

Создание программы для генерации паролей на Python

Акентьев Данила Денисович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данной работе представлено приложение для генерации паролей, разработанное с использованием языка программирования Python и библиотеки Tkinter. Основное внимание уделяется обеспечению надежной безопасности личных данных пользователей через создание уникальных и сложных паролей. Программа предоставляет возможность выбора различных параметров для формирования паролей, включая использование символов верхнего и нижнего регистров, чисел и специальных символов. Данное приложение не только способствует улучшению навыков пользователей в области безопасности, но и способствует популяризации использования надежных средств защиты информации.

Ключевые слова: генерация паролей, безопасность данных, Tkinter, Python, программное обеспечение, цифровая безопасность, пользовательский интерфейс.

Creating a program for generating passwords in Python

Akentev Danila Denisovich

Sholom-Aleichem Priamursky State University

Student

Abstract

This paper presents a password generation application developed using the Python programming language and the Tkinter library. The main focus is on ensuring reliable security of users' personal data through the creation of unique and complex passwords. The program provides the ability to select various parameters for forming passwords, including the use of upper and lower case characters, numbers, and special characters. This application not only helps to improve users' security skills, but also promotes the use of reliable information security tools.

Keywords: password generation, data security, Tkinter, Python, software, digital security, user interface.

1 Введение

1.1 Актуальность

С учетом растущих угроз кибербезопасности и увеличения числа случаев утечки личной информации, проблема защиты данных становится крайне актуальной. Параметры использования паролей как первого уровня

защиты требуют постоянного совершенствования, а простота и доступность методов их создания играют важную роль в обеспечении цифровой безопасности.

1.2 Обзор исследований

В работе А.Н. Иванова, В.Ю. Вычегжанина и Е.П. Суляндзига предложен алгоритм, который использует биометрический генератор случайных чисел для создания случайных паролей. Это решение подкреплено разработкой защищённого приложения для хранения паролей, что позволяет одновременно обеспечивать генерацию сложных и надёжных паролей, а также их безопасное хранение. Такой подход особенно важен в эпоху, когда уязвимость паролей становится критическим вопросом [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. Особое внимание уделено плюсам и минусам различных методов авторизации. Основываясь на результатах опросов и исследований, предложили 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, что существенно снижает их устойчивость к атакам. В качестве решения была создана программа на основе MATLAB, которая автоматически генерирует пароли заданной длины. Это позволяет пользователям легко получать сложные пароли, соответствующие современным требованиям безопасности [3].

Другим значимым исследованием является работа авторов Е.В.Карачанской и М.А. Кострова. В этой статье представлен многофункциональный генератор паролей, который реализует надёжную генерацию паролей, а также их оценку по различным критериям безопасности. Особое внимание уделено функционалу, который позволяет пользователю настраивать пароли по своим предпочтениям, включая генерацию паролей на основе заданных слов с усложнением, что значительно повышает удобство и безопасность. Также рассматривается возможность шифрования и хранения сгенерированных паролей, что делает программу универсальной для широкого круга пользователей. Уникальной особенностью является возможность соблюдения стандартов ФСТЭК, что подчеркивает высокую степень надёжности созданных паролей [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 надёжных паролей и предлагают решение в виде приложения, которое позволяет генерировать случайные пароли, состоящие из комбинации различных символов. В статье подробно описываются особенности интерфейса и процесса компиляции программы в автономное приложение, что позволяет использовать его на машинах без установленной среды MATLAB. Подробно рассматривается возможность

использования данного приложения в организациях, где требуется массовая генерация паролей для большого количества пользователей, что делает его важным инструментом для сетевой безопасности [5].

1.3 Цель исследования

Целью данного исследования является создание эффективного инструмента для генерации надежных паролей, который обеспечит пользователям возможность легко и быстро создавать защищенные учетные записи, тем самым способствуя повышению уровня цифровой безопасности.

2 Материалы и методы

В процессе разработки приложения использовались следующие материалы и методы: язык программирования Python, библиотека Tkinter для создания графического интерфейса, методы генерации случайных чисел для обеспечения уникальности и сложности паролей, а также алгоритмические подходы для проверки выбранных пользователем параметров. В ходе реализации программы проводились тестирования на удобство использования и надежность генерируемых паролей с целью подтверждения их соответствия современным требованиям безопасности.

