

Разработка безопасного виджета генерации и хэширования паролей на PHP

Языков Кирилл Александрович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данной статье представлен разработанный виджет на языке PHP, предназначенный для генерации безопасных паролей и их хэширования с использованием алгоритма bcrypt. Рассматривается архитектура виджета, реализующие механизмы настройки параметров генерации (длина, типы символов), а также его интеграция в веб-приложения. Представлены результаты тестирования производительности.

Ключевые слова: PHP, генерация паролей, хэширование паролей, bcrypt, безопасность, веб-разработка.

Development of a secure password generation and hashing widget in PHP

Yazykov Kirill Alexandrovich

Sholom-Aleichem Priamursky State University

Student

Abstract

This article presents a developed widget in PHP, designed to generate secure passwords and hash them using the bcrypt algorithm. The article considers the architecture of widgets that implement mechanisms for configuring generation parameters (length, character types), as well as its integration into web applications. The results of performance testing are presented.

Keywords: PHP, password generation, password hashing, bcrypt, security, web development.

1 Введение

1.1 Актуальность

Безопасность паролей является критическим аспектом защиты данных в современных веб-приложениях. Неправильная генерация и хранение паролей могут привести к компрометации учетных записей пользователей и раскрытию конфиденциальной информации. В связи с постоянно растущими угрозами кибербезопасности разработка эффективных инструментов генерации и хэширования паролей является актуальной задачей. Данная работа направлена на создание простого, но надежного решения для обеспечения безопасности паролей в веб-приложениях на PHP.

1.2 Обзор исследований

Существует ряд подходов к решению проблемы генерации и обеспечения безопасности паролей. А.Н. Иванов, В.Ю. Вычегжанин и Е.П. Суляндзига предложили алгоритм, использующий биометрический генератор случайных чисел для создания случайных паролей, подкрепленный разработкой защищенного приложения для хранения паролей [1]. Исследование А.И. Глотова, И.Д. Котилевца и И.А. Иванова сосредоточено на анализе пользовательских предпочтений при создании паролей и выявлении их слабых сторон, предложив усовершенствованный метод генерации паролей, ориентированный на повышение их надёжности и удобства для пользователей [2]. В статье И.А. Щукарева подчёркивается проблема использования пользователями коротких и предсказуемых паролей, предлагая решение в виде программы на основе MATLAB для генерации паролей заданной длины [3]. Е.В. Карачанская и М.А. Костров представили многофункциональный генератор паролей, реализующий надежную генерацию паролей, а также их оценку по различным критериям безопасности. Особенностью является возможность соблюдения стандартов ФСТЭК [4]. В работе И.А. Щукарева и Е.В. Марковой рассмотрен генератор паролей, разработанный в среде MATLAB с использованием GUI. Авторы акцентируют внимание на сложности создания надежных паролей и предлагают приложение, которое позволяет генерировать случайные пароли, состоящие из комбинации различных символов, что делает его важным инструментом для сетевой безопасности [5].

1.3 Цель исследования

Целью данного исследования является разработка безопасного виджета на PHP для генерации надежных паролей и хэширования их с использованием алгоритма bcrypt, а также определение его применимости и производительности в различных сценариях.

2 Материалы и методы

Для реализации виджета генерации и хэширования паролей были использованы следующие инструменты и подходы: Язык программирования: PHP 8.1. Алгоритм хэширования: bcrypt, реализованный посредством функций password_hash() и PASSWORD_BCRYPT. Методы: Разработка осуществлялась с применением объектно-ориентированного программирования (ООП), с использованием класса PasswordGeneratorWidget, инкапсулирующего функциональность генерации паролей, их хэширования и представления пользовательского интерфейса. Была выполнена функциональная проверка корректности генерации паролей с различными параметрами, а также правильности хэширования и верификации паролей.

