**Основи інформаційної безпеки**

Практичне заняття №5

Тема: **«Безпечне зберігання паролів»**

Звіт студента групи МІТ-21 Мірошника Андрія

Варіант №17

**GitHub: https://github.com/AndriiMiroshnyk/Cybersecurity2021.git**

***Основні способи зберігання паролів:***

1. Зберігання паролів у відкритому вигляді
2. Шифрування паролів
3. Хешування паролів
4. Використання хешу та солі для зберігання паролів
5. Використання спеціальних функцій

***Зберігання паролів у відкритому вигляді***

Найбільш небезпечний спосіб зберігання паролів. Паролі зберігаються як звичайний відкритий текст. Якщо база даних буде скомпрометована і таблиці паролів будуть вкрадені, зловмиснику не потрібно буде робити жодної важкої роботи, щоб отримати доступ до акаунтів користувачів. Зловмисник входить в систему з іменем користувача та паролями, які були вкрадені з бази даних. Це може мати катастрофічні наслідки. Отже, зберігання паролів у відкритому тексті — жахлива ідея.

***Шифрування паролів***

Шифрування є двостороннім процесом; ви шифруєте пароль за допомогою секретного ключа, а потім можете розшифрувати за допомогою того самого ключа. Цей механізм можна використовувати для зберігання паролів, і це дійсно набагато краще, ніж зберігання паролів у відкритому вигляді, але цей підхід пов’язаний зі своїм набором проблем:

• Ви повинні керувати ключами;

• Ключі шифрування потрібно десь зберігати;

• Ключі можуть бути зламані;

• Як ефективно поділитися цим ключем з іншими людьми?

• Чи кожній клієнтській програмі потрібен доступ до ключа шифрування паролю?

Якщо зловмисник отримує копію ключа шифрування та таблиці ваших паролів, він має доступ до всієї інформації. Після дій зловмисника потрібно змінити ключ шифрування та запропонувати всім користувачам змінити свої паролі, щоб вони були повторно зашифровані.

Це все ще не найкраще рішення через складність керування ключами та можливість розшифрувати паролі.

***Хешування паролів***

Хешування є односторонньою функцією, що означає, що після хешування пароля ви не зможете повернути хеш назад до початкового пароля. Криптографічне хешування має чотири основні властивості:

• Легко обчислити хеш-значення для будь-якого заданого повідомлення.

• Неможливо створити повідомлення, яке має заданий хеш.

• Неможливо змінити повідомлення без зміни хеша.

• Дуже важко знайти два різних повідомлення з однаковим хешем.

Ці властивості роблять хешування набагато кращим рішенням для кодування пароля перед зберіганням, оскільки важко перейти від хешу назад до пароля.

Але є недоліки. Хешування пароля чутливе до двох різних атак. Це атака грубої сили та атака райдужною таблицею. Атака грубої сили – зловмисник пробує різні комбінації паролів, поки не отримає той, який відповідає хешу пароля; простіше, якщо зловмисник уже зламав хешовані таблиці паролів. Зловмисники можуть мати вже підготовлений розширений словник паролів і скомпрометованих паролів через цей словник, поки не знайдуть збіг із хешем у таблиці паролів.

***Використання хешу та солі для зберігання паролів***

Кращий підхід до прямого хешування — збільшити ентропію пароля, який атакується, ускладнюючи відновлення пароля. Ентропія - це міра міцності пароля і того, наскільки легко його вгадати. Якщо розширити пароль додатковими випадковими літерами і цифрами, то вгадати цей пароль набагато важче; це реалізовано додаванням значення солі до пароля перед хешуванням.

Сіль можна згенерувати як випадкове число, а потім додати до пароля перед хешуванням. Додаючи значення солі, ви значно ускладнюєте пароль, і це значно ускладнює застосування атаки грубої сили або райдужної таблиці. Можна включити значення солі разом із хешованим паролем до бази даних; сіль не вважається ключем шифрування. Основна перевага солі, якщо ви використовуєте різну соль для кожного пароля, полягає в тому, що отриманий хеш відрізняється для кожного користувача, якщо він має однаковий пароль.

Проблема зі збереженням паролів та їх хешуванням (навіть із значенням солі) полягає в тому, що, оскільки мікропроцесори та графічні процесори стають швидшими, вони можуть виконувати атаки грубої сили та атаки райдужної таблиці набагато швидше. Потрібне нове рішення, яке зазначене далі.

***Використання спеціальних функцій***

Функція приймає значення пароля, солі та «кількості ітерацій».

Значення кількості ітерацій повторює операцію хешування над паролем кілька разів, щоб отримати похідний ключ для пароля, який можна зберегти в базі даних.

Повторюючи процес хешування над паролем кілька разів, алгоритмічно сповільнюється процес хешування, що значно ускладнює атаки на пароль грубою силою або райдужною таблицею. Це означає, що перевіряється менше паролів.

