Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4**

Преподаватель \_\_\_\_\_\_\_\_\_\_\_ А.Н. Шниперов

подпись, дата инициалы, фамилия

Студент КИ15–17Б, 031510065 \_\_\_\_\_\_\_\_\_\_\_ Т.В. Радионов

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2017

1. **Цели лабораторной работы**

* ознакомиться с основами методов аутентификации;
* освоить механизмы аутентификации и авторизации в WEB-ориентированных информационных системах;
* ознакомится с базовыми моделями доступа в задачах защиты от несанкционированного доступа (НСД);
* изучить алгоритм Диффи-Хеллмана и особенности его применения в задачах разработки механизмов аутентификации;
* освоить использование хеш-функций в задачах безопасного хранения и передачи учётных данных пользователей;
* познакомиться на практике с технологических особенностями разработки механизмов аутентификации в web-ориентированных информационных системах.

1. **Задание лабораторной работы**

Разработать эффективную и безопасную (в заданных условиях действия нарушителя) механизм централизованной аутентификации пользователей и последующей авторизации в web-ориентированной ИС. Разработать её программный прототип, реализующий:

* разработанную Вами схему аутентификации/авторизации;
* одну из базовых моделей доступа (см. теоретические сведения) или некую модифицированную модель на её основе;
* минимальный функционал ИС для пользователей (тематика самой ИС оставляется на выбор студента).

1. **Описание аутентификации и авторизации пользователя в программе**

*Аутентификация:*

После заполнения Пользователем необходимых полей в окне браузера – логина и пароля, и проверки на бота в системе reCaptcha (если данная система определит что-то неладное, то прервет аутентификацию), программа шифрует данные по методу RSA (аббревиатура от фамилий Rivest, Shamir и Adleman), обладая открытым ключом, который она получает от сервера при начале аутентификации и отсылает их на сервер, где происходит расшифрование данных закрытым ключом, известным лишь серверу. Далее происходит сама аутентификация (проверка наличия пользователя с введенным им паролем в базе данных пользователей).

После успешной аутентификации пользователя происходит проверка наличия токена: если токен есть в базе данных сессий (и, разумеется, он связан с авторизируемым пользователем, т.е. сессия существует), то проверяется соответствие IP-адреса клиента с IP-адресом пользователя в сессии (это дополнительная мера защиты) и, в случае успеха, от сервера клиенту возвращается данный токен. В противном же случае происходит генерация нового токена, добавление сессии в базу данных сессий (где помимо связки пользователь-токен еще присутствует поле с временем создания токена и IP-адресом клиента, для которого открывается сессия).

*Авторизация:*

Теперь, когда клиент обладает токеном, он может получать доступ к ресурсам, которые ему доступны путем отправки запроса к доступу с токеном на сервер и получением ответа. А именно это происходит так: на сервер приходит токен, он обрабатывается (определяется обладатель токена, его полномочия в системе), также проверяется «время жизни» токена (при его истечении потребуется заново пройти аутентификацию, т.к. сервер запретит доступ к запрашиваемому ресурсу) и IP-адрес клиента, отправившего запрос, с IP-адресом пользователя, на которого открыта сессия. Если все этапы пройдены успешно, сервер возвращает ответ пользователю, который позволяет получить доступ к ресурсу, в противном случае доступа нет.

*Достоинства и недостатки:*

Преимуществом такой схемы аутентификации и авторизации является то, что злоумышленник, попытавшись перехватить запрос пользователя на сервер во время аутентификации получит логин и пароль, которые зашифрованы методом RSA, дешифрование которых маловероятно. Также система защищена от брута путем введения при аутентификации дополнительной проверки на бота через систему reCaptcha.

Однако есть недостаток в такой системе: токен, который возвращает сервер, может перехватить злоумышленник (а он в открытом виде). Однако мера с проверкой IP-адреса при авторизации клиента исключит всех, у кого IP-адрес отличен, т.е. доступ на ограниченное время может получить только тот, кто воспользовался устройством, с которого осуществлялась аутентификация или же получил доступ к каналу в том месте, где IP-адрес принадлежит пользователю.

1. **Описание web-ориентированной системы, функционала программы и модели доступа**

Web-ориентированная система представляет из себя клиента (веб-сайт), расположенный на Internet Information Services (IIS) и сервер (представляет из себя решение от Microsoft, которое называется Windows Communication Foundation – позволяет создавать службы (обработчиков транзакций), которые обрабатывают запросы от клиента, набор которых и можно назвать web-сервером, который расположен на IIS).

Клиенту изначально доступна форма авторизации, где необходимо ввести логин и пароль и пройти проверку на бота, после чего он может нажать на кнопку «login». Если возникнет какое-либо исключение или ошибка, сообщение отобразится над формой заполняемых полей с соответствующим текстом ошибки.

