Основи інформаційної безпеки

ПЗ №5

Тема: “Безпечне зберігання паролів”

Підготував:

Студент групи МІТ-21

Міхальський Владислав

Основні способи зберігання паролів:

1) Зберігання паролів у відкритому вигляді

2) Шифрування паролів

3) Хешування паролів

4) Використання хешу та солі для зберігання пароля

5) Використання спеціальних функцій

Переваги й недоліки цих способів:

Зберігання паролів у відкритому вигляді є дуже небезпечним, оскільки якщо база паролів буде вкрадена, у зловмисників буде доступ до всіх акаунтів користувачів. Це може призвести до катастрофічних наслідків.

Шифрувати паролі це однозначно краще ніж зберігати їх у відкритому вигляді, але цей спосіб не є цілком досконалим. За допомогаю шифрування, ви шифруєте пароль за допомогою секретного ключа й розшифровуєте за допомогою цього ж ключа. Але, все ж таки цей спосіб не є найкращим, тому що, якщо зловмисник отримає секретний ключ та базу даних паролів якоїсь компанії, він зможе отримати доступ до всіх акаунтів користувачів. А саме це, призводить до катастрофічних наслідків. Щоб захистити користувачів, прийдеться роботи новий секретний ключ та змінювати всі паролі користувачів, що є майже неможливим. Тому цей спосіб не є найкращим в порівнянні з іншими методами.

Шифрування паролів це був хороший метод, до тих пір, поки не з’явилися райдужні таблиці. Райдужні таблиці – це таблиці в яких прописані можливі комбінації символів, поширені паролі й готові хеші для цих паролів, які користувачі можуть використовувати при реєстрації. За допомогою цих таблиць зловмисники можуть легко визначити паролі користувачів. Саме через це, з’явився метод хешування паролів з сіллю.

Сіль – це випадкова комбінація байтів, яка додається до кожного пароля введеного користувачем. Для кожного пароля існує своя сіль, тому для двох однакових паролів буде існувати кардинально різний хеш, тому що сіль – це випадкова комбінація байтів.

Також використовують функції для того, щоб збільшити надійність зберігання паролів. Функції хешують вже захешовані паролі разом із сіллю багато разів й через це підібрати такі паролі стає майже неможливим. Отже, зараз найбезпечніший метод зберігання паролів – це метод хешування з використанням солі, а потім хешування вже захешованого пароля десятки тисяч разів.

Завдання №3

using System;

using System.Security.Cryptography;

using System.Text;

namespace lab5.\_3

{

public class SaltedHash

{

public static byte[] GenerateSalt()

{

const int saltLength = 32;

using (var randomNumberGenerator = new RNGCryptoServiceProvider())

{

var randomNumber = new byte[saltLength];

randomNumberGenerator.GetBytes(randomNumber);

return randomNumber;

}

}

private static byte[] Combine(byte[] first, byte[] second)

{

var ret = new byte[first.Length + second.Length];

Buffer.BlockCopy(first, 0, ret, 0, first.Length);

Buffer.BlockCopy(second, 0, ret, first.Length, second.Length);

return ret;

}

public static byte[] HashPasswordWithSalt(byte[] toBeHashed, byte[] salt)

{

using (var sha256 = SHA256.Create())

{

return sha256.ComputeHash(Combine(toBeHashed, salt));

}

}

}

class Program

{

static void Main(string[] args)

{

const string password = "JustPassw0rd";

byte[] salt = SaltedHash.GenerateSalt();

Console.WriteLine("Password: " + password);

Console.WriteLine("Salt: " + Convert.ToBase64String(salt));

Console.WriteLine();

var hashedPassword = SaltedHash.HashPasswordWithSalt(Encoding.UTF8.GetBytes(password), salt);

Console.WriteLine("Hashed Password: " + Convert.ToBase64String(hashedPassword));

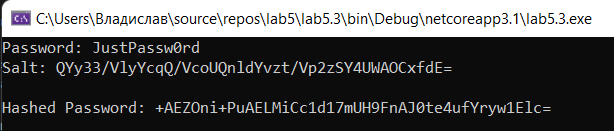
Console.ReadLine();

}

}

}

Результат виконання програми:



Завдання №4

using System;

using System.Security.Cryptography;

using System.Text;

namespace lab5.\_3

{

public class SaltedHash

{

public static byte[] GenerateSalt()

{

const int saltLength = 32;

using (var randomNumberGenerator = new RNGCryptoServiceProvider())

{

var randomNumber = new byte[saltLength];

randomNumberGenerator.GetBytes(randomNumber);

return randomNumber;