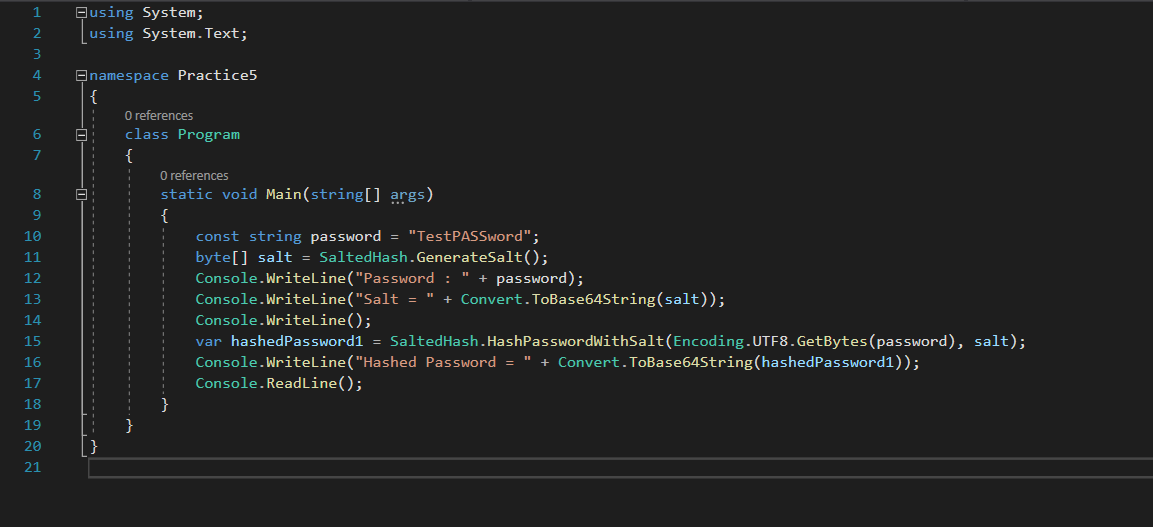
Недоліки: застосування значень кількості ітерацій спричиняє додаткову затримку, тому потрібно знайти прийнятне значення затримки, щоб не сповільнювати систему.

*Завдання 3:*

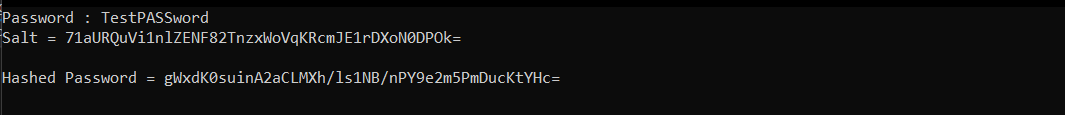
Клас SaltedHash:



Програма:

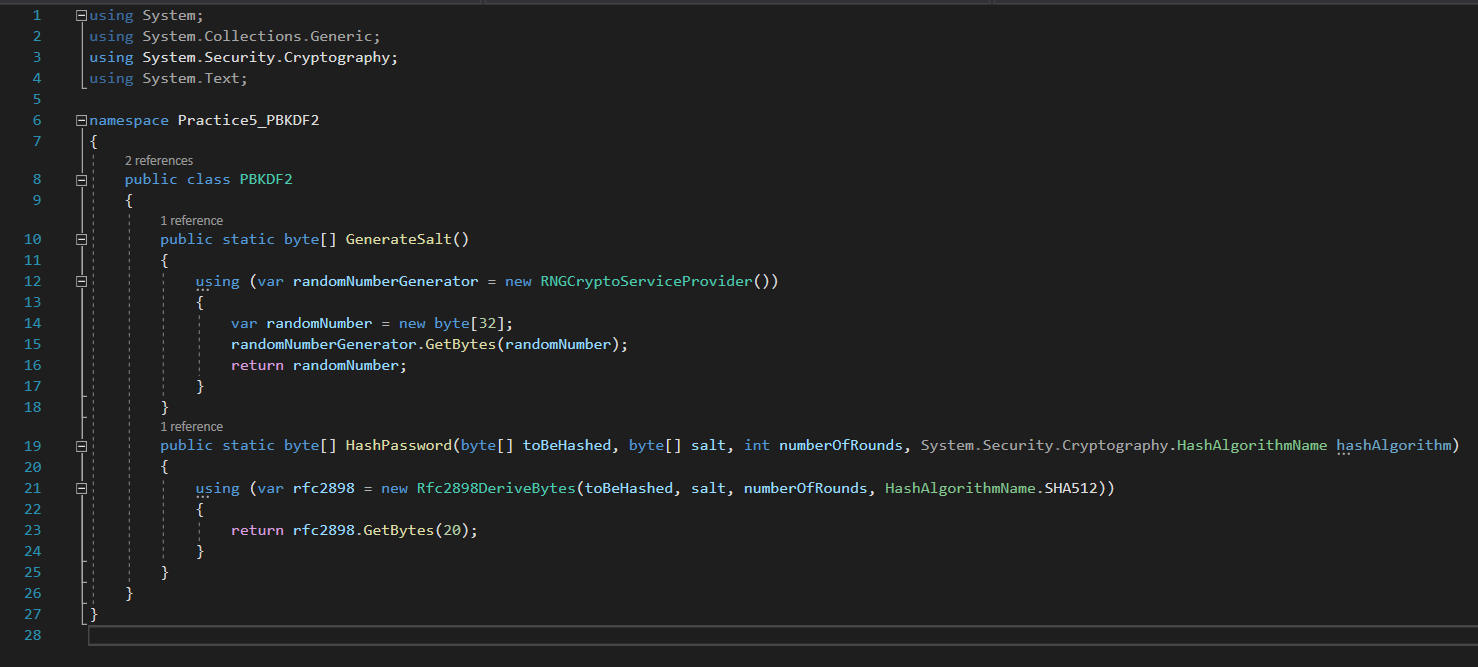


Результати роботи:

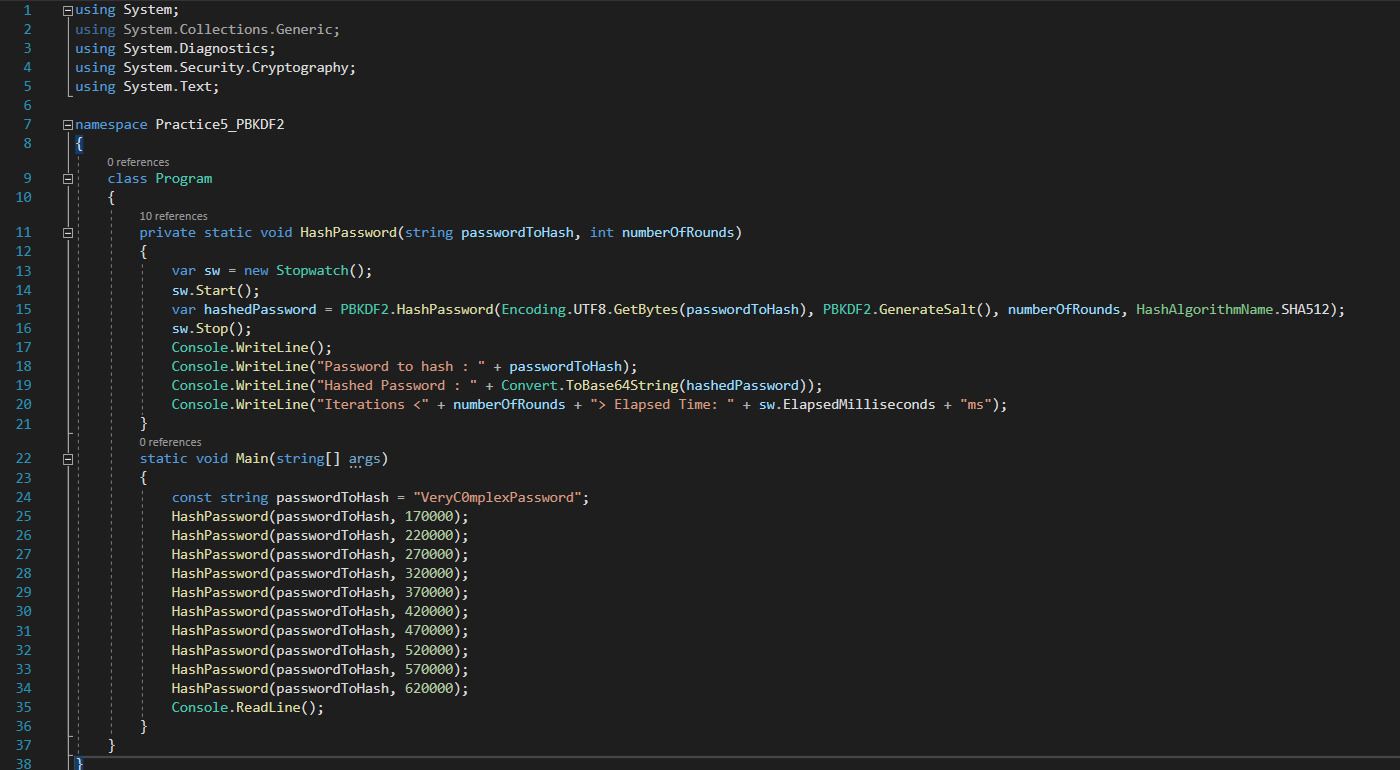


*Завдання 4:*

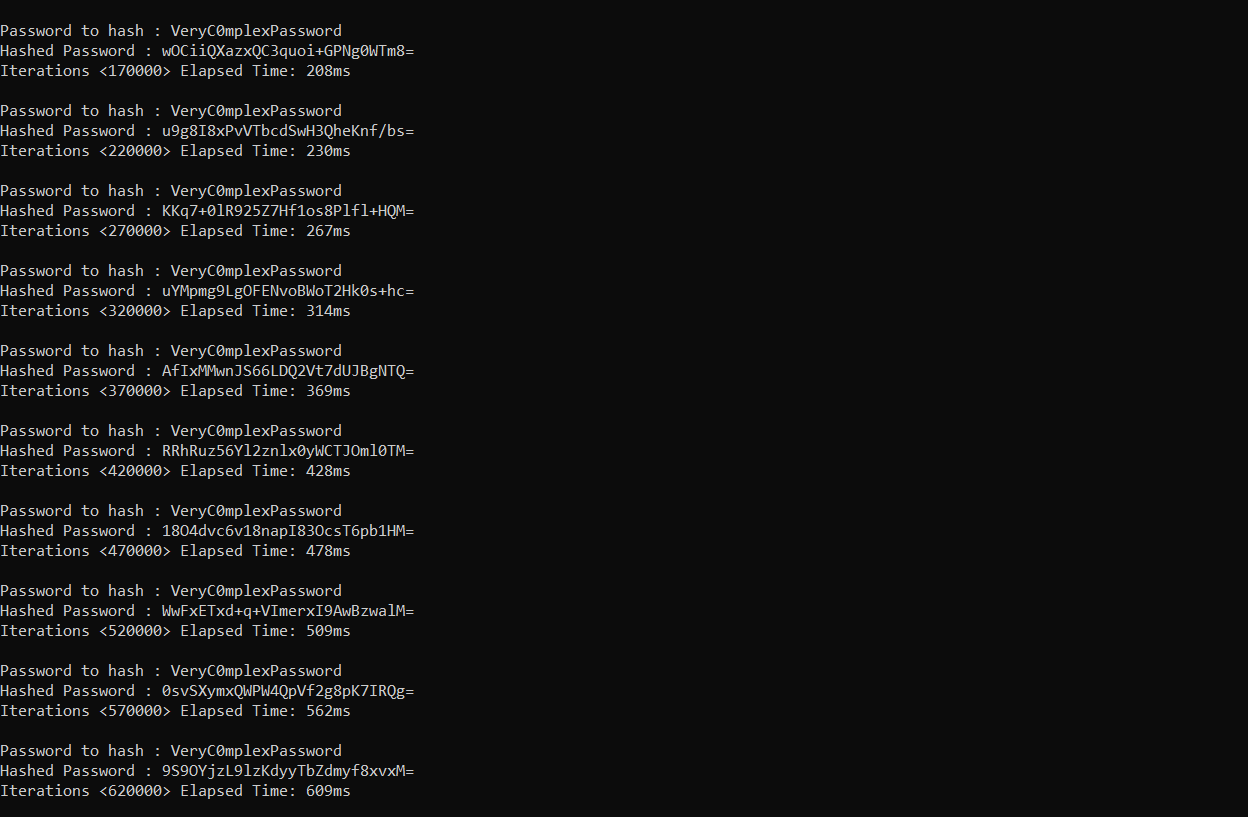
Клас PBKDF2:



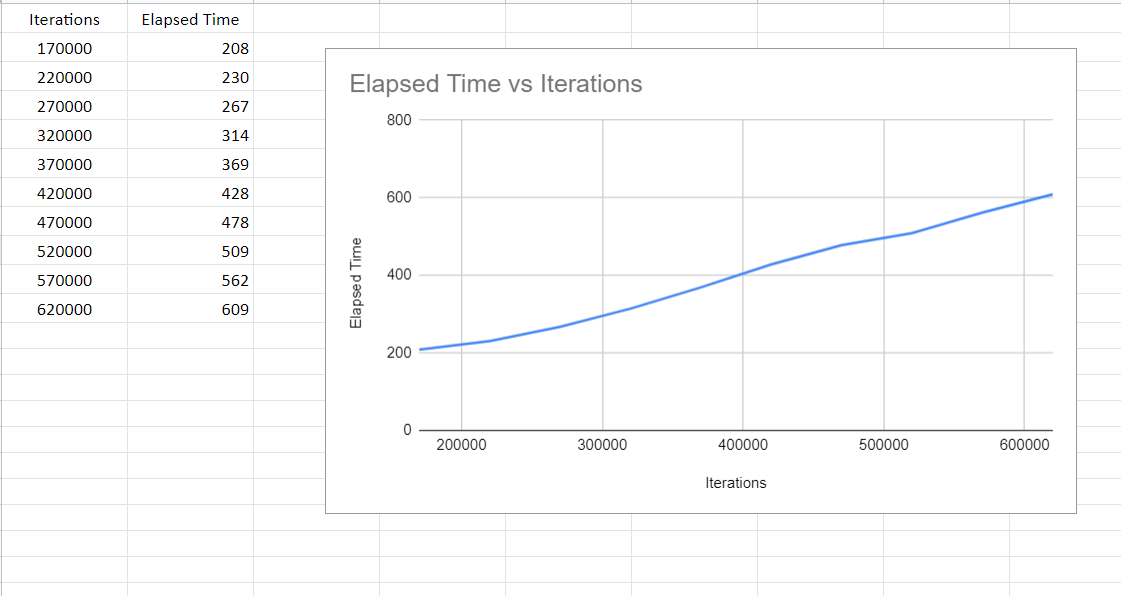
Програма:



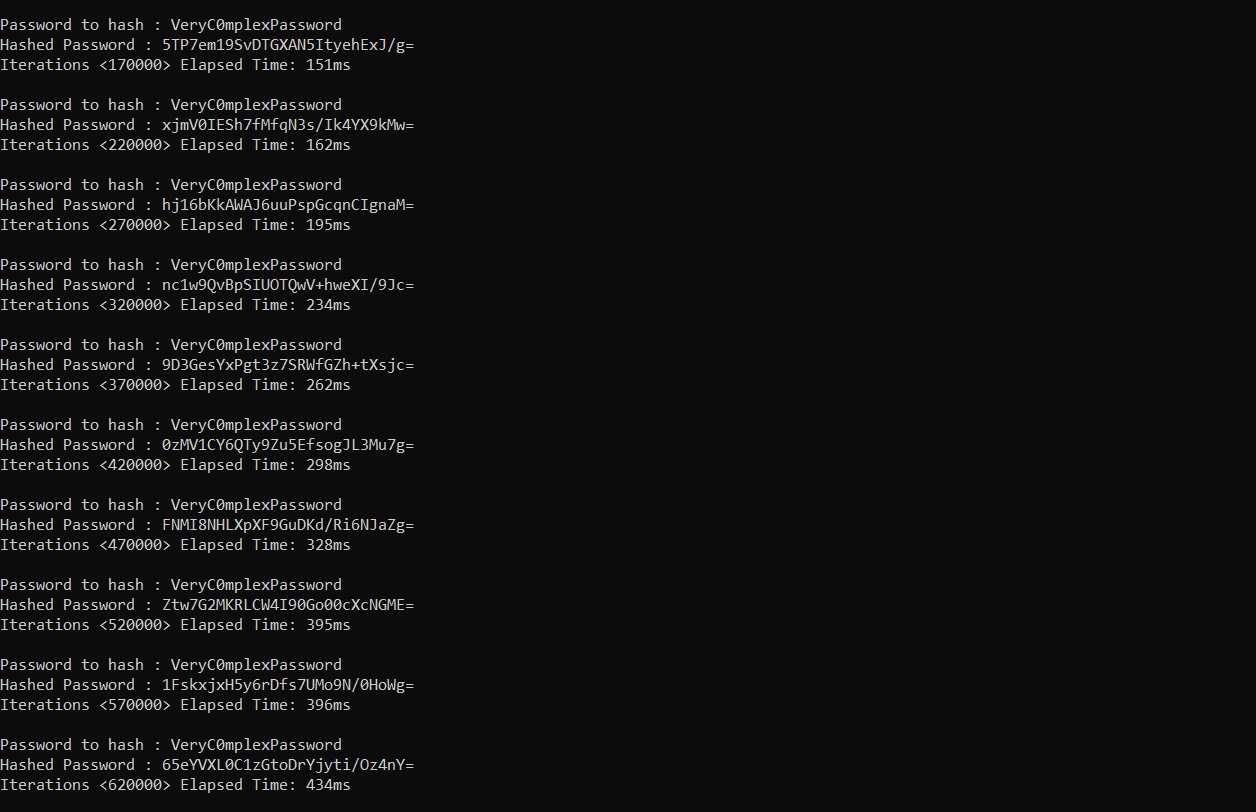
Результати роботи (SHA1):