После аутентификации и авторизации перед пользователем возникнет две кнопки: «User» и «Admin». В зависимости от того, какими правами он наделен (права ролевые – Администратор и Пользователь), то при нажатии на каждую из кнопок появится соответствующее сообщение с текстом «Доступ получен, %Role%» или «У вас нет прав быть %Role%!», где %Role% - роль пользователя в системе.

1. **Листинг программы**
2. Интерфейс сервиса сервера IServiceAuthorization.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.Text;

using System.Security.Cryptography;

namespace WcfService1

{

// ПРИМЕЧАНИЕ. Команду "Переименовать" в меню "Рефакторинг" можно использовать для одновременного изменения имени интерфейса "IServiceAuthorization" в коде и файле конфигурации.

[ServiceContract]

public interface IServiceAuthorization

{

[OperationContract]

bool AccessAction(string token, int rights);

[OperationContract]

string Authentication(string login, string password);

[OperationContract]

RSAParameters MakeCryptoKey();

}

}

1. Логика сервиса сервера ServiceAuthorization.svc.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.Text;

using MySql.Data.MySqlClient;

using System.Security.Cryptography;

using System.ServiceModel.Channels;

namespace WcfService1

{

// ПРИМЕЧАНИЕ. Команду "Переименовать" в меню "Рефакторинг" можно использовать для одновременного изменения имени класса "ServiceAuthorization" в коде, SVC-файле и файле конфигурации.

// ПРИМЕЧАНИЕ. Чтобы запустить клиент проверки WCF для тестирования службы, выберите элементы ServiceAuthorization.svc или ServiceAuthorization.svc.cs в обозревателе решений и начните отладку.

public class ServiceAuthorization : IServiceAuthorization

{

static string myConnectionString = "Database=database1;Data Source=localhost;User Id=root"; // строка соединения сервера с БД

static MySqlConnection myConnection = new MySqlConnection(myConnectionString); // создание соединения с БД

static int timeLifeToken = 30;

static RSAParameters privateKey;

/// <summary>

/// Создание открытого и закрытого ключей RSA

/// </summary>

/// <returns></returns>

public RSAParameters MakeCryptoKey()

{

RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();

privateKey = RSA.ExportParameters(true);

return RSA.ExportParameters(false);

}

/// <summary>

/// Расшифрование RSA

/// </summary>

/// <param name="DataToDecrypt"></param>

/// <param name="RSAKeyInfo"></param>

/// <param name="DoOAEPPadding"></param>

/// <returns></returns>

private byte[] RSADecrypt(byte[] DataToDecrypt, RSAParameters RSAKeyInfo, bool DoOAEPPadding)

{

RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();

RSA.ImportParameters(RSAKeyInfo);

return RSA.Decrypt(DataToDecrypt, DoOAEPPadding);

}

/// <summary>

/// Генерация токена

/// </summary>

/// <returns></returns>

private string GenerateToken()

{

string token = "";

string letters = "AaBbCcDd0123456789EeFfGgHh0123456789IiJjKkLl0123456789MmNnOoPp0123456789QqRrSsTt0123456789UuVvWwXx0123456789YyZz";

Random rnd = new Random();

for (int i = 0; i < 16; i++)

{

int result = rnd.Next(letters.Length - 1);

token += letters[result];

}

return token;

}

/// <summary>

/// Проверка срока жизни токена в БД

/// </summary>

/// <param name="token"></param>

/// <param name="login"></param>

/// <returns></returns>

private bool CheckToken(string token, string login)

{

// Формирование запроса к БД

MySqlCommand cmd = new MySqlCommand();

MySqlDataReader reader;

// получение ip-адреса клиента и проверка на совпадение с ip-адресом существующей сесссии

OperationContext context = OperationContext.Current;

MessageProperties prop = context.IncomingMessageProperties;

RemoteEndpointMessageProperty endpoint = prop[RemoteEndpointMessageProperty.Name] as RemoteEndpointMessageProperty;

string ipAddr = endpoint.Address;

cmd.CommandText = string.Format("SELECT \* FROM sessions WHERE sessions.IPAddress = '{0}' AND sessions.Token = '{1}'", ipAddr, token);

cmd.Connection = myConnection;

myConnection.Open();

reader = cmd.ExecuteReader();

string resultIPAddr = "";

while (reader.Read())

{

resultIPAddr = reader.GetString(3);

}

myConnection.Close();

// IP-адрес сессии есть?

if (resultIPAddr == "")

return false;

// Проверка времени жизни запрашиваемого токена

cmd.CommandText = string.Format("DELETE FROM sessions WHERE (sessions.Token = '{0}') AND " +

