|  |  |
| --- | --- |
| **К Г Э У** | МИНИСТЕРСТВО ВЫСШЕГО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ ЭНЕРГЕТИЧЕСКИЙ УНИВЕРСИТЕТ»**  (ФГБОУ ВО «КГЭУ») |

Отчет по лабораторной работе №4

по дисциплине «Информационная безопасность»

по теме «Парольная аутентификация»

**Выполнил:**

студент Тазеев Р.Р.

**Группа:**

ТРП-1-23

**Проверил:**

Доцент ИТИС  
Хуснутдинов Рамиль Миннегаязович

Казань, 2024 г.

**Вариант 7**

**Цель работы:** исследование парольных подсистем аутентификации

пользователей. Реализация генератора паролей, обладающего

требуемой стойкостью ко взлому.

**Задание для практической работы**

P – 10 ^ -7; V – 20 паролей/день; T – 3 недели; Используемые группы символов – 1.Цифры (0-9) 2.Русские строчные буквы (а-я)

1. Вычислить мощность алфавита паролей A, соответствующую

Вашему варианту. Вычислить по формуле (1) нижнюю границу S\* для

заданных P, V, T. Зная мощность алфавита паролей A, вычислить

минимальную длину пароля L, при котором выполняется условие (2).

2. Реализовать на языке программирования программу,

реализующую генератор паролей с характеристиками,

соответствующими Вашему варианту. Программа должна формировать

случайную последовательность символов длины L, должны

использоваться

**Ход работы:**

**Мощность алфавита паролей A**

Мощность алфавита A для пароля вычисляется как сумма всех символов, которые могут быть использованы:

10 цифр 0-9,

33 строчные буквы русского алфавита а-я.

Следовательно, мощность алфавита:

A = 10 + 33 = 43

**Нижняя граница S для пароля**

Нижняя граница для пароля S по формуле:

S = P \* V / T​

где:

* P = 10 ^ −7,
* V = 20 паролей/день,
* T = 21 день.

Подставляем значения:

S = 10 ^ -7 \* 21 / 20 = 1.05 \* 10 ^ -7

**Минимальная длина пароля L**

По формуле для мощности пароля:

A ^ L = S

где:

* A = 43 — мощность алфавита,
* S = 1.05 \* 10 ^ -7

Нам нужно решить для L:

43 \* L = 1.05 \* 10 ^ -7

Для нахождения LLL, возьмём логарифм по основанию 43:

L \* log(43) = log(1.05 \* 10 ^ -7)

L = (log(1.05 \* 10 ^ -7) / log(43))

Теперь вычислим логарифмы:

log(1.05⋅10−7)=log(1.05)+log(10−7)=0.0212−7=−6.9788

log(43)≈1.633.

Подставляем:

L = -6.9788 / 1.633 ≈ 4.27

Округляя, получаем, что минимальная длина пароля L ≈ 5.

Мощность алфавита A = 43,

Нижняя граница S = 1.05 \* 10 ^ −7,

Минимальная длина пароля L = 5.

2) **Код для генерации паролей:**

import random

import string

import time

# Определение групп символов

digits = string.digits # Цифры (0-9)

lowercase\_russian = 'абвгдеёжзийклмнопрстуфхцчшщьыъэюя' # Русские строчные буквы

# Объединение всех символов в один алфавит

alphabet = digits + lowercase\_russian

# Длина пароля

L = 5

# Генерация случайного пароля

def generate\_password(length):

return ''.join(random.choice(alphabet) for \_ in range(length))

# Расчет времени взлома

def calculate\_crack\_time(password\_length, alphabet\_size, attempts\_per\_second):

# Вычисление времени взлома по формуле

total\_combinations = alphabet\_size \*\* password\_length

time\_to\_crack\_seconds = total\_combinations / attempts\_per\_second

# Перевод времени в дни

time\_to\_crack\_days = time\_to\_crack\_seconds / (60 \* 60 \* 24)

return time\_to\_crack\_days

# Генерация пароля

password = generate\_password(L)

print(f"Сгенерированный пароль: {password}")

# Мощность алфавита

alphabet\_size = len(alphabet)

