, $width, $height, $bg_color);

// Добавляем случайные линии (повышаем сложность)
for ($i = 0; $i < 5; $i++) {
    imageline($image, rand(0, $width), rand(0, $height), rand(0, $width), rand(0, $height), $line_color);
}

// Добавляем случайные пиксели (шум)
for ($i = 0; $i < 100; $i++) {
    imagesetpixel($image, rand(0, $width), rand(0, $height), $pixel_color);
}

// Генерация случайного текста
$characters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789';
$captcha_text = substr(str_shuffle($characters), 0, 6); // 6 случайных символов

// Сохраняем CAPTCHA текст в сессии для проверки
$_SESSION['captcha_text'] = $captcha_text;

// Путь к шрифту
$font = __DIR__ . '/arial.ttf'; // Укажите путь к шрифту TTF
if (!file_exists($font)) {
    die('Шрифт не найден');
}

$font_size = 30; // Размер шрифта
$angle = rand(-30, 30); // Случайный угол наклона текста (от -30 до 30)

// Массив для хранения координат каждого символа
$coordinates = [];

// Генерация координат для каждого символа
$x = 10; // Начальная позиция по оси X
$y = rand($font_size + 10, $height - 10); // Начальная позиция по оси Y

for ($i = 0; $i < strlen($captcha_text); $i++) {
    // Рассчитываем максимальные координаты для каждого символа
    $bbox = imageftbbox($font_size, $angle, $font, $captcha_text[$i]);
    $char_width = abs($bbox[2] - $bbox[0]);
    $char_height = abs($bbox[5] - $bbox[3]);

    // Убедимся, что символ не выйдет за пределы картинки
    if ($x + $char_width > $width - 10) {
        break; // Если символ выходит за пределы картинки, выходим из цикла
    }

    // Генерация случайной позиции для каждого символа
    $coordinates[] = [
        'x' => $x,
        'y' => $y,
        'char' => $captcha_text[$i]
    ];
}

```

Рисунок 5.1. Captcha.php

```
// Сдвигаем позицию X для следующего символа
$x += $char_width + rand(5, 10); // Добавляем случайный отступ между символами
}

// Печатаем символы на картинке
foreach ($coordinates as $coord) {
    imageTTFtext($image, $font_size, $angle, $coord['x'], $coord['y'], $text_color, $font, $coord['char']);
}