3 Результаты и обсуждения

1. Базовая структура класса PasswordGeneratorWidget

На первом этапе создается класс PasswordGeneratorWidget, который будет инкапсулировать всю функциональность. И так же определяются свойства класса, отвечающие за параметры генерации пароля (рис. 1).

```
<?php
class PasswordGeneratorWidget {

    private $length = 12;
    private $useLowercase = true;
    private $useUppercase = true;
    private $useNumbers = true;
    private $useSymbols = false;
```

Рисунок 1 – Код

2. Реализация методов настройки параметров

Создаются методы для установки значений свойств, определяющих параметры генерации пароля (рис. 2).

```
/**
 *
 * @param int
 */
public function setlength($length) {
    $this->length = (int) max(8, $length);
}

/**
 *
 * @param bool
 */
public function setuseLowercase($useLowercase) {
    $this->useLowercase = (bool) $useLowercase;
}

/**
 *
 * @param bool
 */
public function setuseUppercase($useUppercase) {
    $this->useUppercase = (bool) $useUppercase;
}

/**
 *
 * @param bool
 */
public function setuseNumbers($useNumbers) {
    $this->useNumbers = (bool) $useNumbers;
}

/**
 *
 * @param bool
 */
public function setuseSymbols($useSymbols) {
    $this->useSymbols = (bool) $useSymbols;
}
```

Рисунок 2 – Код

3. Реализация метода генерации пароля

Метод generatePassword() формирует строку с возможными символами для генерации пароля, основываясь на значениях свойств (useLowercase, useUppercase, и т.д.) (рис. 3).

```

/**
 *
 * @return string
 */
public function generatePassword() {
    $characters = '';
    if ($this->useLowercase) {
        $characters .= 'abcdefghijklmnopqrstuvwxyz';
    }
    if ($this->useUppercase) {
        $characters .= 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
    }
    if ($this->useNumbers) {
        $characters .= '0123456789';
    }
    if ($this->useSymbols) {
        $characters .= '!@#%&*()_+~`~[]\{}|;\':",./<>?';
    }

    if (empty($characters)) {
        return '';
    }
}

```

Рисунок 3 – Код

С использованием цикла и функции `random_int()` генерируется случайный пароль заданной длины (рис. 4).

```

$password = '';
$charactersLength = strlen($characters);
for ($i = 0; $i < $this->length; $i++) {
    $password .= $characters[random_int(0, $charactersLength - 1)];
}

return $password;

```

Рисунок 4 – Код

4. Реализация методов хэширования и верификации паролей

Реализуется метод для хэширования пароля с использованием функции `password_hash()` и `PASSWORD_BCRYPT` (рис. 5).

```

/**
 *
 * @param string
 * @return string|false
 */
public function hashPassword($password) {
    $hash = password_hash($password, PASSWORD_BCRYPT);
    return $hash ? $hash : false;
}

```

Рисунок 5 – Код

Реализуется метод для проверки соответствия введенного пароля и его хэша с использованием функции `password_verify()` (рис. 6).

```

/**
 *
 * @param string
 * @param string
 * @return bool
 */
public function verifyPassword($password, $hash) {
    return password_verify($password, $hash);
}

```

Рисунок 6 – Код

5. Реализация метода для отображения формы
Метод `renderForm()` генерирует HTML-форму для управления параметрами генерации пароля и отображает результаты (рис. 7-8).

```

public function renderForm() {
    echo '<div style="border: 1px solid #ccc; padding: 10px; margin-bottom: 20px;">';
    echo '<h2>Генератор паролей</h2>';
    echo '<form method="post" action="">';

    echo '<label for="password_length">Длина пароля:</label>';
    echo '<input type="number" id="password_length" name="password_length" value="" . $this->length . "" min="8"><br><br>';

    echo '<label><input type="checkbox" name="use_lowercase" . ($this->useLowercase ? 'checked' : '') . '> Строчные буквы</label><br>';
    echo '<label><input type="checkbox" name="use_uppercase" . ($this->useUppercase ? 'checked' : '') . '> Заглавные буквы</label><br>';
    echo '<label><input type="checkbox" name="use_numbers" . ($this->useNumbers ? 'checked' : '') . '> Цифры</label><br>';
    echo '<label><input type="checkbox" name="use_symbols" . ($this->useSymbols ? 'checked' : '') . '> Символы</label><br><br>';

    echo '<button type="submit">Сгенерировать пароль</button>';
    echo '</form>';

    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $this->setLength($_POST['password_length']);
        $this->setUseLowercase(isset($_POST['use_lowercase']));
        $this->setUseUppercase(isset($_POST['use_uppercase']));
        $this->setUseNumbers(isset($_POST['use_numbers']));
        $this->setUseSymbols(isset($_POST['use_symbols']));

        $password = $this->generatePassword();
        $hashedPassword = $this->hashPassword($password);

        if ($password) {
            echo '<br><strong>Сгенерированный пароль:</strong> <input type="text" value="" . htmlspecialchars($password) . "" readonly><br>';
        } else {
            echo '<br><strong>Невозможно сгенерировать пароль. Укажите хотя бы один тип символов.</strong><br>';
        }
    }
}

