

Password-Manager

Im heutigen Internet-Umfeld benötigen die Nutzer im Alltag eine Vielzahl unterschiedlicher Passwörter. Damit diese ausreichend sicher sind, sollten sie möglichst lang und komplex sein und zudem nicht mehrfach verwendet werden. Diese Bedingungen machen es schwierig, einen Überblick über die eigenen Passwörter zu behalten. Ein Passwort-Manager kann hierbei Abhilfe schaffen.

Konzept

Ein fähiges Password-Manager braucht viele Funktionen und Eigenschaften, die ein weites Spektrum von Diensten anbieten.

In Bezug auf den Anforderungen haben wir verschiedene *Funktionen* entwickelt.

Man unterscheidet zwischen Primär und Support-Funktionen

Primäre Funktionen

- Ein Passwort generieren und speichern.
- Ein Passwort kopieren.
- Ein Passwort löschen
- Alle Passwörter anzeigen.
- Alle Passwörter löschen.

Support-Funktionen

- Password-Generator
- Daten ver/ent-schlüsseln.
- Master-Passwort und Timestamps.

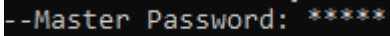
Support Funktionen

1) Master-Passwort und Timestamps

Jede Primär-Funktion vom Password-Manager ist durch ein Master-Passwort gesichert.

Nur Bei richtiger Eingabe darf der Nutzer seine Funktionalitäten benutzen. Beim Eintippen lässt das Modul *stdiomask* das Master-Passwort nicht im Klartext erscheinen.

```
master_password_input = stdiomask.getpass("--Master Password: ")
```



Das Master-Passwort wird nicht bei jeder Abfrage angefordert, sondern wird nach einer einmaligen Eingabe für eine vom Benutzer festgelegten Zeit nicht abgefragt. (5 Minuten zum Beispiel)

Für solche Funktionalität sind **Timestamps** notwendig.

```
if not os.path.isfile("timestamp.txt"):
    timefile = open("timestamp.txt", 'w')
    timefile.write("2021-06-11 18:42:16.058009")
    timefile.close()
timefile = open("timestamp.txt", 'r')
timestampstr = timefile.readline()
timestamp = datetime.datetime.strptime(timestampstr, "%Y-%m-%d %H:%M:%S.%f")
timedelta = datetime.datetime.now() - timestamp
```

- Das Programm schreibt in einem "Timestamp.txt" File den jetzigen Timestamp. *Formatbeispiel: 2021-06-11 18:42:16.058009*
- Dann erstellt es eine End-Variable, die der Endzeit der festgelegten Periode entspricht.
- Zunächst rechnet das Programm die Differenz zwischen die Endzeit und die Startzeit.
- Wenn kein "Timestamp.txt" File existiert, wird ein neues erstellt.