// Отправляем изображение в браузер
imagepng($image);
imagedestroy($image);
?>
```

Рисунок 5.2. Captcha.php

Запуск сессии и установка заголовков для изображения

Запуск сессии: с помощью функции запускается сессия, которая используется для хранения данных между запросами. В данном случае, сессия сохраняет текст CAPTCHA, чтобы позже его можно было проверить при вводе пользователем.

Установка заголовка для изображения: код указывает браузеру, что генерируемый контент — это изображение формата PNG. Это необходимо для корректного отображения изображения в браузере.

Создание изображения и генерация цветов

Создание изображения: создаётся новое пустое изображение с заданными размерами (ширина 200 пикселей, высота 70 пикселей). Для этого используется функция, которая создает изображение с поддержкой полного диапазона цветов (True Color).

Генерация случайных цветов: для фона, текста, линий и шума генерируются случайные цвета с помощью функции, которая выбирает случайные значения для каждого канала RGB (красный, зеленый, синий). Это помогает сделать изображение более сложным и трудным для распознавания.

Заполнение фона и добавление случайных элементов

Заполнение фона: на изображении рисуется прямоугольник, который заполняет все пространство фоном с сгенерированным цветом.

Добавление случайных линий: добавляются несколько случайных линий, которые проходят по изображению. Это усложняет задачу боту, так как линии могут пересекать текст и мешать его распознаванию.

Добавление случайных пикселей (шума): рисуются случайные пиксели, которые создают дополнительные помехи и делают изображение еще более сложным для распознавания.

Генерация текста CAPTCHA

Составление случайного текста: составляется строка из случайных символов, выбранных из набора букв и цифр. Длина строки обычно составляет 6 символов.

Сохранение текста в сессии: сгенерированный текст CAPTCHA сохраняется в сессии, чтобы его можно было использовать позже для проверки введенного пользователем текста.

Настройка шрифта и параметров текста

Путь к шрифту: указывается путь к файлу шрифта в формате TTF (TrueType Font), который будет использоваться для отображения текста на изображении. Также проверяется, доступен ли этот шрифт.

Размер и угол наклона шрифта: устанавливается размер шрифта и генерируется случайный угол наклона текста. Это помогает сделать текст менее читаемым и усложнить распознавание.

Генерация координат для каждого символа

Массив для хранения координат: для каждого символа в CAPTCHA рассчитываются координаты его положения на изображении. Эти координаты затем сохраняются в массиве.

Проверка позиции символа: для каждого символа рассчитываются его размеры, и если его размещение выходит за пределы изображения, процесс добавления символов останавливается.

Сдвиг координат: после того как координаты для одного символа определены, для следующего символа координаты сдвигаются, чтобы символы не перекрывались, но при этом оставались в пределах изображения.

Рисование текста на изображении

Рисование каждого символа: после того как координаты всех символов рассчитаны, каждый символ рисуется на изображении с использованием выбранного шрифта и угла наклона. Это выполняется для каждого символа из массива координат.

Вывод изображения в браузер и освобождение ресурсов

Вывод изображения: сгенерированное изображение отправляется в браузер в формате PNG.

Освобождение памяти: после того как изображение отправлено в браузер, освобождаются все ресурсы, связанные с этим изображением, чтобы избежать утечек памяти.

Заключение

Этот код создает изображение CAPTCHA, которое состоит из случайных символов, случайных линий и пикселей. Все эти элементы служат для усложнения задачи распознавания CAPTCHA для автоматических ботов. Каждый символ имеет случайное положение и угол наклона, что делает изображение еще более сложным.

4 Выводы

В ходе работы был разработан и реализован механизм генерации изображения CAPTCHA, который эффективно препятствует автоматическому распознаванию текста с помощью современных алгоритмов машинного обучения и технологий распознавания образов. Созданный алгоритм использует несколько подходов для повышения сложности изображения, включая случайное размещение символов, наклон текста, добавление случайных линий и пикселей, что значительно усложняет задачу для ботов.

Особое внимание уделено таким аспектам:

1. Генерация случайных текстовых символов, что обеспечивало уникальность каждого запроса и затрудняло подбор правильного кода с использованием заранее подготовленных шаблонов.
2. Применение шумовых элементов, таких как случайные линии и пиксели, что повышало уровень помех на изображении, ухудшая возможности для использования OCR (оптического распознавания символов).
3. Гибкость в настройках шрифта — случайный угол наклона и размер шрифта делали текст менее предсказуемым и затрудняли его распознавание.

Преимуществами предложенного решения являются высокая степень защиты от ботов, а также возможность адаптации под различные требования по сложности CAPTCHA в зависимости от нужд системы. Техническая реализация на PHP и использование сессий для хранения проверочного текста позволяют обеспечивать высокий уровень безопасности и защиту от автоматических атак.

В будущем возможна дальнейшая оптимизация алгоритма, например, через использование более сложных методов генерации текста или внедрение адаптивных шифровальных техник для защиты от новых методов распознавания CAPTCHA.

Библиографический список

1. Яковлев А.В., Шуваева А.М., Пеливан М.А. Противодействие атаке brute force на основе графических объектов типа CAPTCHA // Приборы и системы. управление, контроль, диагностика. 2015. №7. С. 15-19.
2. Жданов О.Н. Распознавание современных CAPTCHA // Научный вестник Воронежского государственного архитектурно-строительного университета. Серия: студент и наука. 2014. С. 221-224.
3. Явич М.П. Пирцхалава И.З. Проблемы развития CAPTCHA // Современная техника и технологии. 2015. №7. С. 92-93.
4. Kumar M., Jindal M.K., Kumar M. A Systematic Survey on CAPTCHA Recognition: Types, Creation and Breaking Techniques // Archives of Computational Methods in Engineering State of the Art Reviews. 2021. T. 29. С. 1107-1136.
5. Tariq N., Khan F.A., Moqurrah S.A., Srivastava G. CAPTCHA Types and Breaking Techniques: Design Issues, Challenges, and Future Research Directions // Cryptography and Security. 2023.