Звіт

До практичного заняття №5

З теми: «Безпечне зберігання паролів»

Підготував:

Студент групи МІТ-21

Чорноконь Денис

1. Опрацювати розділ 5 літературного джерела Haunts Stephen. Applied Cryptography in .NET and Azure Key Vault.

2. Для кожного способу зберігання паролів, описаного у розділі 5 (Safely Storing Passwords), законспектувати основні переваги і недоліки. Зробити висновок про доцільність застосування даного способу зберігання паролів.

3. Розробити клас SaltedHash, що реалізує хешування паролів із додаванням додаткової ентропії. Продемонструвати роботу класу, обчислюючи хеш для заданого пароля та "солі".

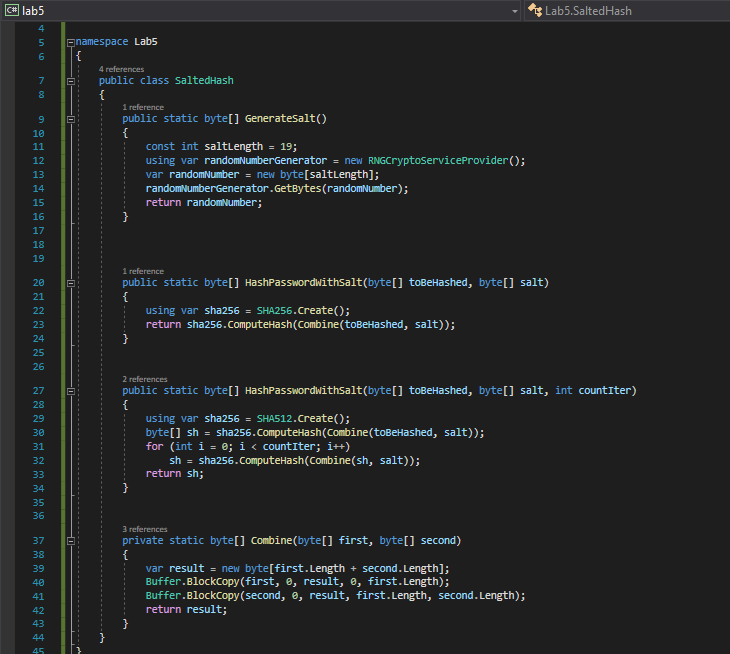
4. Розробити клас PBKDF2, що має наступну функціональність: генерує "сіль", задає алгоритм хешування (MD5, SHA1, SHA256, SHA384, SHA512) та обчислює хеш для заданого числа ітерацій. Створити програму, що обчислює час, витрачений на обчислення хешу для різного числа ітерацій (10 значень із кроком 50'000; перше значення = номер варіанта \* 10'000 ). Побудувати графік залежності витраченого часу від числа ітерацій.

5. Написати програму, що реалізує хешування введеного пароля під час реєстрації користувача та зберігає логін, пароль та "сіль" у пам'яті. Реалізувати можливість автентифікації за логіном і паролем. Число ітерацій = номер варіанта(8) \* 10'000.

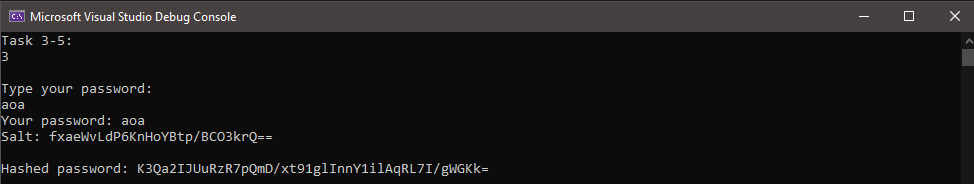
Варіант 26.