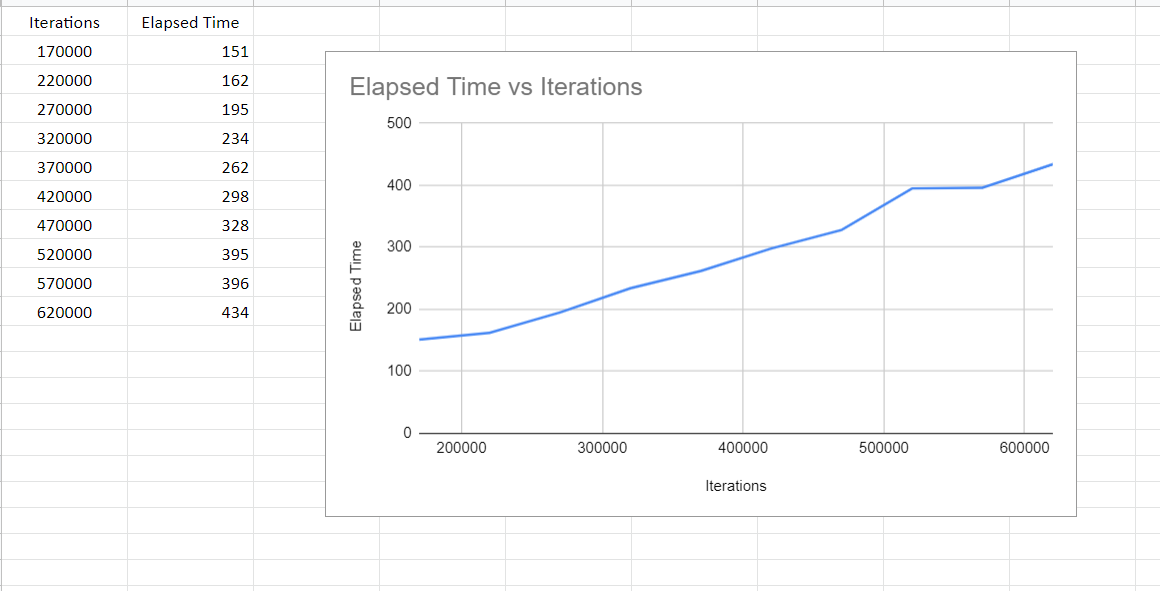
Графік:



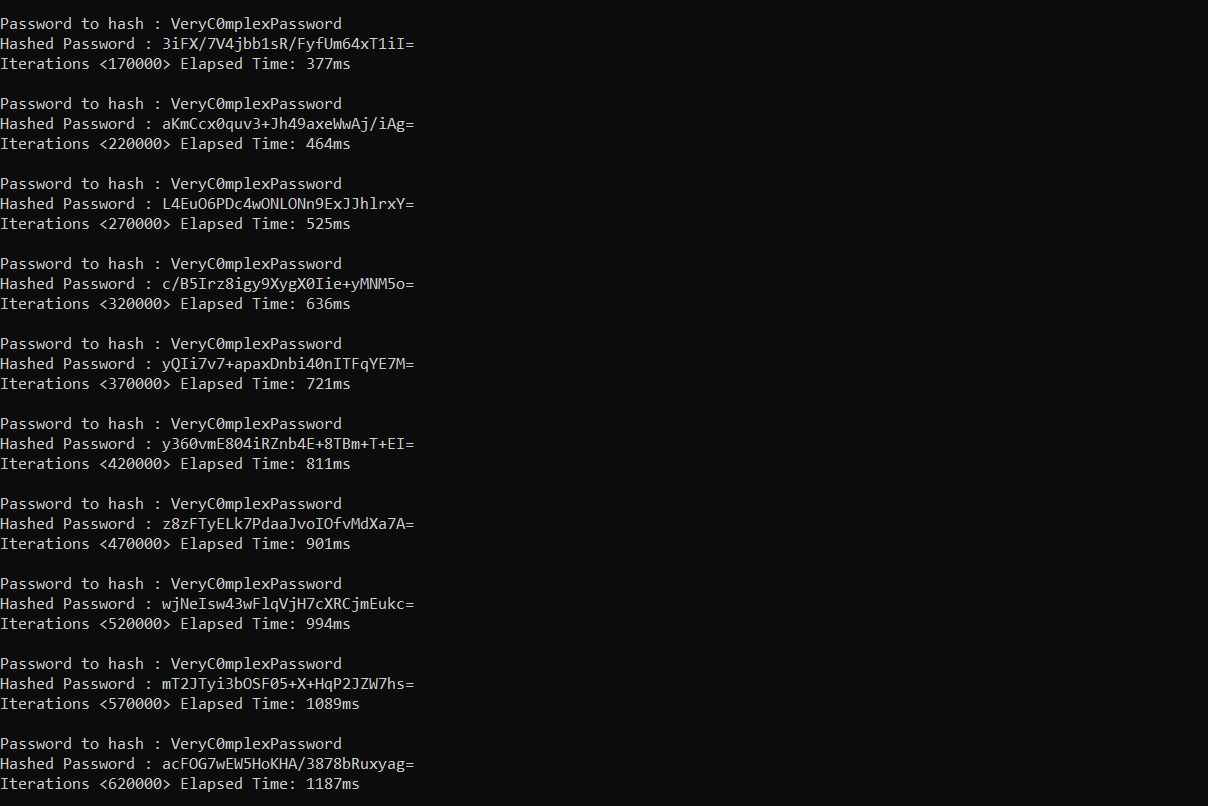
Результати роботи (SHA256):