}

}

private static byte[] Combine(byte[] first, byte[] second)

{

var ret = new byte[first.Length + second.Length];

Buffer.BlockCopy(first, 0, ret, 0, first.Length);

Buffer.BlockCopy(second, 0, ret, first.Length, second.Length);

return ret;

}

public static byte[] HashPasswordWithSalt(byte[] toBeHashed, byte[] salt)

{

using (var sha256 = SHA256.Create())

{

return sha256.ComputeHash(Combine(toBeHashed, salt));

}

}

}

class Program

{

static void Main(string[] args)

{

const string password = "JustPassw0rd";

byte[] salt = SaltedHash.GenerateSalt();

Console.WriteLine("Password: " + password);

Console.WriteLine("Salt: " + Convert.ToBase64String(salt));

Console.WriteLine();

var hashedPassword = SaltedHash.HashPasswordWithSalt(Encoding.UTF8.GetBytes(password), salt);

Console.WriteLine("Hashed Password: " + Convert.ToBase64String(hashedPassword));

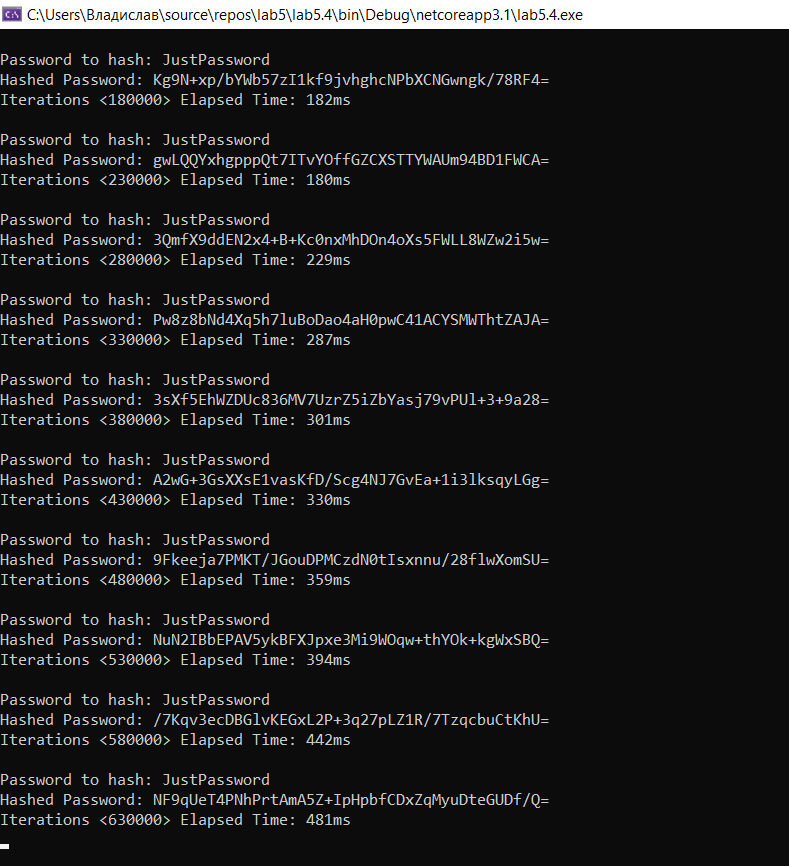
Console.ReadLine();

