ПЗ №3 Хеш-функції та перевірка цілісності інформації.

Беслюбняк Віталій МІТ-21

Хід Роботи

using System;

using System.Text;

using System.Security.Cryptography;

struct User

{

public string Hmac\_seq;

public int id;

public User(string Hmac\_seq, int id)

{

this.Hmac\_seq = Hmac\_seq;

this.id = id;

}

}

struct Users

{

public User[] users;

public int amount;

public Users()

{

this.users = new User[10];

this.amount = -1;

}

}

class Crypto

{

public static byte[] generate\_sec\_random\_number(int length)

{

var randomNumberGenerator = new RNGCryptoServiceProvider();

var randomNumber = new byte[length];

randomNumberGenerator.GetBytes(randomNumber);

return randomNumber;

}

public static byte[] ComputeHashMd5(byte[] dataForHash)

{

using (var md5 = MD5.Create())

{

return md5.ComputeHash(dataForHash);

}

}

public static byte[] ComputeHashSha1(byte[] toBeHashed)

{

using (var sha1 = SHA1.Create())

{

return sha1.ComputeHash(toBeHashed);

}

}

public static byte[] ComputeHashSha256(byte[] toBeHashed)

{

using (var sha256 = SHA256.Create())

{

return sha256.ComputeHash(toBeHashed);

}

}

public static byte[] ComputeHashSha384(byte[] toBeHashed)

{

using (var sHA384 = SHA384.Create())

{

return sHA384.ComputeHash(toBeHashed);

}

}

public static byte[] ComputeHashSha512(byte[] toBeHashed)

{

using (var sha512 = SHA512.Create())

{

return sha512.ComputeHash(toBeHashed);

}

}

}

class Console\_Gui

{

public static void MENU()

{

Console.Clear();

Console.WriteLine(" [\*] Choose option [\*]");

Console.WriteLine("[=========================]");

Console.WriteLine(" | 1. Hash |");

Console.WriteLine(" | 2. Hmac |");

Console.WriteLine(" | 3. Exit |");

Console.WriteLine("[=========================]");

}

public static void MENU\_HMAC()

{

Console.Clear();

Console.WriteLine(" [\*] Choose option [\*]");

Console.WriteLine("[=========================]");

Console.WriteLine(" | 1. Register |");

Console.WriteLine(" | 2. Login |");

Console.WriteLine(" | 3. Print |");

Console.WriteLine(" | 4. Back |");

Console.WriteLine("[=========================]");

}

public static void MENU\_HASH()

{

Console.Clear();

Console.WriteLine(" [\*] Choose option [\*]");

Console.WriteLine("[=========================]");

Console.WriteLine(" | 1. MD5 |");

Console.WriteLine(" | 2. SHA1 |");

Console.WriteLine(" | 3. SHA256 |");

Console.WriteLine(" | 4. SHA384 |");

Console.WriteLine(" | 5. SHA512 |");

Console.WriteLine(" | 6. Hack |");

Console.WriteLine(" | 7. Back |");

Console.WriteLine("[=========================]");

}

}

class Hmac

{

public static byte[] ComputeHmacsha256(byte[] toBeHashed, byte[] key)

{

using (var hmac = new HMACSHA256(key))

{

return hmac.ComputeHash(toBeHashed);

}

}

public static byte[] ComputeHmacsha1(byte[] toBeHashed, byte[] key)

{

using (var hmac = new HMACSHA1(key))

{

return hmac.ComputeHash(toBeHashed);

}

}

public static byte[] ComputeHmacsha512(byte[] toBeHashed, byte[] key)

{

using (var hmac = new HMACSHA512(key))

{

return hmac.ComputeHash(toBeHashed);

}

}

public static byte[] ComputeHmacmd5(byte[] toBeHashed, byte[] key)

{

using (var hmac = new HMACMD5(key))

{

return hmac.ComputeHash(toBeHashed);

}

}

}

class Globals

{

public static bool found;

public static bool exists;

}

class Program

{

static void Main(string[] args)

{

byte[] message = Crypto.generate\_sec\_random\_number(32);

Users listofusers = new Users();

do

{

Console.Clear();

Console\_Gui.MENU();

string? answer\_menu = Console.ReadLine();

switch (answer\_menu)

{

case "1":

Console.Clear();

Console\_Gui.MENU\_HASH();

string? answer\_hash = Console.ReadLine();

switch (answer\_hash)

{

case "1":

byte[] resultmd5 = Crypto.ComputeHashMd5(message);

Console.WriteLine(Convert.ToBase64String(resultmd5));

Console.ReadKey();

break;

case "2":

byte[] resultsha1 = Crypto.ComputeHashSha1(message);

