密碼「資」多少

106 33 鄭書帆

目錄

核心動機

享受數位科技便利的同時，我們從未認真檢視過身邊的數位漏洞，因上次參加交大電機一日營就對程式有初步的了解和興趣，加上在新聞上看到密碼外洩的問題而造成被取帳號和錢財，高度依賴 Google、Instagram等多元線上平台，每個人都擁有數個帳號，因此想進行探討。

密碼是守護個人隱私的第一道防線，我觀察到身邊許多人習慣使用單一、連續或是用生日作為密碼，為了驗證這個現象，我設計 Google 表單進行習慣調查，並經由調查結果證實，多數人在『密碼習慣』上存在漏洞（例如：密碼重複率高），這促使我想透過本次自主學習，深入探究密碼安全機制與程式設計，並思考如何讓大眾提升數位生活中的安全。

目標：利用客觀的 Python 程式與問卷大調查，找出潛在的密碼破口，並落實帳戶防禦加強。

自主學習歷程

- **階段 1-2（探索）：發現資訊不安全的問題，深入了解國內外密碼知識與駭客盜取帳號密碼方式。**
- **階段 3（實作）：從零自學 Python 程式，並寫出測試密碼檢測程式。**
- **階段 4-5（成果）：設計 Google 表單收集各年齡層數據，並對成果進行對比。**

甚麼是暴力破解

- 駭客利用電腦來破解密碼，將所有的組合逐一嘗試直到破解成功。
- 像是一個四位數的數字密碼，電腦可在0.0000001 秒內成功解開。
- 預防方式:將密碼長度拉長至12位數並加入大小寫和特殊符號。

甚麼是撞庫攻擊

- 駭客利用某個網站不幸外洩的帳號密碼，透過自動的程式去大量嘗試登入其他網站。
- 駭客看準大眾習慣在不同網站「重複使用同一組帳密」的漏洞。
- 預防方式: 每個網站都應設定不同密碼，並開啟雙重驗證。

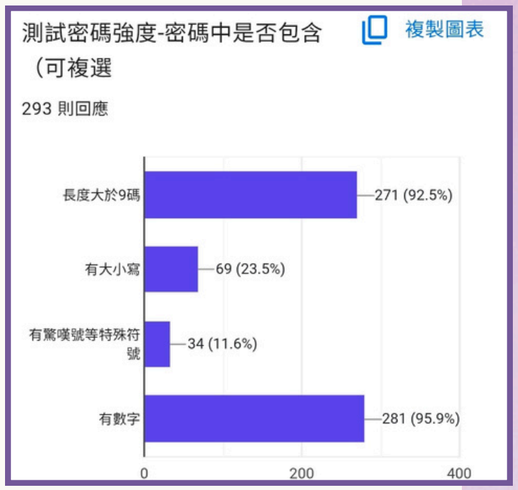
甚麼是passkey

- 一種主打「免密碼登入」的新方法，旨在讓傳統密碼在網路上徹底消失。
- 技術原理：採用非對稱加密技術，由使用者裝置保管「私鑰」，網站伺服器保管「公鑰」，登入時自動進行數學配對。
- 對抗暴力破解：因為網站根本不儲存任何密碼字串，駭客的 Python 暴力猜測程式完全沒有目標可以攻擊，解決撞庫風險

