Основи інформаційної безпеки

ПЗ №5

Тема: “Безпечне зберігання паролів”

Підготував:

Студент групи МІТ-21

Пономаренко Владислав

(З-3)

(КОД)

using System;

using System.Security.Cryptography;

using System.Text;

namespace lab5x3

{

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)

{

Console.WriteLine("Enter password");

string password = Console.ReadLine();

byte[] salt = SaltedHash.GenerateSalt();

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

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

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

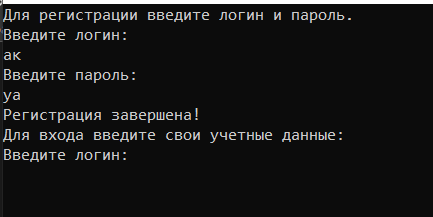
Console.WriteLine("Hashed Password: " + Convert.ToBase64String(hashedPassword));

}

}

}

(РЕЗУЛЬТАТ)



(З-4)

(КОД)

using System;

using System.Diagnostics;

using System.Security.Cryptography;

using System.Text;

namespace lab5x4

{

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[] HashPasswordMD5(byte[] toBeHashed, byte[] salt, int numberOfRounds)

{

using (var rfc2898 = new Rfc2898DeriveBytes(toBeHashed, salt, numberOfRounds, HashAlgorithmName.MD5))

{

return rfc2898.GetBytes(16);

}

}

public static byte[] HashPasswordSHA1(byte[] toBeHashed, byte[] salt, int numberOfRounds)

{

using (var rfc2898 = new Rfc2898DeriveBytes(toBeHashed, salt, numberOfRounds, HashAlgorithmName.SHA1))

{

return rfc2898.GetBytes(20);

}

}

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

{

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

{

return rfc2898.GetBytes(32);

}

}

public static byte[] HashPasswordSHA384(byte[] toBeHashed, byte[] salt, int numberOfRounds)

{

using (var rfc2898 = new Rfc2898DeriveBytes(toBeHashed, salt, numberOfRounds, HashAlgorithmName.SHA384))

{

return rfc2898.GetBytes(48);

}

}

public static byte[] HashPasswordSHA512(byte[] toBeHashed, byte[] salt, int numberOfRounds)

{

using (var rfc2898 = new Rfc2898DeriveBytes(toBeHashed, salt, numberOfRounds, HashAlgorithmName.SHA512))

{

return rfc2898.GetBytes(64);

}

}

}

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Enter password");

string ToHash = Console.ReadLine();

HashPassword(ToHash, 220000);

HashPassword(ToHash, 720000);

HashPassword(ToHash, 1220000);

HashPassword(ToHash, 1720000);

HashPassword(ToHash, 2220000);

HashPassword(ToHash, 2720000);

HashPassword(ToHash, 3220000);

HashPassword(ToHash, 3720000);

HashPassword(ToHash, 4220000);

HashPassword(ToHash, 4720000);

Console.ReadLine();

}

private static void HashPassword(string password, int number)

{

var hash = new Stopwatch();

hash.Start();

var hashedPassword = PBKDF2.HashPasswordSHA256(Encoding.UTF8.GetBytes(password), PBKDF2.GenerateSalt(), number);

hash.Stop();

Console.WriteLine();

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

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

Console.WriteLine("Iterations <" + number + ">");

