Основи інформаційної безпеки

ПЗ №11-12

“Автентифікація та авторизація користувачів”

Підготував:

Студент групи МІТ-21

Міхальський Владислав

Завдання №1

using System;

using System.Collections.Generic;

using System.Security;

using System.Security.Cryptography;

using System.Security.Principal;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading;

namespace lab11

{

class User

{

public string Login { get; set; }

public string PasswordHash { get; set; }

public byte[] Salt { get; set; }

public string[] Roles { get; set; }

}

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 Protector

{

private static Dictionary<string, User> \_users = new Dictionary<string, User>();

public static void Register(string userName, string password, string[] roles = null)

{

if (\_users.ContainsKey(userName))

{

Console.WriteLine("This user is already registered!");

} else

{

byte[] passwordToHash = Encoding.UTF8.GetBytes(password);

byte[] generatedSalt = PBKDF2.GenerateSalt();

byte[] hashedPassword = PBKDF2.HashPasswordSHA256(passwordToHash, generatedSalt, 2500);

string hashedPasswordString = Convert.ToBase64String(hashedPassword);

User newUser = new User();

newUser.Login = userName;

newUser.PasswordHash = hashedPasswordString;

newUser.Salt = generatedSalt;

newUser.Roles = roles;

\_users.Add(userName, newUser);

Console.WriteLine("New user was successfully registered!");

}

}

public static bool CheckPassword(string userName, string password)

{

if (\_users.ContainsKey(userName))

{

User user = \_users[userName];

byte[] enteredPasswordBytes = Encoding.UTF8.GetBytes(password);

byte[] hashedPassword = PBKDF2.HashPasswordSHA256(enteredPasswordBytes, user.Salt, 2500);

string enteredPasswordHash = Convert.ToBase64String(hashedPassword);

if (enteredPasswordHash == user.PasswordHash)

{

Console.WriteLine("This password is correct!");

return true;

} else

{

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

return false;

}

} else

{

Console.WriteLine("There is no registered user with this name");

return false;

}

}

public static void LogIn(string userName, string password)

{

// Створюється екземпляр автентифікованого користувача

var identity = new GenericIdentity(userName, "OIBAuth");

// Виконується прив’язка до ролей, до яких належить користувач

var principal = new GenericPrincipal(identity, \_users[userName].Roles);

// Створений екземпляр автентифікованого користувача з відповідними

// ролями присвоюється потоку, в якому виконується програма

System.Threading.Thread.CurrentPrincipal = principal;

Console.WriteLine("You were logged in!");

}

public static void OnlyForAdminsFeature()

{

// Перевірка того, що потік програми виконується автентифікованим користувачем із певними ролями

if (Thread.CurrentPrincipal == null)

{

throw new SecurityException("Thread.CurrentPrincipal cannot be null.");

}

// Перевірка того, що автентифікований користувач належить до ролі "Admins"

if (!Thread.CurrentPrincipal.IsInRole("Admin"))

{

throw new SecurityException("User must be a member of Admins to access this feature.");

}

// У разі, якщо перевірка пройшла успішно, виконується захищена частина програми

Console.WriteLine("You have access to this secure feature.");

}

}

class Program

{

static void Main(string[] args)

{

for (int i = 0; i < 4; i++)

{

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

Console.WriteLine("Enter login: ");

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

Console.WriteLine("Enter password: ");

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

Console.WriteLine("Enter roles separated by comma:");

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

Regex sWhitespace = new Regex(@"\s+"); // прибирає зайві пробіли

string rolesWithoutSpaces = sWhitespace.Replace(rolesString, "");

string[] roles = rolesWithoutSpaces.Split(',');

Protector.Register(login, password, roles);

Console.WriteLine();

}

Console.WriteLine("All users were registered");

Console.WriteLine();

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.ToString(Console.ReadLine());

if(Protector.CheckPassword(enteredLogin, enteredPassword))

{

Protector.LogIn(enteredLogin, enteredPassword);

try

{

Protector.OnlyForAdminsFeature();

}

catch (Exception ex)

{

Console.WriteLine($"{ex.GetType()}: {ex.Message}");