**Хід роботи**

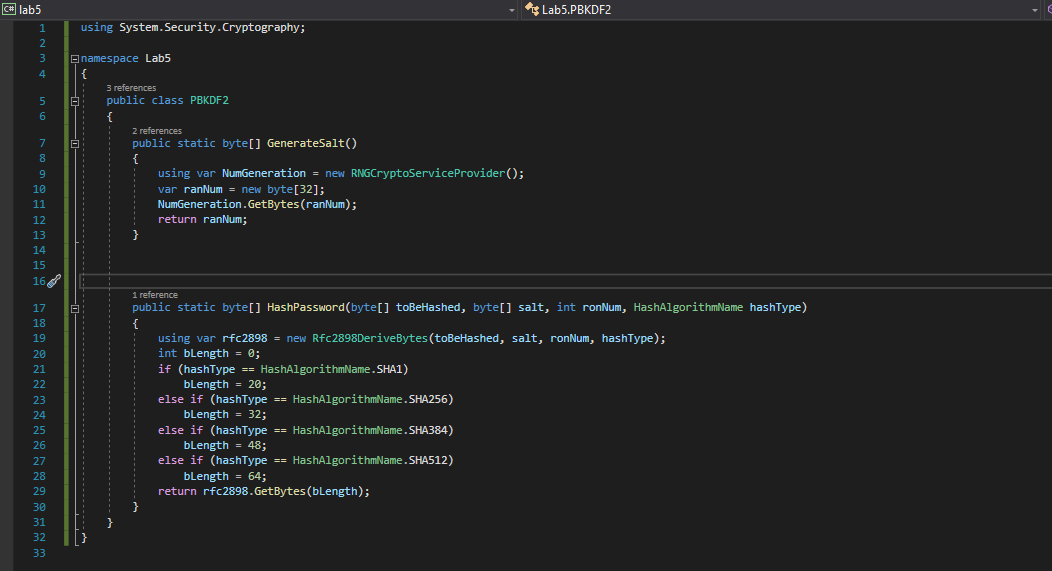
Завдання 3. Розробив клас SaltedHash, що реалізує хешування паролів із додаванням додаткової ентропії.



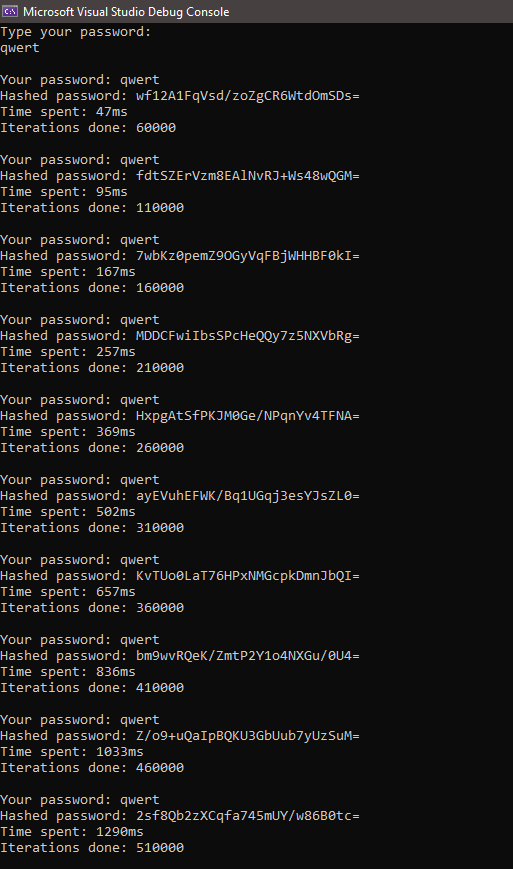
Ось, що виводиться на консоль:



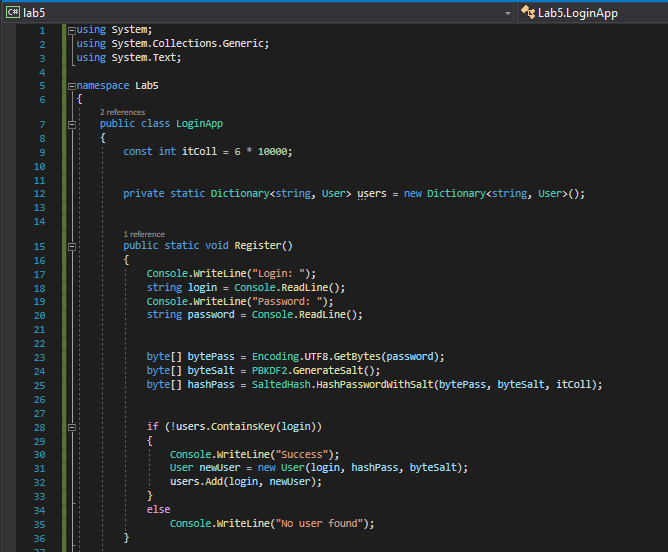
Завдання 4. Розробив клас PBKDF2, що генерує "сіль", задає алгоритм хешування (MD5, SHA1, SHA256, SHA384, SHA512) та обчислює хеш для заданого числа ітерацій. Створив програму, що обчислює час, витрачений на обчислення хешу для різного числа ітерацій (10 значень із кроком 50'000; перше значення = 6 \* 10'000 ).