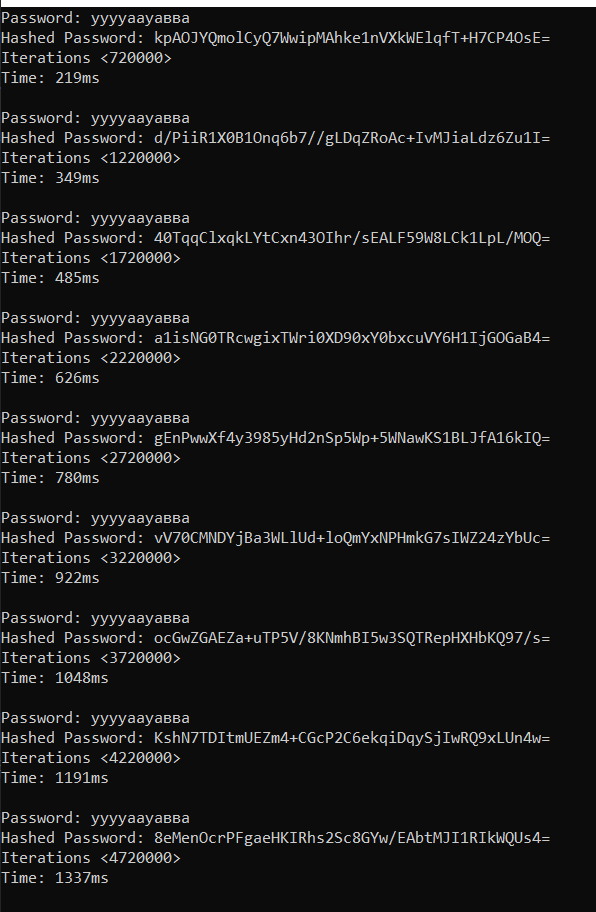
Console.WriteLine("Time: " + hash.ElapsedMilliseconds + "ms");

}

}

}

(РЕЗУЛЬТАТ)



(З-5)

using System;

using System.Security.Cryptography;

using System.Text;

namespace lab5x5

{

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[] HashSHA256(byte[] toBeHashed, byte[] salt, int numberOfRounds)

{

using (var sha256 = new Rfc2898DeriveBytes(toBeHashed, salt, numberOfRounds, HashAlgorithmName.SHA256))

{

return sha256.GetBytes(64);

}

}

}

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Для регистрации введите логин и пароль:");

Console.WriteLine("Введите логин: ");

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

byte[] salt = PBKDF2.GenerateSalt();

Console.WriteLine("Введите пароль: ");

string password = Convert.ToBase64String(PBKDF2.HashSHA256(Encoding.Unicode.GetBytes(Convert.ToString(Console.ReadLine())), salt, 220000));

Console.WriteLine("Регистрация завершена!");

Console.WriteLine("Для входа введите свои учетные данные:");

Console.WriteLine("Введите логин: ");

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

Console.WriteLine("Введите пароль: ");

string ePassword = Convert.ToBase64String(PBKDF2.HashSHA256(Encoding.Unicode.GetBytes(Convert.ToString(Console.ReadLine())), salt, 220000));

if (login != eLogin)

{

Console.WriteLine("Введен неверный логин!");

}

else if (password != ePassword)

{

Console.WriteLine("Введен неверный пароль!");

}

else

{

Console.WriteLine("Авторизация завершена!");

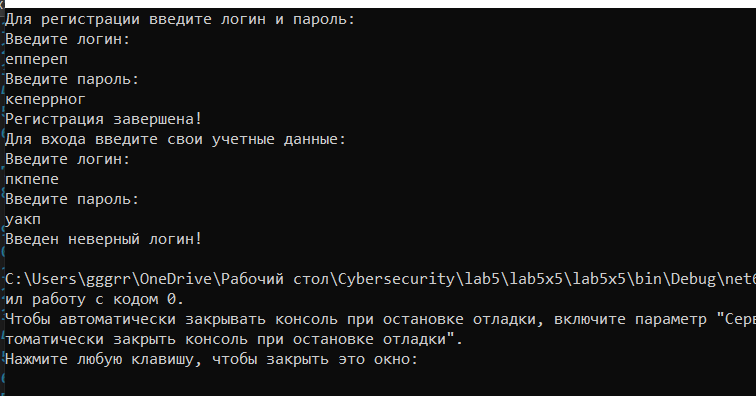
}

}

}

}

(РЕЗУЛЬТАТ)



(Посилання)

<https://github.com/VladPonomarenkoi/Cybersecurity>

(ВИСНОВОК)

У ході лабораторної роботи я навчився, генерувати випадкову сіль. Навчився хешувати паролі різними алгоритмами. Навчився хешувати захешований пароль разом із сіллю й визначив час, який витрачається на певну кількість таких ітерацій.