W poszukiwaniu bezpiecznego hasła!

Moduł 2

Obowiązkowo

- ✓ Wiem jak korzystać z zewnętrznego API.
- ✓ Umiem pracować ze stringami i wycinać ich fragmenty.
- ✓ Potrafię pracować z plikami tekstowymi - odczytywać je i zapisywać.
- ✓ Wiem czym róznią się stringi od byte'ów.
- ✓ Rozumiem na czym polega hashowanie.

Opcjonalnie

- ✓ Potrafię korzystać z biblioteki logging by efektywnie zarządzać logami aplikacji.

- ✓ Umiem tworzyć instalatory dla bibliotek aby móc wykorzystywać stworzony przez siebie kod w innych projektach.

Projekt 2

Czy moje hasło jest bezpieczne?

Opis zadania

Wszyscy używamy haseł... ale czy te, których używamy są odpowiednio skomplikowane? Jeżeli nawet są to czy kiedykolwiek nasze hasło wyciekło?

Bezpieczeństwo hasła

Jakie hasło jest dobre? Przyjmijmy kilka założeń:

- Musi mieć odpowiednią długość. Powiedzmy, że musi posiadać minimalnie 8 znaków.
- Musi zawierać przynajmniej jedną cyfrę.
- Musi zawierać przynajmniej jeden znak specjalny.
- Musi zawierać wielkie i małe litery.

Czy to wystarczy by spać spokojnie?

Absolutnie nie. Hasło takie jak `ZAQ!2wsxCDE#` spełnia wszystkie te założenia, jednakże gwarantuje Wam, że znajduje się w wielu słownikach. **Co to znaczy?**

Jak “złamać” hasło?

Intruz może złamać hasło na dwa sposoby. Dysponuje hashem hasła (np. z wyciekniętej bazy danych) albo próbujemy zalogować się do usługi “strzelając” w nią różnymi hasłami.

Brute force

Metoda ta jest niezwykle niewydajna, polega na tym, że testujemy kolejne hasła według jakiegoś klucza. np. 123, 124, 125, aaa, aab, aac, aad, aae, itd. Po prostu sprawdzamy wszystkie możliwe kombinacje haseł.

Metoda słownikowa

Posiadamy słownik z hasłami i testujemy konkretne hasła.

`admin123, adam88, kacper88, wojtek87, 1988adam, 1988kacper` itd.

Jak widzicie hasła mogą być też personalizowane. Jednakże co gdy atakujący nic o nas nie wie? Używa popularnych haseł.

To oznacza, że nasze hasło musi być możliwie niepopularne!

Co sprawia, że hasło jest popularne?

To proste! Występowało w wielu wyciekach. Jak się o tym dowiedzieć? Jest genialna usługa **haveibeenpwnd.com**, która posiada także swoje API! Nie tylko możemy sprawdzić czy hasło do naszego maila wyciekło, ale także czy nasze hasło jest bezpieczne.

Jak działa have I been pwnd API?

Załóżmy, że posiadasz hasło `kacper1988` musisz zahaszować hasło korzystając z algorytmu sha1 i uzyskasz C67A7945A04E56EBAB8ED275705AD144C183EB1A to hash tego hasła.
Wysłanie tego hasha do haveibeenpwnd może spowodować, że Twoje hasło wycieknie dlatego wysyłamy im tylko **5 pierwszych znaków hasha,** a serwis odpowie wszystkimi *końcówkami* hashy jakie zna haveibeen pwnd wraz z informacją ile razy dane hasło wyciekło.
FE79AA691EE2A185FF37CF49FD96B24E5A6:3
FCD28A776255B32F1EDD73E9ED2B97D9CF3:14

Czy hasło wyciekło?

Jeśli API odpowiedziało **Twoją końcówką** to tak.. Twoje hasło wyciekło. Polecamy najszybszą jego zmianę!

Treść zadania

Dobra.. co właściwie jest do zrobienia? Zapisz wszystkie Twoje hasła w pliku tekstowym **passwords.txt.**

Twój program otwiera plik **passwords.txt**, a następnie przetwarza go linia po linii.
Jeśli hasło spełnia wszystkie wymogi dobrego hasła oraz jeżeli nie wyciekło ani razu
to powinno zostać zapisane do osobnego pliku o nazwie **bezpieczne.txt.**
Opis API: https://haveibeenpwned.com/API/v3

Dodatkowo

1. Każde zapytanie możesz zapisywać za pomocą loggera do pliku.
Zwróć uwagę, że zapisywanie wszystkich hashy do pliku
nie będzie najbezpieczniejsze!

2. Biblioteka wraz z walidatorem będzie na tyle przydatna,
że warto stworzyć jej paczkę instalacyjną by była dostępna
dla innych Twoich projektów.

Jak dalej rozwinąć ten projekt?

Dalsze pomysły

- ✓ Pomyśl o innych sposobach sprawdzania bezpiecznego hasła.
  Dla przykładu, czy nie jest zbyt podobne do nazwy użytkownika lub innego stringa?

- ✓ W sieci jest dużo słowników haseł, możesz pobrać przykładowy słownik i użyć go też do sprawdzenia bezpieczeństwa Twojego hasła.

- ✓ Możesz dołożyć np. sprawdzanie mocy hasła i zapisywać je w pliku wynikowym aby ocenić, które hasło jest lepsze, a które jest gorsze?