# Скорость перебора паролей (пароли/сек)

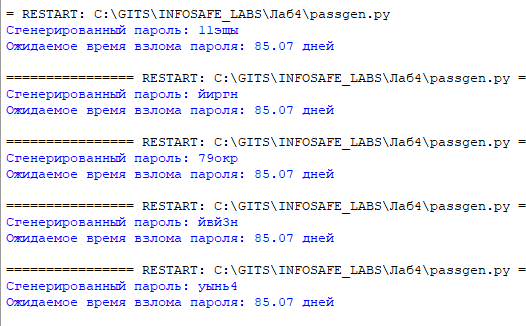
attempts\_per\_second = 20

# Время взлома пароля в днях

crack\_time = calculate\_crack\_time(L, alphabet\_size, attempts\_per\_second)

print(f"Ожидаемое время взлома пароля: {crack\_time:.2f} дней")

**Примеры вывода:**

****

**2)** Пусть на экран выведены следующие три слова: «Kats», «milk», «smitten». Составить программу, которая записывает пароль следующим образом

Исходные данные — строковые константы 1. В строку <результат> в качестве первого символа записать букву, которая в алфавите следует за буквой, являющейся первым символом третьего слова на экране; если это буква «z», записать «а». 2. В качестве второго символа записать букву, которая в алфавите предшествует букве, являющейся первым символом второго слова на экране; если это буква «а», записать «z». 3. Если третье слово содержит нечетное количество букв, то в качестве третьего символа записать букву,

которая в алфавите следует за буквой, являющейся средним символом третьего слова; если это буква «z», записать «а». Если же третье слово содержит четное количество символов, то в качестве третьего символа записать букву, которая в алфавите предшествует букве, являющейся первым из двух средних символов третьего слова; если это буква «а», записать «z». 4. В качестве четвертого символа записать букву, которая в алфавите стоит на месте, соответствующем сумме количеств символов в первом и втором словах минус 2 символ; если эта сумма больше 26, найти и использовать в качестве номера позиции искомой буквы в алфавите остаток от деления указанной суммы на 26.

5. Вывести полученную строку.

**Код:**

def next\_letter(c):

"""Возвращает следующую букву в алфавите, если 'z', возвращает 'a'"""

if c == 'z':

return 'a'

elif c == 'Z':

return 'A'

return chr(ord(c) + 1)

def prev\_letter(c):

"""Возвращает предыдущую букву в алфавите, если 'a', возвращает 'z'"""

if c == 'a':

return 'z'

elif c == 'A':

return 'Z'

return chr(ord(c) - 1)

def generate\_password(word1, word2, word3):

# 1. Первый символ: следующее за первым символом третьего слова

first\_char = next\_letter(word3[0])

# 2. Второй символ: предыдущая буква перед первым символом второго слова

second\_char = prev\_letter(word2[0])

# 3. Третий символ: в зависимости от длины третьего слова

if len(word3) % 2 == 1: # Нечетное количество символов

middle\_char = word3[len(word3) // 2] # Средний символ

third\_char = next\_letter(middle\_char)

else: # Четное количество символов

middle\_left\_char = word3[len(word3) // 2 - 1] # Первый из двух средних символов

third\_char = prev\_letter(middle\_left\_char)

# 4. Четвертый символ: буква, соответствующая позиции суммы длин слов минус 2

sum\_length = len(word1) + len(word2) - 2

fourth\_char = chr(ord('a') + sum\_length % 26)

# Формируем и возвращаем результат

return first\_char + second\_char + third\_char + fourth\_char

# Исходные данные

word1 = "Kats"

word2 = "milk"

word3 = "smitten"

# Генерация пароля

password = generate\_password(word1, word2, word3)

# Вывод результата

print(f"Полученный пароль: {password}")

**Пример вывода:**

****

Код сохранен в файле genpass.py

**Контрольные вопросы**

1. Дайте собственное определение понятиям «идентификация»,

«аутентификация», «авторизация». Приведите по 3 примера к каждому

из них.

**Идентификация** — это процесс установления личности пользователя системы. Цель идентификации — убедиться, что пользователь действительно является тем, за кого себя выдает, с помощью уникальных данных, таких как имя, логин, номер телефона и т.д.