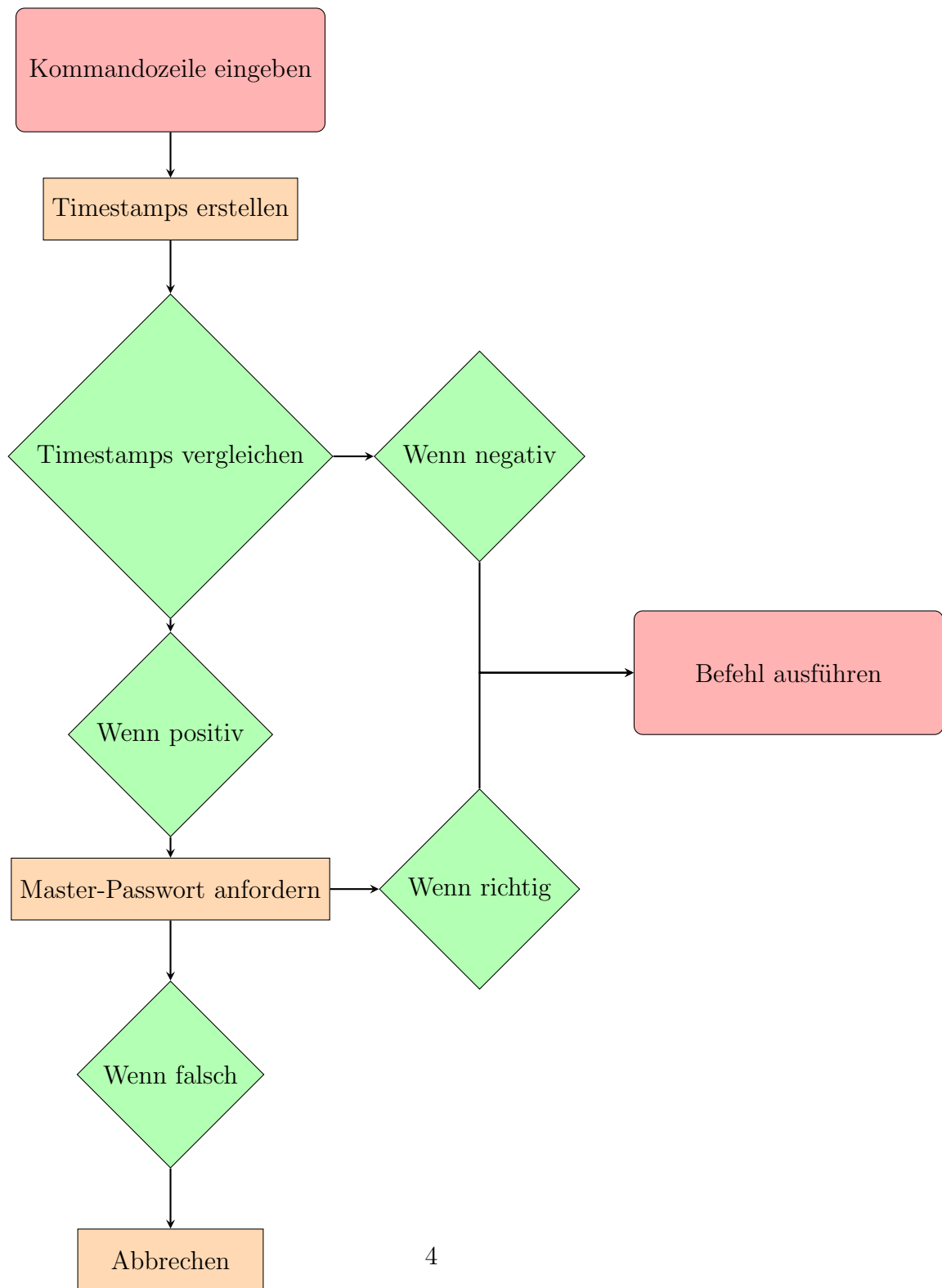
Durch Kombination von dem Master-Passwort und Timestamps ist die Einsetzung eines Sicherheitssystems für das Programm möglich.

kombinierter Code:

```
if not os.path.isfile("timestamp.txt"):
    timefile = open("timestamp.txt", 'w')
    timefile.write("2021-06-11 18:42:16.058009")
    timefile.close()
timefile = open("timestamp.txt", 'r')
timestampstr = timefile.readline()
timestamp = datetime.datetime.strptime(timestampstr, "%Y-%m-%d %H:%M:%S.%f")
timedelta = datetime.datetime.now() - timestamp
if timedelta > datetime.timedelta(minutes=1):
    master_password_input = stdiomask.getpass("--Master Password: ")
    if master_password_input == master_password:
        timefile = open("timestamp.txt", 'w')
        timefile.write(str(datetime.datetime.now()))
        --FUNKTION AUSFÜHREN
else:
    timefile = open("timestamp.txt", 'w')
    timefile.write(str(datetime.datetime.now()))
    --FUNKTION AUSFÜHREN
```

- Ein positives Ergebnis führt zu keiner Abfrage des Master-Passwortes.
- Ein negatives Ergebnis führt zu einer Abfrage des Master-Passwortes.

Dieser Code wird bei jedem primären Befehl verwendet, wird aber im Quellcode nicht jedes Mal gezeigt.



2 - Passwörter generieren

```
lower = string.ascii_lowercase
upper = string.ascii_uppercase
numbers = string.digits
symbols = string.punctuation

password = lower + upper + numbers + symbols

temp = random.sample(password, password_length)
password = "".join(temp)
return password
```

Der obere Code bezeichnet die vorinitialisierte Strings, die man als Stringkonstante verwendet.

In Python ergibt die Zeichenfolge :

- ASCII-lowercase: die Kleinbuchstaben 'abcdefghijklmnopqrstuvwxyz'.
- ASCII-uppercase: die Großbuchstaben 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'.
- Digits: alle Zahlen.
- Punctuation: alle Symbole der ASCII-Tabelle.

Das Passwort besteht dann aus Klein-/Großbuchstaben, Zahlen und Symbolen.

sample() ist eine eingebaute Funktion des *random* Moduls, die eine Liste von Elementen einer bestimmten Länge zurückgibt, in diesen Fall die eingegebenen Passwortlänge. Die *join()* Funktion fügt diese dann in einem String zusammen. Anschließend ist das Passwort generiert.