}

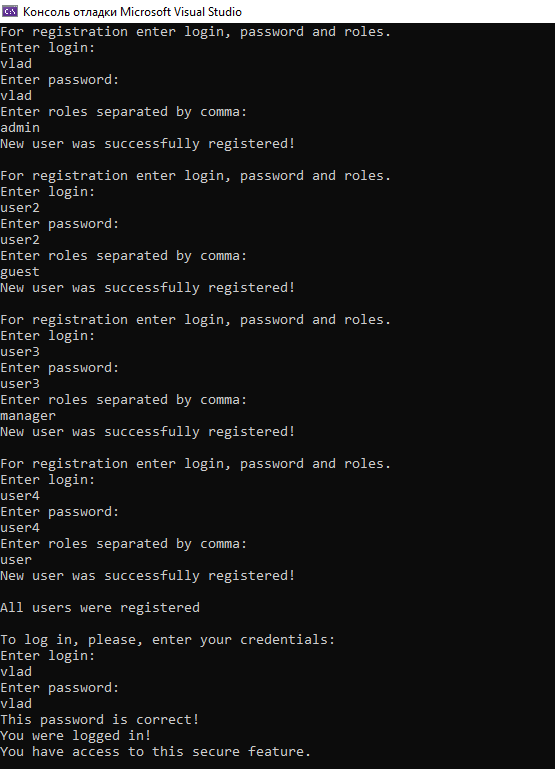
}

}

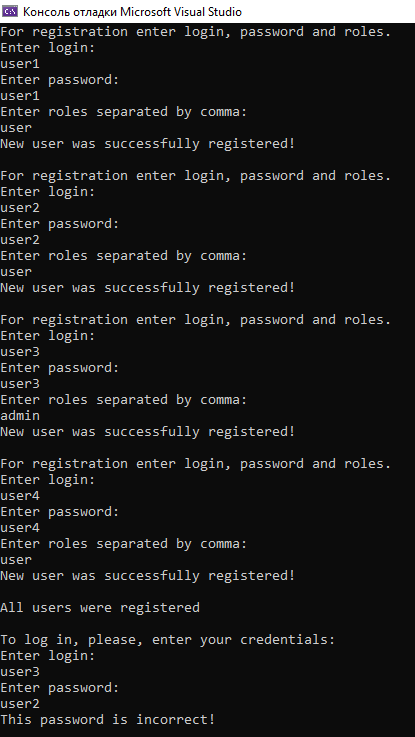
}

}

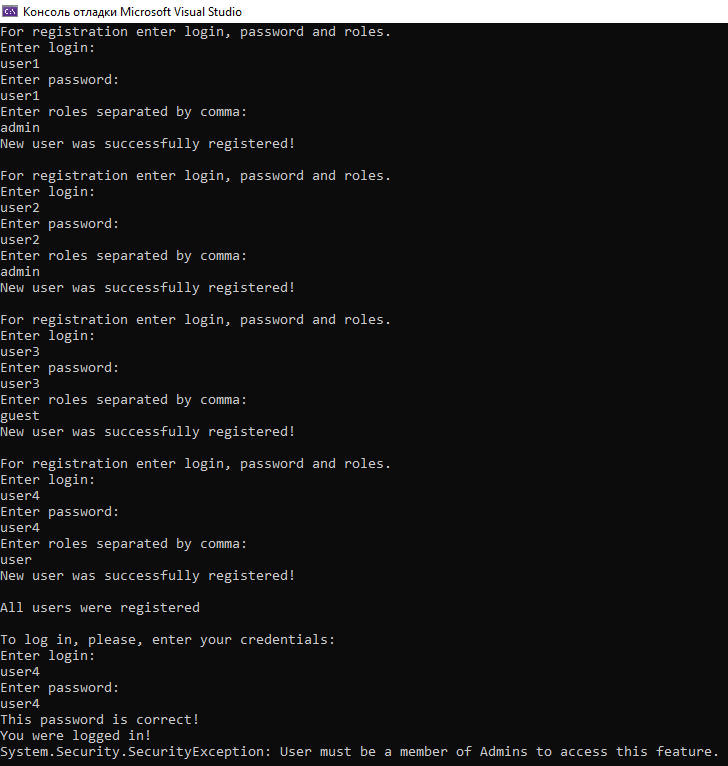
Результат (4 користувача успішно зареєструвалися та отримали свої ролі. Та автентифікація ролі адміністратора пройшла успішно):



Результат (4 користувача зареєструвалися успішно, автентифікація пройшла невдало):



Результат 4 користувача зареєструвалися успішно, автентифікація НЕ адміністратора пройшла вдало):



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

Висновок: у ході лабораторної роботи я навчився виконувати реєстрацію користувачів з виданням для них певних ролей. Я закріпив навики хешування пароля за допомогою алгоритму SHA256 разом із сіллю та кількістю ітерацій 2500. Також, зміг звіріти правильність пароля хешуючи отриманий пароль від користувача (для виконання входу користувачів).