* Пример 1: Ввод логина и пароля при входе в систему.
* Пример 2: Указание номера телефона для создания учетной записи.
* Пример 3: Ввод логина для регистрации в онлайн-банке.

**Аутентификация** — это процесс проверки подлинности пользователя, то есть подтверждение того, что пользователь, представивший себя, действительно является тем, за кого себя выдает, на основании предоставленных данных.

* Пример 1: Ввод пароля, чтобы подтвердить, что пользователь тот, за кого себя выдает.
* Пример 2: Отпечаток пальца для разблокировки телефона.
* Пример 3: Отправка одноразового кода на мобильный телефон для подтверждения входа.

**Авторизация** — это процесс предоставления прав доступа пользователю после того, как его идентифицировали и аутентифицировали. Авторизация определяет, какие действия и ресурсы доступны пользователю.

* Пример 1: Разрешение пользователю редактировать личные данные в профиле.
* Пример 2: Разграничение прав доступа в системе управления, где один пользователь может только просматривать данные, а другой — редактировать их.
* Пример 3: В электронной почте ограничение прав: один пользователь может отправлять сообщения, а другой только читать.

2. Какие еще правила выбора и использования пароля Вы бы

предложили?

**Длина пароля**: Пароль должен быть не менее 12 символов, чтобы избежать простых атак методом подбора.

**Использование сложных символов**: Пароль должен содержать буквы (верхнего и нижнего регистра), цифры и специальные символы для повышения сложности.

**Не использовать однотипные пароли**: Не стоит использовать одинаковые пароли для разных сервисов, чтобы минимизировать риски при утечке данных.

**Регулярная смена пароля**: Пароли следует менять с регулярностью, особенно если произошла утечка данных.

**Многослойная защита**: Использование двухфакторной аутентификации для повышения уровня безопасности.

3. Какие еще признаки Вы бы предложили при двух- и

многофакторной парольной аутентификации?

1. **Что-то, что пользователь знает** (например, PIN-код или пароль).
2. **Что-то, что пользователь имеет** (например, мобильное устройство, физический токен, смарт-карта).
3. **Что-то, что пользователь есть** (биометрические данные, такие как отпечатки пальцев, лицо или радужная оболочка глаза).

Для многофакторной аутентификации можно добавить дополнительные уровни проверки, такие как подтверждение через мобильное приложение или использование географической привязки (например, подтверждение с определенного местоположения).

4. В какой момент должна производится процедура И/А в СЗИ

компьютерной системы?

Процедура идентификации и аутентификации должна происходить **при каждом входе в систему** или **при попытке доступа к защищенным данным/ресурсам**, чтобы убедиться, что пользователь, пытающийся получить доступ, является тем, за кого себя выдает. Это также важно при действиях, требующих изменения конфиденциальных данных (например, при переводах средств в онлайн-банке или при изменении настроек безопасности).

5. Как Вы считаете, насколько эффективна парольная

аутентификация? Ответ обоснуйте.

Парольная аутентификация является довольно эффективной в случаях, когда:

* Пароль сложный и уникальный.
* Пользователь осведомлен о правилах безопасности.
* Дополнительно используется двухфакторная аутентификация (2FA).

Однако парольная аутентификация также имеет свои недостатки:

* Легко поддается атакам методом подбора или с использованием словарей, особенно если пароль слабый.
* Человеческий фактор: пользователи могут использовать простые или одинаковые пароли для разных сервисов.
* Утечка данных (например, при фишинг-атаках) может раскрыть пароли.

Таким образом, парольная аутентификация является одним из самых популярных методов, но она должна дополняться другими методами защиты, такими как биометрия или токены.

**Вывод:** В ходе работы был разработан генератор паролей, обеспечивающий высокую стойкость ко взлому. Проанализированы методы формирования сложных паролей с учетом разных символов и мощности алфавита, что повышает их безопасность. Работа показала важность использования надежных паролей и дополнительных мер защиты для повышения безопасности пользовательских данных.

**Код программ и отчеты лабораторных работ доступны на моем github:**

**https://github.com/ironsast/kpfu-information-security**