調查(google表單)
由約300個人進行回答

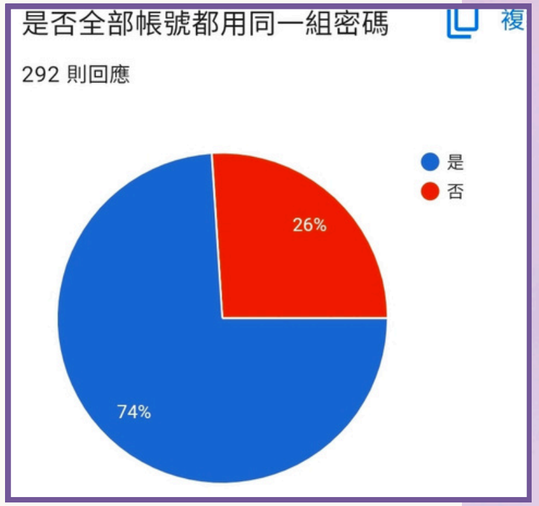
大多數人密碼未加入大小寫和特殊符號

「多數人的密碼習慣以純數字或英文小寫為主。缺乏大小寫混合與特殊符號（如:@、#），讓駭客的暴力破解程式在極短時間內猜出密碼，存在高度的安全漏洞。」



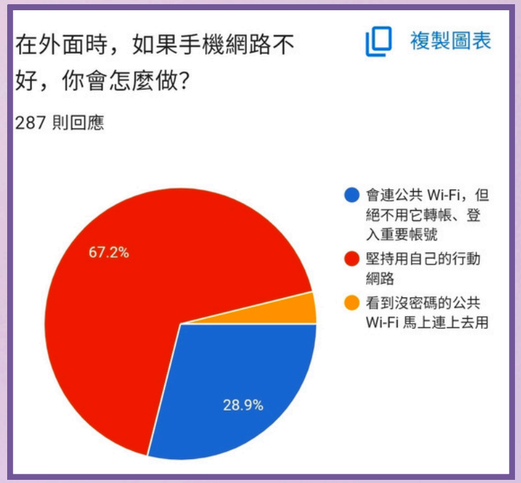
多數帳號密碼皆相同情形普遍

『一組密碼走天下』是校園中最普遍的資安盲點。任何一個網站的資料外洩，駭客就能利用這組帳密引發『撞庫攻擊』，輕鬆攻破個資與電子信箱。」

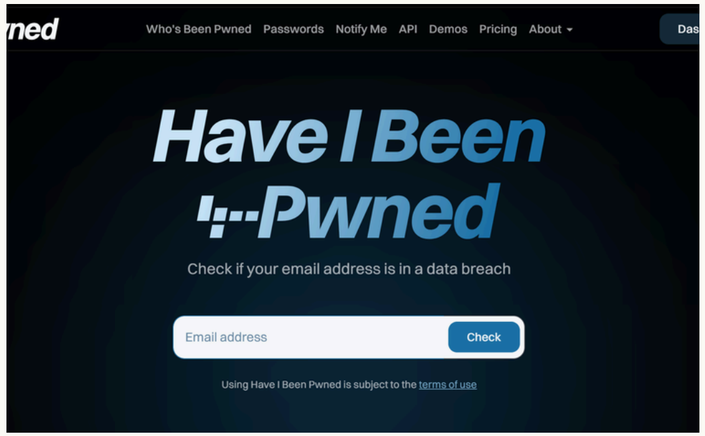


多數人不會隨意使用不安全的公共Wifi

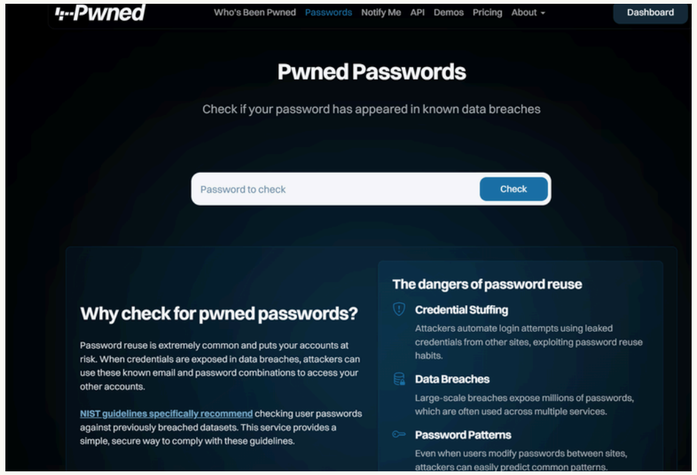
多數同學對公共網路具備防範意識。這能有效減少被駭客利用『中間人攻擊』、建立假熱點來盜用帳密的風險。」