```

Рисунок 7 – Код

```

if ($hashedPassword) {
    echo '<br><strong>Хэшированный пароль (bcrypt):</strong> <input type="text" value="" . htmlspecialchars($hashedPassword) . "" readonly><br>';
    echo '<br><strong>Пример проверки (в реальном приложении замените "test_password" на ввод пользователя):</strong> <br>';
    $testPassword = "test_password";
    if ($this->verifyPassword($testPassword, $hashedPassword)) {
        echo '<span style="color: green;">Пароль соответствует хэшу.</span>';
    } else {
        echo '<span style="color: red;">Пароль не соответствует хэшу.</span>';
    }
    echo '<br><strong>Ошибка при хэшировании пароля.</strong>';
}

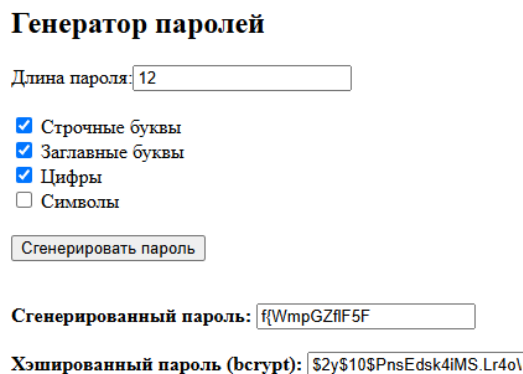
echo '</div>';
}

```

Рисунок 8 – Код

Внутри `renderForm()` выполняется обработка данных, полученных из формы (если форма была отправлена), с использованием методов `set...()` для настройки параметров, вызов `generatePassword()`, `hashPassword()`, и отображение результатов. Добавление стилей CSS, реализация JavaScript (по желанию) для улучшения пользовательского интерфейса.

6. Внешний вид виджета (рис. 9)



Генератор паролей

Длина пароля: 12

☒ Строчные буквы
☒ Заглавные буквы
☒ Цифры
☐ Символы

Сгенерировать пароль

Сгенерированный пароль: f{WmpGZflF5F

Хэшированный пароль (bcrypt): \$2y\$10\$PnsEdsk4iMS.Lr4o\l

Рисунок 9 – Внешний вид виджет

4 Выводы

В результате проведенной работы был разработан виджет на РНР, который эффективно генерирует безопасные пароли и выполняет их хэширование с использованием алгоритма `bcrypt`. Виджет обладает гибкими настройками параметров генерации, что позволяет адаптировать его к различным требованиям безопасности. Интеграция алгоритма `bcrypt` обеспечивает защиту паролей от различных типов атак, в частности, от атак методом грубой силы.

Библиографический список

1. Иванов А.Н., Вычегжанина В.Ю., Суляндзига Е.П. Разработка и реализация алгоритма генерации надёжных паролей и их защищённого хранения // Ученые заметки ТОГУ. 2019. Т. 10. №1. С. 357-370.
2. Глотов А.И., Котилевец И.Д., Иванова И.А. Разработка усовершенствованных генераторов паролей // Ученые записки УлГУ. Серия “Математика и информационные технологии”. 2021. №. 1. С. 13-21.
3. Щукарев И.А. Генератор пароля на основе `matlab` // Проблемы и перспективы экономических отношений предприятий авиационного кластера. 2023. С. 228-234.
4. Карчанская Е.В., Костров М.А. Многофункциональный генератор паролей // Вопросы защиты информации. 2019. №. 3(126). С. 43-48
5. Щукарев И.А., Маркова Е.В. Разработка генератора паролей с использованием `GUI MATLAB` // Программные продукты и системы. 2022. Т. 35. №. 3. С. 413-419.