3) Passwort verschlüsseln / *Encrypt*

Diese Funktion funktioniert laut dem Prinzip der *Caesar Verschlüsselung*. jeder Buchstabe soll durch einen anderen Buchstaben (mit einer Rechtsverschiebung) ersetzt werden.

```
encrypted = ""
for i in range (len(data)):
    char = data[i]
    if (char.isupper()):
        encrypted += chr((ord(char) + shift - 65) % 26 + 65 )
    elif (char.islower()):
        encrypted += chr((ord(char) + shift - 97) % 26 + 97 )
    elif (char.isdigit()):
        number = (int(char) + shift) % 10
        encrypted += str(number)
    else:
        encrypted += char
return encrypted
```

Ausführung der Verschlüsselung:

- Verteilung der Datei in einer **chars** Liste
- Jede Buchstabe verschiebt sich um eine bestimmte Zahl 'Shift = 5' im Alphabet.
- Alle verschobene Buchstaben wieder zu einem verschlüsselten Wort addieren.

Beispiel:

- Wort : **abc** Key = 1

Entschlüsselung :

a + 1 = b

b + 1 = c

c + 1 = d

Entschlüsseltes Wort : **bcd**

4) Passwort entschlüsseln / *Decrypt*

Genau wie die *Encrypt* Funktion, funktioniert die *Decrypt* laut dem Prinzip der *Caesar Verschlüsselung*. Sie ist zwar die umgekehrte Version von *Encrypt* (eine Linksverschiebung).

```
def decrypt(data, shift):
    decrypted = ""
    for i in range (len(data)):
        char = data[i]
        if (char.isupper()):
            decrypted += chr((ord(char) - shift - 65) % 26 + 65 )
        elif (char.islower()):
            decrypted += chr((ord(char) - shift - 97) % 26 + 97 )
        elif (char.isdigit()):
            number = (int(char) - shift) % 10
            decrypted += str(number)
        else:
            decrypted += char
    return decrypted
```

Ausführung der Entschlüsselung:

- Verteilung der Datei in einer **chars** Liste .
- Jede Buchstabe schiebt sich um eine bestimmt Zahl 'Shift = 5' im Alphabet zurück.
- Dann werden alle Buchstaben wieder zu einem entschlüsselten Wort addiert.

Beispiel:

Verschlüsseltes Wort : **bcd** Key = 1

Entschlüsselung :

b - 1 = a

c - 1 = b

d - 1 = c

Entschlüsseltes Wort : **abc**

Primäre Funktionen

1) Passwörter generieren und speichern

Kommandozeile: add - Titel - Benutzername - Passwortlänge

```
C:\Users\Plotzko\PycharmProjects\pythonProject2>main.py add Moodle Student 10
--Master Password: *****
Password generated and copied to Clipboard
```

```
shift = 5
file = open("passwords.txt", 'a')
password = generator(password_length)
pyperclip.copy(password)
file.write(
    encrypt(titel,shift) + " "*(25-len(encrypt(titel,shift))) + "|" +
    encrypt(name,shift) + " "*(25 - len(encrypt(name,shift)))+ "|" +
    encrypt(password,shift) + " "*(4-len(encrypt(name,shift))) + "\n")
print("Password generated and copied to Clipboard")
```

Die Funktion generiert mithilfe der *generator* Methode ein zufälliges Passwort und kopiert es mit *pyperclip.copy()* Methode an die Zwischenablage des Computers.

Zunächst verschlüsselt das Programm die Daten (Titel, Benutzername, Passwort) durch die *Encrypt* Funktion und speichert die im "Passwords.txt" File.

2) Passwörter kopieren

Kommandozeile: copy - Titel

```
C:\Users\Plotzko\PycharmProjects\pythonProject2>main.py copy Twitter
Benutzername:Admin
Password copied to clipboard.
```

```
shift = 5
found = False
zeilen = file.readlines()
for rows in zeilen:
    column = rows.split('|')
    titel = decrypt(column[0],shift)
    if x.lower() in titel.lower():
        print("Benutzername:" + decrypt(column[1],shift))
        pyperclip.copy(decrypt(column[2][column[2].index(''):],shift))
        print("Password copied to clipboard.")
        found = True
if found == False:
    print("Title not found.")
```

Diese Funktion ist eine Vervollständigung der Support-Funktion: kopieren(x)

Das Programm öffnet den File "Passwords.txt" und nach dem eingegebenen Titel suchen. In seinem rohen Zustand, wäre diese Suche nicht möglich. Deswegen muss das Programm den Inhalt des Files mithilfe der *Decrypt* Funktion entschlüsseln.

- Wenn nicht gefunden, zeigt das Programm eine "Title not found." Nachricht.
- Wenn gefunden, kopiert das Programm das gehörigen Passwort und gibt die Nachricht "Password copied to clipboard." zum Benutzer aus.