國際資安外洩資料庫 HIBP 應用
-如何檢測並應對



安全



Have I Been Pwned - 全球最大的歷史外洩個資查核平台

這是全球的資安檢測平台，收集數十億筆被駭客外洩的帳號與密碼，使用者只要輸入電子信箱，就能查詢自己的個資是否曾暴露在風險中。

「一組密碼走天下？」

若查出該帳號遭到外洩時因停止使用該帳號並且將其他相似的帳密皆更改，避免駭客將這組帳密改登別得社群。

定期檢視與防範意識

除了被動查詢，更應建立主動防範習慣，若發現密碼已被列入外洩資料，代表該密碼已被駭客盜取，可進行快速破解，必須立即全面汰換。

實作:自製python密碼檢查器

我一開始不太了解打一些不屬於他的指令發現不對，前往查找之後一步步打出一個初稿，再加上循環的指令等等的問題一一解決。

最初版的python

- Def = define
- re=regular expression
- RElif = else if
- #給人看的，電腦並不會管
- len( )=length( )
- +=有疊加的意思

以上這些有相同的意思但不能改過去

不能寫成=+這樣會像是等於某一個正的數質

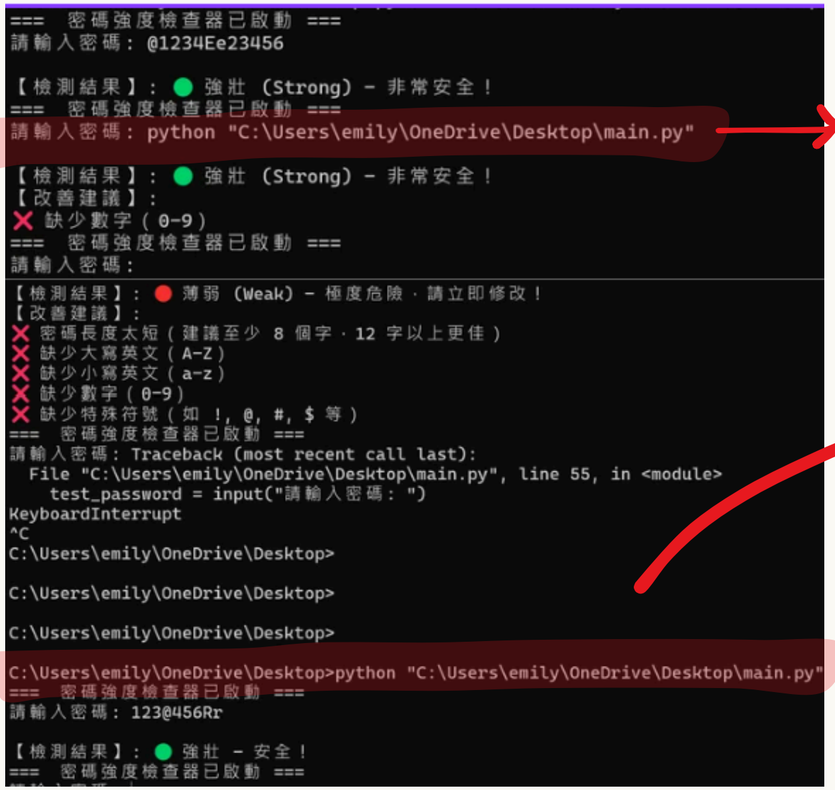
1.if如果
2.elif那不然
3.else否則(最後的希望)

append附加的意思
append後面得括號內的東西放入前面那個Feeback的〔 〕
re是一個內建的工具箱就像開頭的import re意思是搬住工具箱到python中
,password意思是去哪裡找，是一個找有沒有你前面那一些元素的地方
Return Level Feeback意思是把剛剛前面的東西傳回來。至於傳回來的內容就有Level跟Feedback。

最初版的main.py

```
import re

def check_password_strength(password):
    score = 0
    feedback = []

    # 1. 檢查長度是否達到 8 碼、12 碼
    if len(password) >= 12:
        score += 2
    elif len(password) >= 8:
        score += 1
    else:
        feedback.append("❌ 密碼長度太短（建議至少 8 個字，12 字以上更佳）")

    # 2. 檢查大寫字母
    if re.search(r"[A-Z]", password):
        score += 1
    else:
        feedback.append("❌ 缺少大寫英文（A-Z）")

    # 3. 檢查小寫字母
    if re.search(r"[a-z]", password):
        score += 1
    else:
        feedback.append("❌ 缺少小寫英文（a-z）")

    # 4. 檢查數字
    if re.search(r"[0-9]", password):
        score += 1
    else:
        feedback.append("❌ 缺少數字（0-9）")

    # 5. 檢查特殊符號
    if re.search(r"[!@#$%^&*(),.?\":{}|<>_]", password):
        score += 1
    else:
        feedback.append("❌ 缺少特殊符號（如 !, @, #, $ 等）")

    # 綜合評級
    if score >= 5:
        level = "🟢 強壯 (Strong) - 非常安全！"
    elif 3 <= score <= 4:
        level = "🟡 中等 (Medium) - 建議加強"
    else:
        level = "🔴 薄弱 (Weak) - 極度危險，請立即修改！"

    return level, feedback
```