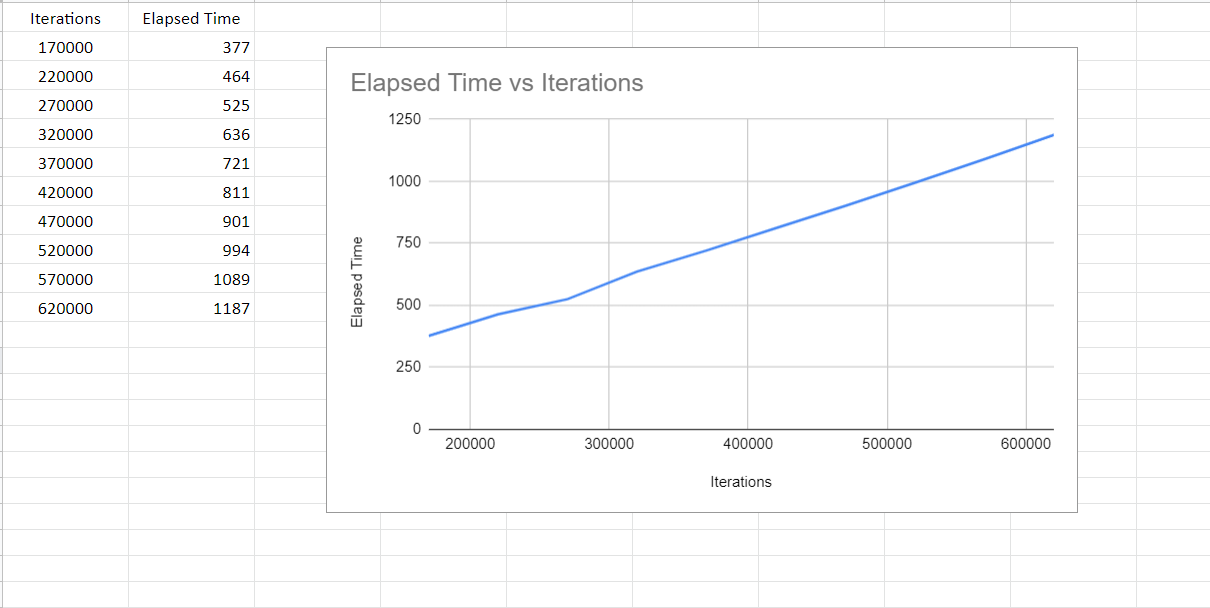
Графік:

****

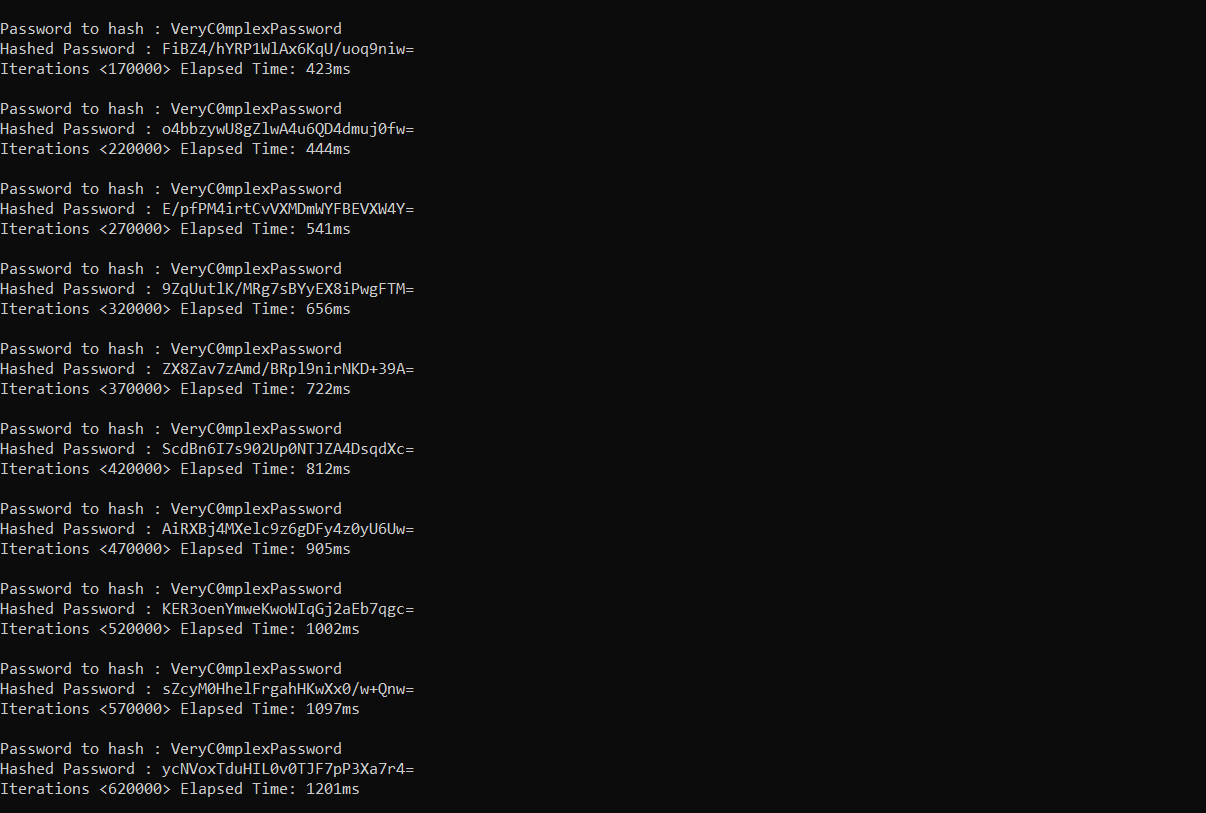
Результати роботи (SHA384):