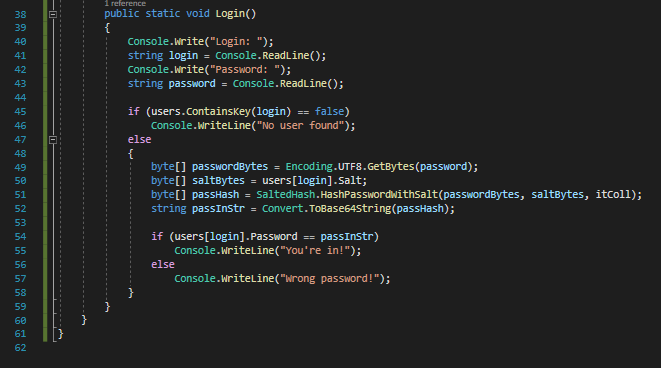


Ось, що виводиться на консоль:

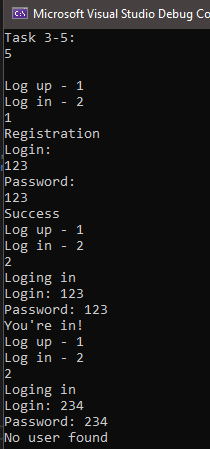


Завдання 5. Написаd програму, що реалізує хешування введеного пароля під час реєстрації користувача та зберігає логін, пароль та "сіль" у пам'яті. Реалізував можливість автентифікації за логіном і паролем. Число ітерацій = 6 \* 10'000.

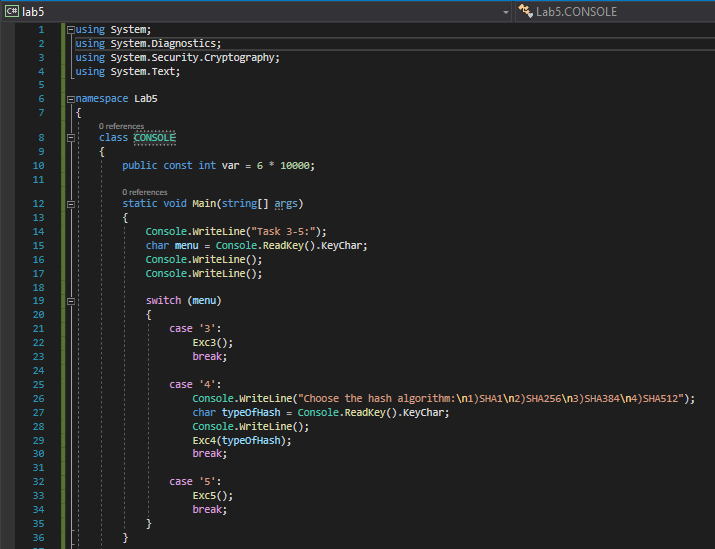


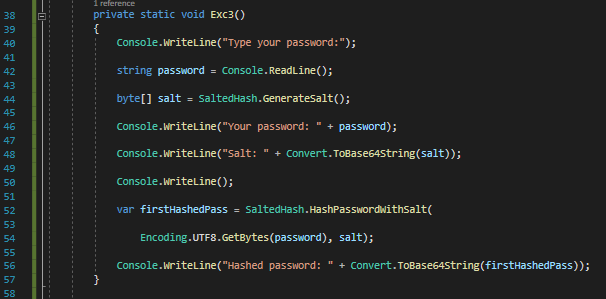


Ось, що виводиться на консоль:

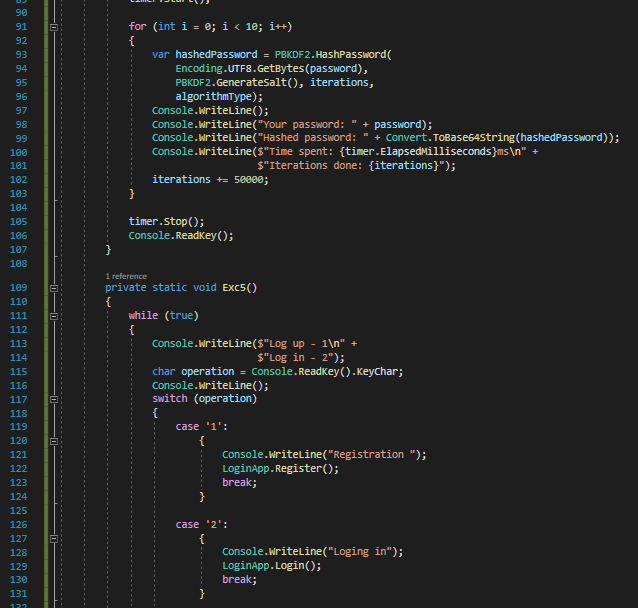


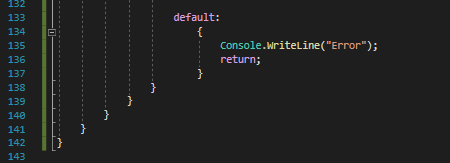
Вивід через клас CONSOLE











**Висновок**: отже після виконання роботи зрозумів, що алгоритми хешування не так сильно впливають на час виконання, на відміну від кількості ітерацій.