3 Результаты и обсуждения

На первом этапе импортируем необходимые библиотеки. `random` используется для генерации случайных символов, `string` предоставляет предопределенные наборы символов (таких как буквы и цифры), а `tkinter` и `messagebox` позволяют создавать графический интерфейс и выводить сообщения об ошибках пользователю (рис.1).

```
import random
import string
import tkinter as tk
from tkinter import messagebox
```

Рисунок 1. Импорт библиотек

```
def generate_password(length, use_uppercase, use_lowercase, use_numbers, use_special_chars):
    char_sets = ""
    if use_uppercase:
        char_sets += string.ascii_uppercase
    if use_lowercase:
        char_sets += string.ascii_lowercase
    if use_numbers:
        char_sets += string.digits
    if use_special_chars:
        char_sets += string.punctuation

    if not char_sets:
        raise ValueError("Не выбрано ни одного типа символов для генерации пароля.")

    password = ''.join(random.choice(char_sets) for _ in range(length))
    return password
```

Рисунок 2. Генерация пароля

Предложенный код отвечает за генерацию пароля (рис.2). Она принимает следующие параметры:

- `length`: длина запрашиваемого пароля.
- `use_uppercase`, `use_lowercase`, `use_numbers`, `use_special_chars`: булевы значения, указывающие, следует ли включать верхний регистр, нижний регистр, цифры и специальные символы в пароль.

Первым делом, функция формирует строку `char_sets`, которая будет содержать все возможные символы для использования в пароле, в зависимости от заданных параметров. Если ни один тип символов не был выбран, функция выбрасывает исключение, предупреждая пользователя. Затем функция создает пароль, выбирая случайные символы из `char_sets` заданное количество раз (`length`) и возвращает получившуюся строку.

```
def on_generate(): 1 usage
    try:
        length = int(entry_length.get())
        use_uppercase = var_uppercase.get()
        use_lowercase = var_lowercase.get()
        use_numbers = var_numbers.get()
        use_special_chars = var_special_chars.get()

        if length < 1 or length > 128: # Ограничение длины пароля от 1 до 128 символов
            messagebox.showerror( title: "Ошибка", message: "Длина пароля должна быть от 1 до 128 символов.")
            return

        password = generate_password(length, use_uppercase, use_lowercase, use_numbers, use_special_chars)
        entry_password.delete( first: 0, tk.END)
        entry_password.insert( index: 0, password)

    except ValueError as e:
        messagebox.showerror( title: "Ошибка", str(e))
```

Рисунок 3. Функция обработки

Данная функция срабатывает при нажатии кнопки генерации пароля (рис.3). Она выполняет несколько шагов:

Считывает длину пароля и параметры, определяющие, какие типы символов использовать.

Проверяет, что длина пароля находится в пределах от 1 до 128 символов. Если нет, появляется сообщение с ошибкой.

Вызывает функцию `generate_password`, передавая ей необходимые параметры, и получает сгенерированный пароль.

Очищает поле ввода пароля и вставляет в него новый пароль.

В случае возникновения ошибок (например, если пользователь ввел недопустимые данные), выводится сообщение об ошибке.

```
def copy_password(): 1 usage
    password = entry_password.get()
    if password:
        root.clipboard_clear() # Очищаем буфер обмена
        root.clipboard_append(password) # Копируем сгенерированный пароль в буфер обмена
        messagebox.showinfo( title: "Копирование", message: "Пароль скопирован в буфер обмена.")
```

Рисунок 4. Функция копирования пароля

Эта функция отвечает за копирование сгенерированного пароля в буфер обмена (рис.4). Она получает текст из поля ввода пароля. Если пароль существует, буфер обмена очищается, после чего следующая строка кода (которая отсутствует в представленном фрагменте) должна бы добавить этот пароль в буфер обмена для дальнейшего использования.