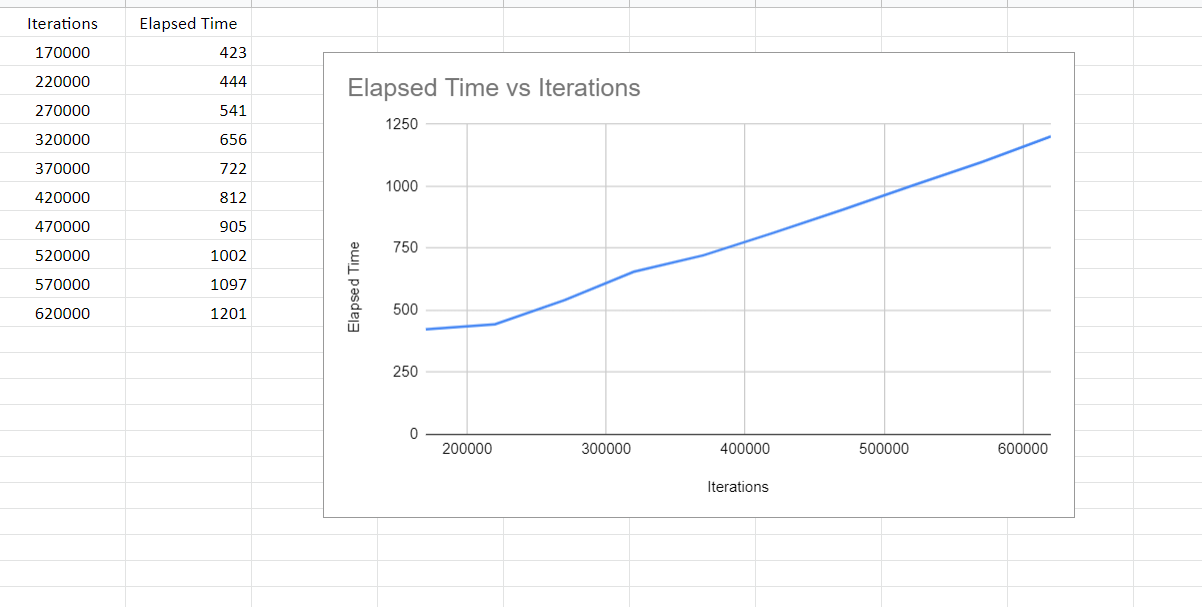
Графік:



Результати роботи (SHA512):