}

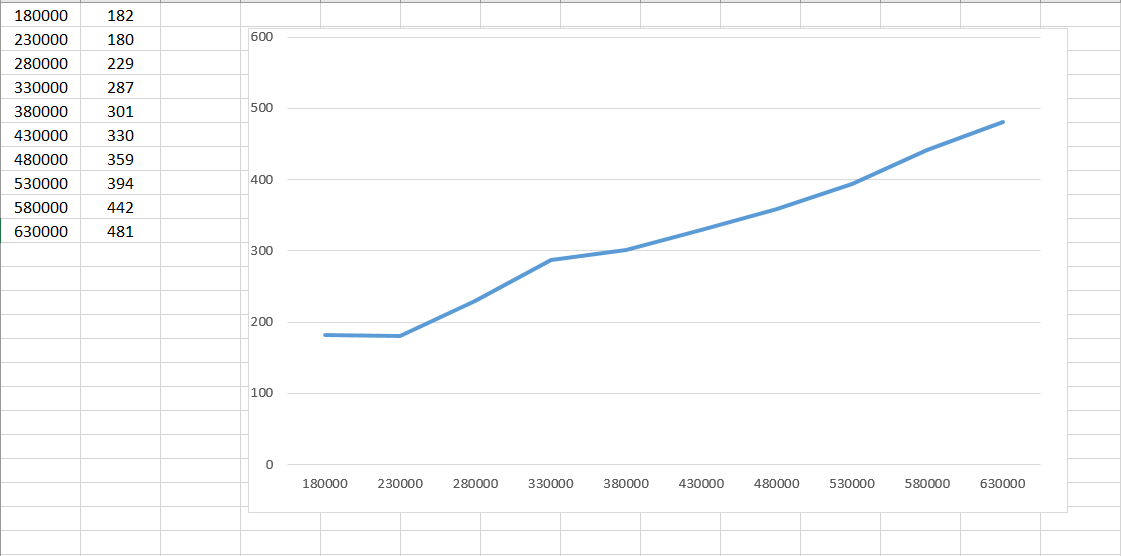
}

}

Результат виконання програми:



Графік:



Завдання №5

using System;

using System.Security.Cryptography;

using System.Text;

namespace lab5.\_5

{

public class PBKDF2

{

public static byte[] GenerateSalt()

{

using (var randomNumberGenerator = new RNGCryptoServiceProvider())

{

var randomNumber = new byte[32];

randomNumberGenerator.GetBytes(randomNumber);

return randomNumber;

}

}

public static byte[] HashPasswordSHA256(byte[] toBeHashed, byte[] salt, int numberOfRounds)

{

using (var rfc2898 = new Rfc2898DeriveBytes(toBeHashed, salt, numberOfRounds, HashAlgorithmName.SHA256))

{

return rfc2898.GetBytes(32);

}

}

}

class Program

{

static void Main(string[] args)

{

Console.WriteLine("For registration enter login and password.");

Console.WriteLine("Enter login: ");

string login = Convert.ToString(Console.ReadLine());

byte[] salt = PBKDF2.GenerateSalt();

Console.WriteLine("Enter password: ");

string password = Convert.ToBase64String(PBKDF2.HashPasswordSHA256(Encoding.Unicode.GetBytes(Convert.ToString(Console.ReadLine())), salt, 180000));

Console.WriteLine("Registration complete!");

Console.WriteLine("To log in, please, enter your credentials:");

Console.WriteLine("Enter login: ");

string enteredLogin = Convert.ToString(Console.ReadLine());

Console.WriteLine("Enter password: ");

string enteredPassword = Convert.ToBase64String(PBKDF2.HashPasswordSHA256(Encoding.Unicode.GetBytes(Convert.ToString(Console.ReadLine())), salt, 180000));

if (login != enteredLogin)

{

Console.WriteLine("Entered login is incorrect!");

}

else if (password != enteredPassword)

{

Console.WriteLine("Entered password is incorrect!");

}

else

{

Console.WriteLine("Authorization complete!");

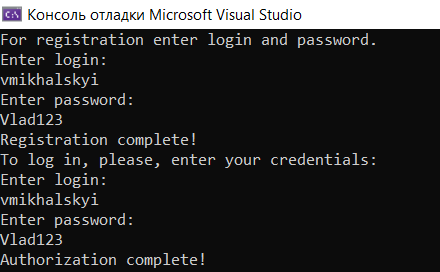
}

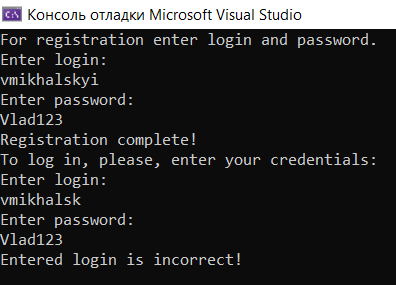
}

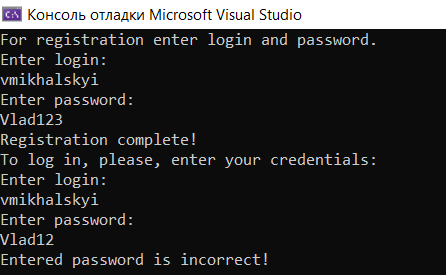
}

}

Результати:







Посилання на гітхаб репозиторій - <https://github.com/vmikhalskyi/Security-basics-2021.git>

Висновок: у ході лабораторної роботи я навчився, генерувати випадкову сіль, що значно покращує безпеку будь-якого пароля. Навчився хешувати паролі різними алгоритмами хешування разом із сіллю. Навчився хешувати декілька тисяч разів вже захешований пароль разом із сіллю й визначив час, який витрачається на певну кількість таких ітерацій. Дізнався що називають райдужними таблицями.