Звіт до практичної 5

Ремез Максим міт-21

Завдання 3

using System;

using System.Security.Cryptography;

using System.Text;

namespace Task5.\_2

{

public class Hash

{

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 = "My name is Max";

byte[] salt = Hash.GenerateSalt();

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

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

Console.WriteLine();

var hashedPassword1 = Hash.HashPasswordWithSalt(Encoding.UTF8.GetBytes(password), salt);

Console.WriteLine("Hashed Password = " + Convert.ToBase64String(hashedPassword1));

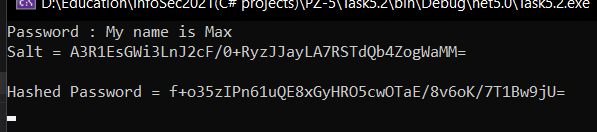
Console.ReadLine();

}

}

}

Результат у консолі:



Завдання 4

using System;

using System.Diagnostics;

using System.Security.Cryptography;

using System.Text;

namespace Task4

{

//клас PBKDF2

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[] HashPassword(byte[] toBeHashed, byte[] salt, int numberOfRounds, System.Security.Cryptography.HashAlgorithmName hashAlgorithm)

{

using (var rfc2898 = new Rfc2898DeriveBytes(toBeHashed, salt, numberOfRounds, hashAlgorithm))

{

return rfc2898.GetBytes(20);

}

}

}

class Program

{

static void Main(string[] args)

{

//число ітерацій - номер варіанта\*10000 з кроком 50к (10шт)

const string ToHashPassword= "My name is Max";

HashPassword(ToHashPassword, 240000);

HashPassword(ToHashPassword, 290000);

HashPassword(ToHashPassword,340000);

HashPassword(ToHashPassword, 390000);

HashPassword(ToHashPassword,440000);

HashPassword(ToHashPassword, 490000);

HashPassword(ToHashPassword, 540000);

HashPassword(ToHashPassword, 590000);

HashPassword(ToHashPassword, 640000);

HashPassword(ToHashPassword, 690000);

Console.ReadLine();

}

private static void HashPassword(string passwordToHash, int numberOfRounds)

{

var SS = new Stopwatch();

SS.Start();

var hashedPassword = PBKDF2.HashPassword(Encoding.UTF8.GetBytes(passwordToHash), PBKDF2.GenerateSalt(), numberOfRounds, HashAlgorithmName.SHA512);

SS.Stop();

Console.WriteLine();

Console.WriteLine("Password to hash : " + passwordToHash);

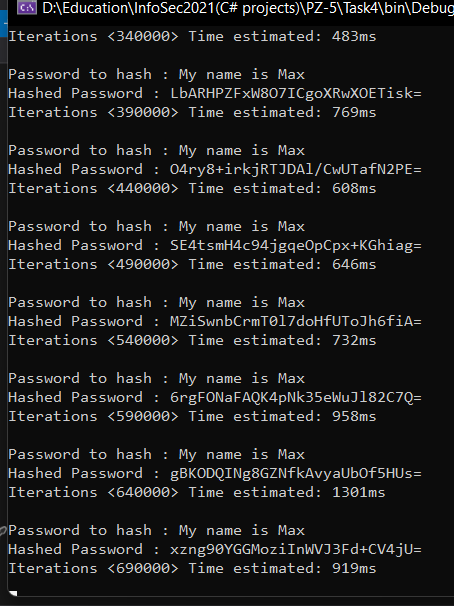
Console.WriteLine("Hashed Password : " + Convert.ToBase64String(hashedPassword));

Console.WriteLine("Iterations <" + numberOfRounds + "> Time estimated: " + SS.ElapsedMilliseconds + "ms");

}

}

}



Протягом роботи я навчився генерувати випадкову сіль різними алгоритмами хешування, що значно покращує безпеку будь-якого пароля. Хешувння проводилося тисячі разів із захешованим паролем та сіллю і визначив час, який витрачається на певну кількість таких ітерацій.