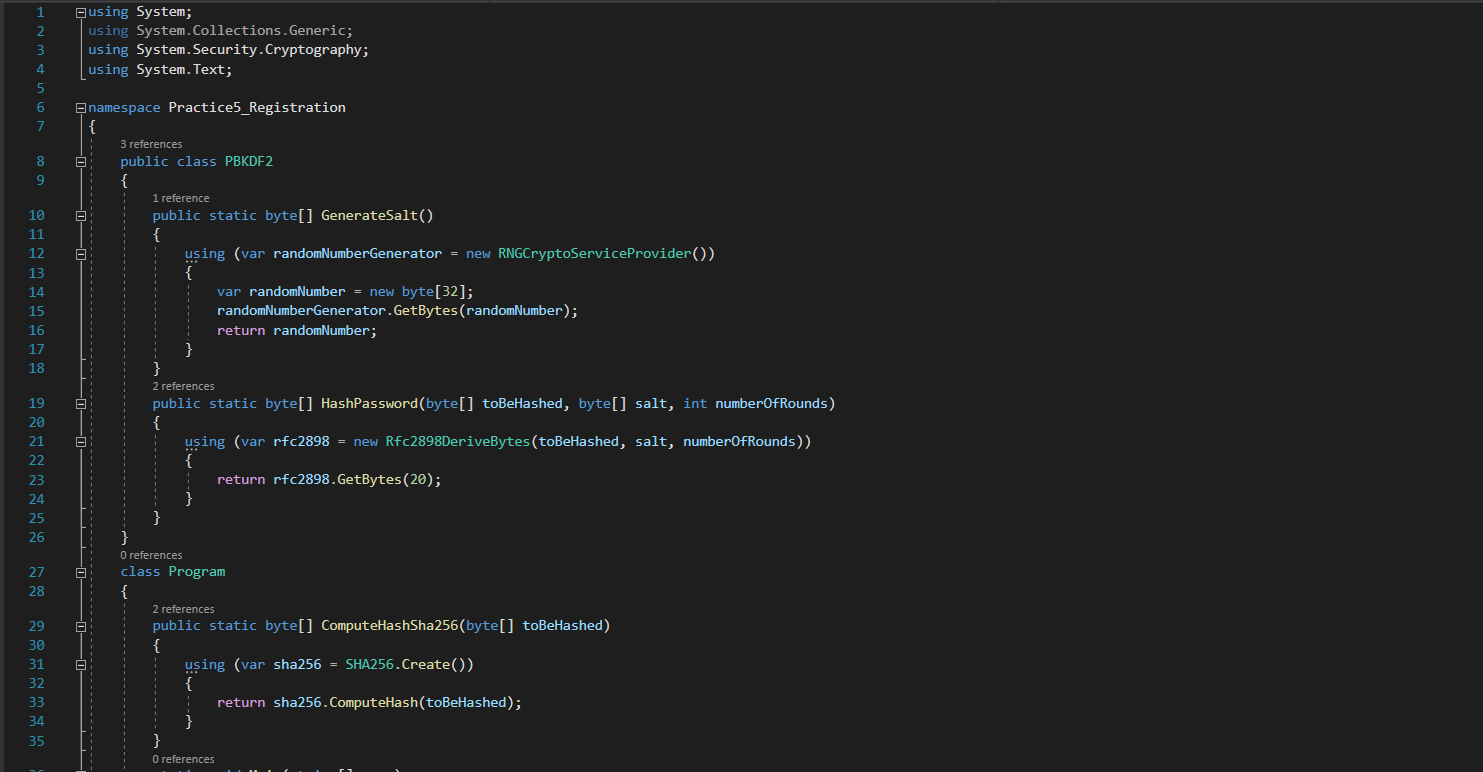


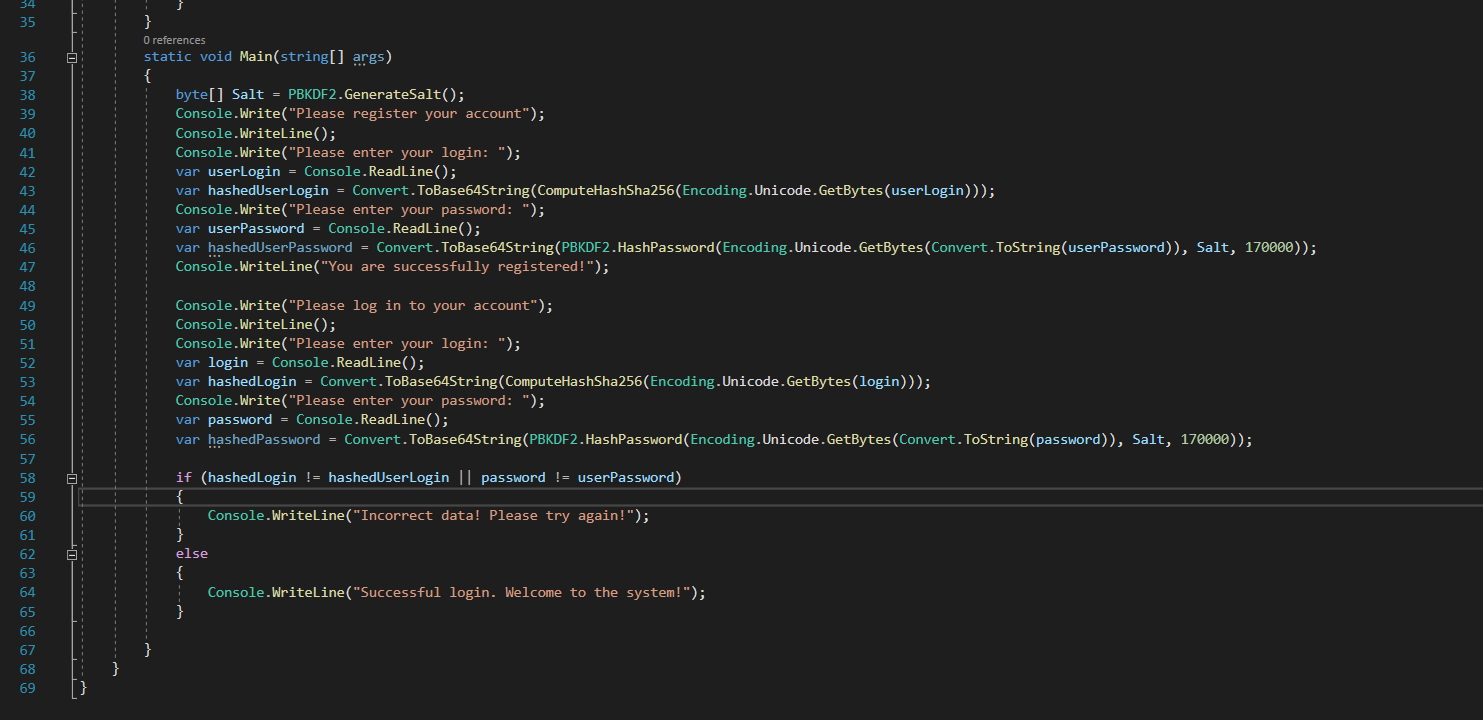
Графік:



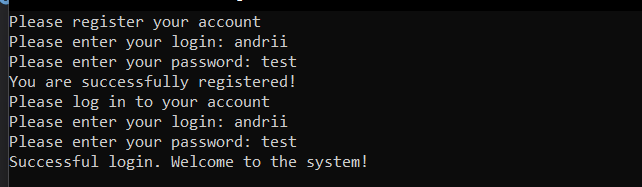
*Завдання 5:*

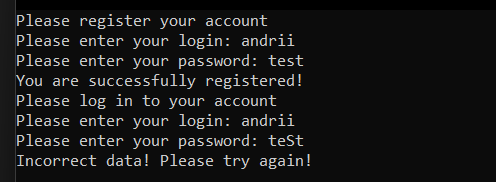
Програма:

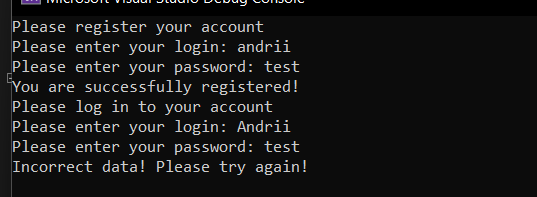




Результати роботи:







**Висновок:**

Надійне зберігання паролів є важливим завданням для будь-якої системи. На жаль, все ще існують системи, які використовують паролі відкритого тексту. На щастя, кількість таких систем зменшується, але такі системи все ще існують. Краще рішення, хоча і не ідеальне, полягає в тому, щоб використовувати хешування для захисту паролів, бажано більш надійний хеш, як-от SHA-256. Хоча це набагато краще рішення, ніж відкритий текст, таке хешування все ще чутливе до грубої сили або атак за допомогою райдужних таблиць.

Ще кращим рішенням є хешування за допомого солі, при якому ви збільшуєте ентропію оригінального пароля, генеруючи випадкове число, яке ви додаєте до пароля перед хешуванням. Вам потрібно зберегти це значення солі, але процес “засолювання” значно ускладнює виконання атак грубої сили та атак райдужних таблиць; це гарне рішення, яке сьогодні використовується багатьма компаніями, але є більш краще рішення. Кращим рішенням є використання спеціальних функцій (наприклад PBKDF2). Процес PBKDF2 подібний до техніки хешування за допомогою солі, але у функцію додається аргумент - кількість ітерацій. Метою цього способу є алгоритмічне сповільнення хешування, щоб зробити атаки грубою силою складнішими. Замість того, щоб сучасні графічні процесори намагалися отримати мільярди хешів в секунду, функція зменшує кількість спроб, наприклад до п'яти або десяти хешів в секунду. Це робить процес зберігання паролів більш надійним і безпечним.

Треба пам’ятати про число, яке ви використовуєте для підрахунку ітерацій; затримка, викликана повторюваним хешем, спричинить помітну затримку, коли користувач спробує увійти, наприклад, на сайт. Затримка неминуча, але потрібно знайти прийнятне значення затримки, щоб не сповільнювати систему.