Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Пермский национальный исследовательский

политехнический университет»

(ПНИПУ)

Кафедра вычислительной математики и механики

**Лабораторная работа № 1**

по дисциплине:

«**Информационная безопасность и защита информации**»

по теме: «Анализ надежности пароля»

Выполнил:

Студент группы ИСТ-19-1бзу

Соболь Е. В.

Проверил:

доцент кафедры ВММБ

Труфанов А. Н.

Пермь

2022 г.

**Цель работы:**

Ознакомиться с основными методами криптографической защиты информации. Получить практические создания модуля оценки надёжности пароля.

**Описание алгоритма проверки надежности пароля:**

Нагляднее надёжность пароля можно оценить с помощью времени, необходимого для перебора всех возможных комбинаций символов. Число возможных паролей при выбранной мощности алфавита определяется длиной пароля L из соотношения S = NL. Величина S называется мощностью пространства паролей. Для пароля, состоящего из заглавных (или только строчных) букв латинского алфавита, N = 26. Если пароль состоит из заглавных и строчных латинских букв, то N = 52. Для пароля, состоящего из цифр и строчных (или заглавных) букв А = 36. В случае если для создания пароля использованы все печатные символы кодовой таблицы (включая знаки препинания и специальные символы), мощность алфавита N = 95. При скорости перебора паролей V необходимое для перебора всех паролей составляет T = S/V.

**Основная часть:**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Мощность алфавита | Длина пароля | Количество комбинаций |
| 1 | 10 | 6 | 1000000 |
| 2 | 10 | 7 | 10000000 |
| 3 | 10 | 8 | 100000000 |
| 4 | 26 | 6 | 308915776 |
| 5 | 26 | 7 | 8031810176 |
| 6 | 26 | 8 | 208827064576 |
| 7 | 36 | 6 | 2176782336 |
| 8 | 36 | 7 | 78364164096 |
| 9 | 36 | 8 | 2821109907456 |
| 10 | 95 | 6 | 735091890625 |
| 11 | 95 | 7 | 69833729609375 |
| 12 | 95 | 8 | 6634204312890625 |

**Листинг:**

main.py

from PwdStrengthAnalysis import PwdStrengthAnalysis  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 pwd = input("Введите пароль: ")  
 s = float(input("Введите скорость перебора паролей в секунду (s): "))  
 m = float(input("Введите количество неправильных попыток (m): "))  
 v = float(input("Введите время паузы в секундах (v): "))  
 a = PwdStrengthAnalysis(pwd, s, m, v)  
 a.get\_format\_time(a.analyze\_the\_password())

PwdStrengthAnalysis.py

class PwdStrengthAnalysis:  
  
 def \_\_init\_\_(self, password, s, m, v):  
 self.password = password  
 self.password\_length = len(password)  
 self.s = s  
 self.m = m  
 self.v = v  
 self.n = self.\_\_class\_\_.\_\_alphabet\_power(password)  
 self.pwd\_space\_power = self.n \*\* self.password\_length  
 print(f"M = {self.pwd\_space\_power}")  
  
 \_\_complexity\_of\_character\_types = {  
 'symbol': 33,  
 'small': 26,  
 'capital': 26,  
 'number': 10  
 }  
  
 @staticmethod  
 def \_\_define\_symbol(symbol):  
 ascii\_code = ord(symbol)  
 if ((32 <= ascii\_code) and (ascii\_code <= 47)) or \  
 ((58 <= ascii\_code) and (ascii\_code <= 64)) or \  
 ((91 <= ascii\_code) and (ascii\_code <= 96)) or \  
 ((123 <= ascii\_code) and (ascii\_code <= 126)):  
 return True  
 return False  
  
 @staticmethod  
 def \_\_define\_number(symbol):  
 ascii\_code = ord(symbol)  
 if (48 <= ascii\_code) and (ascii\_code <= 57):  
 return True  
 return False  
  
 @staticmethod  
 def \_\_define\_capital\_latin(symbol):  
 ascii\_code = ord(symbol)  
 if (65 <= ascii\_code) and (ascii\_code <= 90):  
 return True  
  
 return False  
  
 @staticmethod  
 def \_\_define\_small\_latin(symbol):  
 ascii\_code = ord(symbol)  
 if (97 <= ascii\_code) and (ascii\_code <= 122):  
 return True  
  
 return False  
  
 @classmethod  
 def \_\_alphabet\_power(cls, password):  
 alphabet = list(set(password))  
 n = 0  
 type\_symbol = []  
 for symbol in alphabet:  
 if cls.\_\_define\_symbol(symbol):  
 type\_symbol.append('symbol')  
 if cls.\_\_define\_number(symbol):  
 type\_symbol.append("number")  
 if cls.\_\_define\_small\_latin(symbol):  
 type\_symbol.append("capital")  
 if cls.\_\_define\_capital\_latin(symbol):  
 type\_symbol.append("small")  
 type\_symbol = list(set(type\_symbol))  
 for i in type\_symbol:  
 n += cls.\_\_complexity\_of\_character\_types[i]  
 print(f"N = {n}")  
 return n  
  
 def \_\_waiting\_time\_for\_incorrect\_input(self):  
 result = (self.pwd\_space\_power // self.m) \* self.v  
 print(f"Время потраченное на ожидание:", result)  
 return result  
  
 @staticmethod  
 def get\_format\_time(sec):  
 t\_dict = {  
 "year": 31536000,  
 "month": 2592000,  
 "days": 86400,  
 "hours": 3600,  
 "minuts": 60  
 }  
  
 for key in t\_dict:  
 temp = t\_dict[key]  
 t\_dict[key] = sec // temp  
 sec -= t\_dict[key] \* temp  
  
 print(  
 f"ИТОГ: "  
 f"{t\_dict['year']} л. {t\_dict['month']} мес. "  
 f"{t\_dict['days']} д. {t\_dict['hours']} ч. "  
 f"{t\_dict['minuts']} мин. {sec} сек."  
 )  
  
 def analyze\_the\_password(self):  
 print(self.pwd\_space\_power, " ", self.s)  
 t = self.pwd\_space\_power / self.s  
 print(f"Время активного перебора пароля: {t} сек.")  
 result\_sec = self.\_\_waiting\_time\_for\_incorrect\_input() + t  
 print("Общее время перебора (с учётом ожиданий): ", result\_sec)  
 return result\_sec