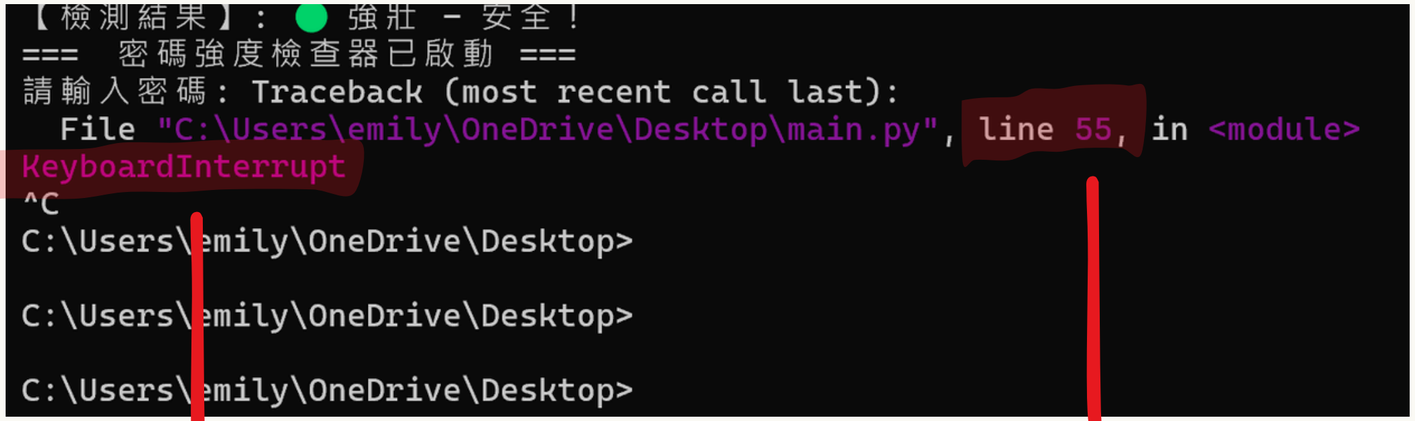
實作:錯誤



誤把指令當作密碼

成功讓程式心去抓檔案

程式在執行到第 55 行，有人在鍵盤上按下了 Ctrl + C 鍵。這個動作會發出中斷，強制結束正在執行中的程式



被打字輸入的東西打斷

在第五十五行

我後來對Main.py進行調整兩次。一開始不知道那個檔案要去重新再下指令，讓他再去讀一次。
新增加while True的循環後又改動了程式結果發生他把我的指令當成密碼進行檢測的狀況。
後來我向專業人士請教後才知道由於python會一直在循環裡所以要先按ctrl C來停止循環才能再次下達指令。

實作:加入黑名單與離開程式

```
# 黑名單，不能用的密碼
PASSWORD_BLACKLIST = ["123456", "1234567", "12345678", "password", "00000000"]

# 先檢查這組密碼有沒有在黑名單裡面
if password in PASSWORD_BLACKLIST:
    print("\n【檢測結果】：🔴 薄弱 - 危險")
    print("改善建議：此密碼過於簡易，請更換密碼！")

# 如果不在黑名單，才開始走你原本寫的那些密碼表規則檢查
else:
    print("\n【密碼表強度檢查中...】")
```

經過討論，我發現製作黑名單的程式甚至有兩三種以上的做法，至於我覺得最好了解的。有兩種，一種是把黑名單列在最上面，讓他先經過黑名單的程式後再去跑密碼檢測的程式。另一個是每一次檢測（大小寫，數字長度）都先把黑名單寫入最前面的程式if，其他的向後順延。我也是在這裡學到原來除了if elfi跟Else 以後，else可以用很多次。由於第二種方法幾乎調整個程式，於是我選擇第一種方法。

