Основи інформаційної безпеки

ПЗ №13-14

“Логування”

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;

using NLog;

namespace lab13

{

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

{

private static Logger log = NLog.LogManager.GetCurrentClassLogger();

public static byte[] GenerateSalt()

{

log.Trace("Generating of salt started");

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)

{

log.Trace("Hashing of password started");

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

{

return rfc2898.GetBytes(32);

}

}

}

class Protector

{

private static Logger log = NLog.LogManager.GetCurrentClassLogger();

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

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

{

log.Trace("Checking of user's existance started");

if (\_users.ContainsKey(userName))

{

log.Warn("Entered user is already registered");

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;

log.Trace("Registering of new user started");

\_users.Add(userName, newUser);

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

}

}

public static bool CheckPassword(string userName, string password)

{

log.Trace("Checking of user's existance started");

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);

log.Trace("Comparing of passwords started");

if (enteredPasswordHash == user.PasswordHash)

{

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

return true;

}

else

{

log.Warn("Entered password is incorrect");

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

return false;

}

}

else

{

log.Warn("Entered login is incorrect");

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

return false;

}

}

public static void LogIn(string userName, string password)

{

log.Trace("Creating of exemplar of authenticated user");

// 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()

{

log.Trace("Checking of user's authentication started");

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

if (Thread.CurrentPrincipal == null)

{

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

}

log.Trace("CHecking of user's role started");

// 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)

{

Logger log = NLog.LogManager.GetCurrentClassLogger();

log.Trace("Registration of users started");

log.Info("Registraion of users");

int i = 0;

int iCurr = 0;

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

{

log.Debug($"Change of i variable: Previous={iCurr} Current={i}");

iCurr = i;

log.Trace($"Registration of {i+1} user");

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();

}

log.Trace("Registration of users finished");

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

Console.WriteLine();

log.Trace("Entering of credentials of user to log in started");

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());

log.Info("Logging in of user");

log.Trace("Checking of entered password started");

if (Protector.CheckPassword(enteredLogin, enteredPassword))

{

log.Trace("Logging in of user started");

Protector.LogIn(enteredLogin, enteredPassword);

try

{

log.Trace("Checking for Admins feature started");

Protector.OnlyForAdminsFeature();

}

catch (Exception ex)

{

log.Error(ex, "No Admin role for user");

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

}

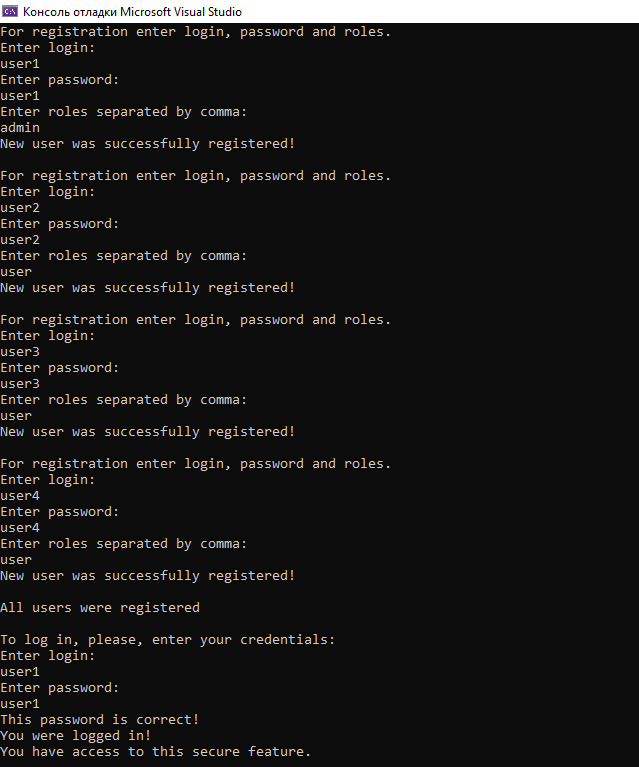
}

}

}

}

Результат:



Посилання на гітхаб репозиторій:

<https://github.com/vmikhalskyi/Security-basics-2021.git>

Висновок: у ході лабораторної роботи я повторив закріплені навички проведення реєстрації користувачів, хешування паролів з сіллю та хешування вже захешованих паролів декілька тисяч разів, що дає нам більший захист. Також, навчився написанню логів різних рівнів, а саме:

Trace - всі кроки виконання програми;

Debug - всі випадки зміни значень змінних та переходу у розгалуженнях/циклах;

Info - всі випадки, що стосуються предметної області (реєстрація, вхід, вихід користувача);

Warn - всі випадки "підозрілої" активності користувачів (невірний логін/пароль)

Error - всі випадки оброблених виняткових ситуацій;