3) Ein Passwort löschen

Kommandozeile: delete - Titel

```
C:\Users\Plotzko\PycharmProjects\pythonProject2>main.py delete Twitter
Press Y for Yes or else for No.
Do you really want to delete: Twitter
y
Done.
```

```
with open("passwords.txt", "r+") as file:
    print("Press Y for Yes or else for No.")
    choice = input("Do you really want to delete: " + str(x) + "\n")
    if choice == "y":
        lines = file.readlines()
        file.seek(0)
        for line in lines:
            if encrypt(x.lower(), shift) not in line.lower():
                file.write(line)
            else:
                found = True
                print("Done.")
        file.truncate()
        if found == False:
            print("Title not found.")
    else:
        print("Cancelled.")
```

Das Programm fragt nach einer Bestätigung des Benutzers und fordert eine Eingabe von:

- **Y**, um den Prozess fortzusetzen.
- **else**, um den Prozess abubrechen.

Im Bestätigungsfall löscht das Programm die Zeile des eingegebenen Titels. Wenn der Titel nicht existiert, wird die Meldung "*Title not found.*" angezeigt.

4) Passwörter bzw.Daten anzeigen

Kommandozeile: `display - Titel`

```
C:\Users\Plotzko\PycharmProjects\pythonProject2>main.py display
--Master Password: *****
Facebook          |Nizar            |bIH%"+,9A$/}{M:.rqy
Facebook2         |student          |>U#AC
Twitter           |Admin            |$<`N2YlX~QWtpjakP%Hq
```

```
shift = 5
file = open("passwords.txt", 'r')
for i in file:
    data = i.split("|")
    print(decrypt(data[0], shift) + " " * (25 - len(data[0])) + "|" +
          decrypt(data[1], shift) + " " * (25 - len(data[1])) + "|" +
          decrypt(data[2], shift) + " " * (4 - len(data[2]))))
```

Nach Eingabe der Kommandozeile:

- Speichert das Programm durch eine *for-Schleife* die Daten in einer Liste, entschlüsselt sie Mithilfe der *Decrypt* Funktion und zeigt die originale (entschlüsselte) Version im Bildschirm an.

FORMAT: Titel — Benutzernamen — Password

5) Alle Passwörter löschen

Kommandozeile: clear

```
C:\Users\Plotzko\PycharmProjects\pythonProject2>main.py clear
--Master Password: *****
Press Y for Yes or else for No.
Do you really want to clear all your passwords ?
y
All passwords cleared.
```

```
master_password_input = stdiomask.getpass("--Master Password: ")
if master_password == master_password_input:
    print("Press Y for Yes or else for No.")
    choice = input("Do you really want to clear all your passwords ? \n ")
    if choice == "y":
        os.remove("passwords.txt")
        print("All passwords cleared.")
    else:
        print("Canceled.")
else:
    print("Access denied. Wrong Master Password.")
```

Dieser Befehl hat das gleiche Konzept wie der Befehl *"delete"*. Der Unterschied ist, dass er, nicht nur eins, sondern alle Passwörter Bzw. Daten löscht. Das Programm löscht also das *Passwords.txt* File.

Im Gegensatz zu den anderen Funktionen, die nur eine periodische Master-Password Eingabe brauchen, ist die Eingabe des Master-Passwordes und die **Y** Bestätigung immer erforderlich, um diesen Befehl zu verwenden.



Schlusswort

Deutschland hat ein Passwortproblem. Das Hasso-Plattner-Institut (HPI) veröffentlicht jedes Jahr eine Liste der beliebtesten Passwörter der Deutschen. Die Datengrundlage für die 2019er Analyse bilden rund 67 Millionen Zugangsdaten, die in Kombination mit deutschen Mailadressen öffentlich im Internet zugänglich waren. Besorgniserregend ist, dass das häufigste Passwort deutscher Nutzer auch weit im 21. Jahrhundert immer noch "123456" ist. Mit dem Passwort-Manager Programm ist diese Problem zu vermeiden/verhindern. Es bietet eine große Bandbreite und Funktionsvielfalt. Einige von ihnen waren vielleicht nicht einfach umzusetzen, aber sie sind es absolut wert, da sie so viel Wert auf das Endprodukt legen. Einer der größten Vorteile ist, dass dieses Programm immer aktualisiert und verbessert werden kann.

Von:

- Kaouri Nizar
- Emre Bugday
- Jammal Mahmoud

Fach: Betriebssysteme und Rechnernetze (SS2021)

Dozent: Prof. Dr. Christian Baun