將密碼強度檢查器啟動移到While True迴圈外面，讓它不會每一次都出現，再加上離開程式的指令，且Break必須在 while True裡面，並且要換一行才可使用。

```
# --- 執行測試 ---
print("===  密碼強度檢查器已啟動 ===")

while True:  =迴圈
    test_password = input("請輸入密碼，輸入q離開: ")
    #增加離開程式
    if test_password.lower() == "q":
        print("程式結束")
        break

    level, feedback = check_password_strength(test_password)

    print(f"\n【檢測結果】: {level}")
    if feedback:
        print("【改善建議】:"+"、".join(feedback))
```

將密碼強度檢查器啟動移到While True迴圈外面，讓它不會每一次都出現。

再加上離開程式的指令

讓排版更簡潔，不會分成很多行占空間。

```
【檢測結果】：👎 黑名單
改善建議：此密碼在黑名單過於簡易，請更換密碼！

【檢測結果】：👎 黑名單
===  密碼強度檢查器已啟動  ===
請輸入密碼：123456

【檢測結果】：👎 黑名單
改善建議：此密碼在黑名單過於簡易，請更換密碼！

【檢測結果】：👎 黑名單
===  密碼強度檢查器已啟動  ===
請輸入密碼：1234567

【檢測結果】：👎 黑名單
改善建議：此密碼在黑名單過於簡易，請更換密碼！

【檢測結果】：👎 黑名單
===  密碼強度檢查器已啟動  ===
請輸入密碼：1234567

【檢測結果】：👎 黑名單
改善建議：此密碼在黑名單過於簡易，請更換密碼！

【檢測結果】：👎 黑名單
===  密碼強度檢查器已啟動  ===
請輸入密碼：123456

【檢測結果】：👎 黑名單
改善建議：此密碼在黑名單過於簡易，請更換密碼！

【檢測結果】：👎 黑名單
===  密碼強度檢查器已啟動  ===
請輸入密碼：
```

1

2

不同程式造成重複列印的情況

實作:黑名單重複出現

原因:print() 與 return 意思重疊

- 第 1 次列印(print)：

當密碼符合黑名單時，程式會先執行 print("\n【檢測結果】：👎 黑名單")，直接在終端機畫面上印出這一行字。

- 第 2 次列印（return)：

接著程式執行 return "👎 黑名單", [ ]，將 "👎 黑名單" 這個字串當作結果，如果呼叫這個程式又把這個回傳的結果拿去 print( ) 印出來，畫面上就會出現第二次的黑名單字樣。

```
# 先檢查這組密碼有沒有在黑名單裡面
if password in PASSWORD_BLACKLIST:
    print("\n【檢測結果】：👎 黑名單")
    print("改善建議：此密碼在黑名單過於簡易，請更換密碼！")
    return"👎 黑名單", []
```

1

2

```
import re


def check_password_password):
    score = 0
    feedback = []

    # 黑名單，不能用的密碼
    PASSWORD_BLACKLIST = ["123456", "1234567", "12345678", "password", "00000000"]

    # 先檢查這組密碼有沒有在黑名單裡面
    if password in PASSWORD_BLACKLIST:
        return "👎黑名單\n 改善建議：此密碼在黑名單過於簡易，請更換密碼", []



    # 如果不在黑名單，才開始走你原本寫的那些密碼表規則檢查
    else:
        print("\n【密碼表強度檢查中...】")


        #檢查長度是否達到 8 碼、12 碼
        if len(password) >= 12:
            score += 2
        elif len(password) >= 8:
            score += 1
        else:
            feedback.append("❌ 密碼長度太短（建議至少 8 個字，12 字以上更佳）")

        #檢查大寫字母
        if re.search(r"[A-Z]", password):
            score += 1
        else:
            feedback.append("❌ 缺少大寫英文")

        #檢查小寫字母
        if re.search(r"[a-z]", password):
            score += 1
        else:
            feedback.append("❌ 缺少小寫英文")

        #檢查有沒有數字
        if re.search(r"[0-9]", password):
            score += 1
        else:
            feedback.append("❌ 缺少數字")

        #檢查有無特殊符號
        if re.search(r"[!@#$%^&*(),.?\":{}|<>_]", password):
            score += 1
        else:
            feedback.append("❌ 缺少特殊符號")

        # 最後結果
        if score >= 5:
            level = "🟢 強壯 - 安全！"
        elif 3 <= score <= 4:
            level = "🟡 中等 - 加強"
        else:
            level = "🔴 薄弱 - 危險，請立即修改！"

        return level, feedback



# --- 執行測試 ---
print("===  密碼強度檢查器已啟動 ===")

while True:
    test_password = input("請輸入密碼，輸入q離開: ")
    #增加離開程式
    if test_password.lower() == "q":
        print("程式結束")
        break

    level, feedback = check_password_strength(test_password)

    print(f"\n【檢測結果】: {level}")
    if feedback:
        print("【改善建議】:"+"、".join(feedback))
```