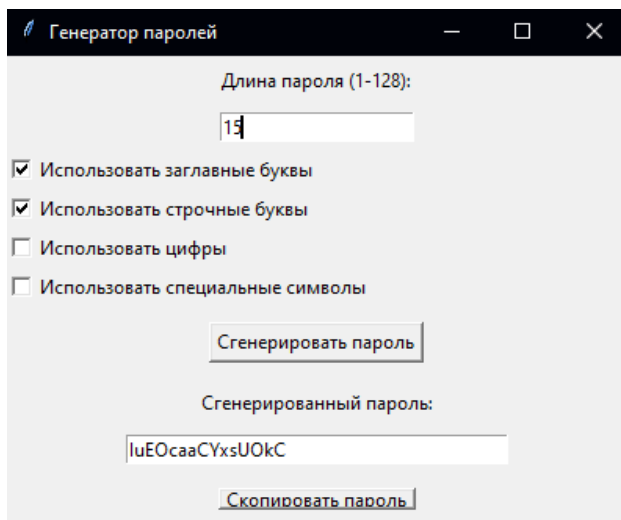


Рисунок 5. Внешний вид приложения

Далее был добавлен внешний вид приложения (рис.5). Код был написан с помощью библиотеки tkinter следующим образом (рис.6-7).

```
# Создаем основное окно
root = tk.Tk()
root.title("Генератор паролей")

# Убираем размеры окна
root.geometry("400x300")

# Поле для ввода длины пароля
label_length = tk.Label(root, text="Длина пароля (1-128):")
label_length.pack(pady=5)
entry_length = tk.Entry(root)
entry_length.pack(pady=5)

# Чекбоксы для выбора типов символов
var_uppercase = tk.BooleanVar()
var_lowercase = tk.BooleanVar()
var_numbers = tk.BooleanVar()
var_special_chars = tk.BooleanVar()

checkbox_uppercase = tk.Checkbutton(root, text="Использовать заглавные буквы", variable=var_uppercase)
checkbox_lowercase = tk.Checkbutton(root, text="Использовать строчные буквы", variable=var_lowercase)
checkbox_numbers = tk.Checkbutton(root, text="Использовать цифры", variable=var_numbers)
checkbox_special_chars = tk.Checkbutton(root, text="Использовать специальные символы", variable=var_special_chars)

checkbox_uppercase.pack(anchor=tk.W)
checkbox_lowercase.pack(anchor=tk.W)
checkbox_numbers.pack(anchor=tk.W)
checkbox_special_chars.pack(anchor=tk.W)

# Кнопка для генерации пароля
button_generate = tk.Button(root, text="Сгенерировать пароль", command=on_generate)
button_generate.pack(pady=10)
```

Рисунок 6. Код внешнего вида приложения

```
# Поле для отображения сгенерированного пароля
label_password = tk.Label(root, text="Сгенерированный пароль:")
label_password.pack(pady=5)
entry_password = tk.Entry(root, width=40) # Увеличиваем ширину поля для пароля
entry_password.pack(pady=5)

# Кнопка для копирования пароля
button_copy = tk.Button(root, text="Скопировать пароль", command=copy_password)
button_copy.pack(pady=10)

# Запуск основного цикла приложения
root.mainloop()
```

Рисунок 7. Код внешнего вида приложения

4 Выводы

В данной работе была разработана простая и эффективная программа для генерации паролей, использующая библиотеку Tkinter для создания интерактивного графического интерфейса. Основная цель приложения заключалась в обеспечении пользователей инструментом для создания надежных паролей, которые соответствуют требованиям безопасности в цифровом пространстве.

Программа предоставляет пользователю возможность выбирать различные типы символов для формирования паролей, включая верхний и нижний регистры, цифры и специальные символы. Это достигается с помощью логического подхода в функции генерации, который включает в себя проверку выбранных параметров и динамическое формирование набора символов. Подходящее использование случайности и разнообразия символов позволяет создавать пароли, которые значительно уменьшают риск их предсказания или вычисления злоумышленниками.

Проект иллюстрирует важность использования уникальных и сложных паролей для защиты личной информации, а также демонстрирует возможности библиотеки Tkinter как инструмента для разработки пользовательских интерфейсов. С помощью данного приложения пользователи могут легко и быстро генерировать надежные пароли, что подчеркивает актуальность темы безопасности в информационных технологиях.

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.