Console.WriteLine(Convert.ToBase64String(resultsha1));

Console.ReadKey();

break;

case "3":

byte[] resultsha256 = Crypto.ComputeHashSha256(message);

Console.WriteLine(Convert.ToBase64String(resultsha256));

Console.ReadKey();

break;

case "4":

byte[] resultsha384 = Crypto.ComputeHashSha384(message);

Console.WriteLine(Convert.ToBase64String(resultsha384));

Console.ReadKey();

break;

case "5":

byte[] resultsha512 = Crypto.ComputeHashSha512(message);

Console.WriteLine(Convert.ToBase64String(resultsha512));

Console.ReadKey();

break;

case "6":

for (var i = 10000000; i <= 99999999; i++)

{

byte[] a = Encoding.Unicode.GetBytes(i.ToString());

byte[] res = Crypto.ComputeHashMd5(a);

Console.WriteLine("[#] Current key: {0} [#]", i);

if (Convert.ToBase64String(res) == "po1MVkAE7IjUUwu61XxgNg==" || Convert.ToBase64String(res) == "{564c8da6-0440-88ec-d453-0bbad57c6036}")

{

Console.WriteLine("[#] DECRYPTED [#]");

Console.WriteLine("[#] Key: '{0}' [#]", i);

Console.ReadKey();

break;

}

}

break;

case "7":

break;

}

break;

case "2":

Console.Clear();

Console\_Gui.MENU\_HMAC();

string? answer\_hmac = Console.ReadLine();

switch (answer\_hmac)

{

case "1":

if (listofusers.amount != 9)

{

Console.WriteLine("[\*] Enter new login [\*]");

string? login\_reg = Console.ReadLine();

Console.WriteLine("[\*] Enter new password [\*]");

string password\_reg\_str = Console.ReadLine();

byte[] password\_reg = Encoding.Unicode.GetBytes(password\_reg\_str);

string PotentialUser = Convert.ToBase64String(Hmac.ComputeHmacsha256(Encoding.UTF8.GetBytes(login\_reg), password\_reg));

for (int i = 0; i < 10; i++)

{

if (listofusers.users[i].Hmac\_seq == PotentialUser)

{

Console.WriteLine("[!] This login is already exists [!]");

Globals.exists = true;

Console.ReadKey();

break;

}

}

if (Globals.exists == true)

{

Globals.exists = false;

break;

}

listofusers.amount++;

User newUser = new User(PotentialUser, listofusers.amount);

listofusers.users[listofusers.amount] = newUser;

Console.WriteLine("[#] You have successfully registered! [#]");

Console.WriteLine($"\n[#] Your Login: {login\_reg} [#]\n\n[#] Password: {password\_reg\_str} [#]\n\n[#] HMAC sequence {PotentialUser} [#]");

Console.ReadKey();

login\_reg = "";

password\_reg\_str = "";

Array.Clear(password\_reg);

break;

}

else

{

Console.WriteLine("[!] DB is overfulled [!]");

Console.ReadKey();

break;

}

case "2":

Console.WriteLine("[\*] Enter login [\*]");

string login\_log = Console.ReadLine();

Console.WriteLine("[\*] Enter new password [\*]");

string password\_log\_str = Console.ReadLine();

byte[] password\_log = Encoding.Unicode.GetBytes(password\_log\_str);

string User = Convert.ToBase64String(Hmac.ComputeHmacsha256(Encoding.UTF8.GetBytes(login\_log), password\_log));

for (int i = 0; i < 10; i++)

{

if (listofusers.users[i].Hmac\_seq == User)

{

Console.WriteLine("[#] You have successfully signed in! [#]");

Console.WriteLine($"\n[#] Your Login: {login\_log} [#]\n\n[#] Password: {password\_log\_str} [#]\n\n[#] HMAC sequence {User} [#]");

Globals.found = true;

Console.ReadKey();

break;

}

}

if (Globals.found == false)

{

Console.WriteLine("[!] Incorrect user data. Nothing found [!]");

Console.ReadKey();

}

Globals.found = false;

login\_log = "";

password\_log\_str = "";

Array.Clear(password\_log);

break;

case "3":

if (listofusers.amount == -1)

{

Console.WriteLine("[!] DB is Empty [!]");

Console.ReadKey();

}

else

{

for (int i = 0; i <= listofusers.amount; i++)

{

Console.WriteLine($"[#] amount: {listofusers.users[i].id} - \'{listofusers.users[i].Hmac\_seq}\' [#]");

}

Console.ReadKey();

}

break;

case "4":

break;

}

break;

case "3":

Environment.Exit(0);

break;

}

} while (true);

}

}