點我開啟密碼檢查器

成效對比

- 分成兩組密碼

- 556677[約需不到 1 秒鐘內破解]

當將密碼改成成@55Ee66Ff77Gg@時[約需 65 萬年破解]

- 12345678[約需萬分之一秒內被瞬間破解]

當將密碼改成#A123456a時[約需不到 1 秒鐘被破解

暴力破解

暴力破解本質上是數學的排列組合問題

計算公式:

$$T = \frac{C^L}{S}$$

- T:所需的時間
- C：可能性總數
  - 數字
  - 純英文小寫
  - 大小寫混合＋數字
  - 加上特殊符號
- L：密碼的長度（位數）
- S：駭客系統每秒能嘗試的密碼次數

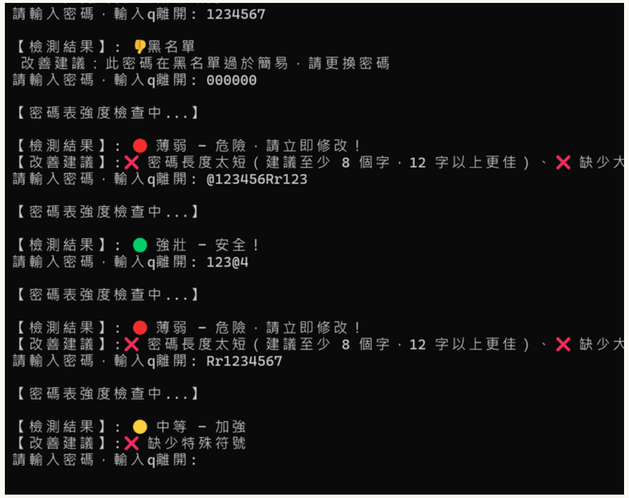
before

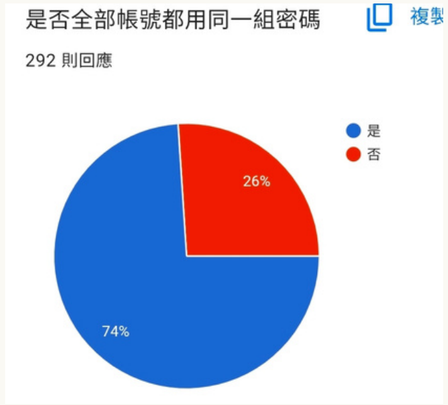
```
===  密碼強度檢查器已啟動  ===
請輸入密碼，輸入q離開： 556677

【密碼表強度檢查中...】

【檢測結果】：🔴 薄弱 - 危險，請立即修改！
【改善建議】：❌ 密碼長度太短（建議至少 8 個字，
請輸入密碼，輸入q離開： 12345678

【檢測結果】：👎黑名單
 改善建議：此密碼在黑名單過於簡易，請更換密碼
請輸入密碼，輸入q離開：
```

after

```
碼表強度檢查中...】

檢測結果】：🟢 強壯 - 安全！
輸入密碼，輸入q離開： @55Ee66Ff77Gg@

密碼表強度檢查中...】

檢測結果】：🟢 強壯 - 安全！
輸入密碼，輸入q離開： #A12345678a
```

心得反思





在這次自主學習中，我經歷從數據調查到程式實作的一連串探究。數據讓我了解到真正的問題才能對症下藥，並讓我能深入理解程式編碼和網路相關原理，這次的學習過程使我學會耐心檢查一行行的編碼和基本指令，我也在步步的編寫中體會多次錯誤再更正，最後終於成功時十分有成就感

Thank You!