"(ADDDATE(sessions.CreateTime, INTERVAL {1} SECOND) < NOW())", token, timeLifeToken);

cmd.Connection = myConnection;

myConnection.Open();

int result = cmd.ExecuteNonQuery();

myConnection.Close();

// Проверка на выполнение удаления токена из БД

if (result > 0)

return false;

cmd.CommandText = string.Format("SELECT \* FROM sessions WHERE sessions.Token = '{0}'", token);

cmd.Connection = myConnection;

myConnection.Open();

// Отправка запроса к БД и возврат ответа

reader = cmd.ExecuteReader();

// Преобразование ответа в строку и обработка результата

string resultToken = "";

while (reader.Read())

{

resultToken = reader.GetString(1);

}

myConnection.Close();

// Сессия существует?

if (resultToken == "")

return false;

else

return true;

}

private bool CheckToken(string login)

{

// Формирование запроса к БД

MySqlCommand cmd = new MySqlCommand();

cmd.CommandText = string.Format("SELECT \* FROM sessions WHERE sessions.Login = '{0}'", login);

cmd.Connection = myConnection;

myConnection.Open();

// Отправка запроса к БД и возврат ответа

MySqlDataReader reader = cmd.ExecuteReader();

// Преобразование ответа в строку и обработка результата

string resultLogin = "";

while (reader.Read())

{

resultLogin = reader.GetString(0);

}

myConnection.Close();

// Если сессия не существует

if (resultLogin == "")

{

return false;

}

// Проверка истечения времени сессии

cmd.CommandText = string.Format("DELETE FROM sessions WHERE (sessions.Login = '{0}') AND " +

"(ADDDATE(sessions.CreateTime, INTERVAL {1} SECOND) < NOW())", login, timeLifeToken);

cmd.Connection = myConnection;

myConnection.Open();

int result = cmd.ExecuteNonQuery();

myConnection.Close();

// Проверка на выполнение каких-либо удалений из БД

if (result > 0)

return false;

else

return true;

}

/// <summary>

/// Разрешение на действие для обладателя токена

/// </summary>

/// <param name="token"></param>

/// <param name="rights"></param>

/// <returns></returns>

public bool AccessAction(string token, int rights)

{

MySqlCommand cmd = new MySqlCommand();

// Если время жизни токена истекло

if (!CheckToken(token, ""))

return false;

// Запрос к БД sessions с возвратом Login

cmd.CommandText = string.Format("SELECT \* FROM sessions WHERE sessions.Token = '{0}'", token);

cmd.Connection = myConnection;

myConnection.Open();

string resultLogin = "";

// Отправка запроса к БД и возврат ответа

MySqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

resultLogin = reader.GetString(0);

}

myConnection.Close();

// Если в БД не нашлось пользователей с таким токеном, то запрет доступа

if (resultLogin == "")

{

return false;

}

cmd.CommandText = string.Format("SELECT \* FROM users WHERE users.Login = '{0}'", resultLogin);

cmd.Connection = myConnection;

myConnection.Open();

int resultRights = 0;

// Отправка запроса к БД и возврат ответа

reader = cmd.ExecuteReader();

while (reader.Read())

{

resultRights = Convert.ToInt32(reader.GetString(2));

}

myConnection.Close();

// Если в БД не нашлось пользователей с таким правом, то запрет доступа

if (resultRights == 0)

{

return false;

}

// Если права запроса совпадают с правами в БД

if (resultRights == rights)

return true;

return false;

}

/// <summary>

/// Аутентификация пользователя

/// </summary>

/// <param name="login"></param>

/// <param name="password"></param>

/// <returns></returns>

public string Authentication(string login, string password)

{

// Попытка установить соединение

try

{

myConnection.Open();

}

catch (Exception ex)

{

return "404";

}

// Расшифрование логина и пароля

UnicodeEncoding byteConverter = new UnicodeEncoding();

byte[] input = Convert.FromBase64String(login);

byte[] output = RSADecrypt(input, privateKey, false);

login = byteConverter.GetString(output);

input = Convert.FromBase64String(password);

output = RSADecrypt(input, privateKey, false);

password = byteConverter.GetString(output);

// Формирование запроса к БД

MySqlCommand cmd = new MySqlCommand();

cmd.CommandText = string.Format("SELECT \* FROM users WHERE users.Login = '{0}' AND users.Password = '{1}'", login, password);

cmd.Connection = myConnection;

// Отправка запроса к БД и возврат ответа

MySqlDataReader reader = cmd.ExecuteReader();

// Преобразование ответа в строку и обработка результата

string resultLogin = "";

string resultPassword = "";

while (reader.Read())

{

resultLogin = reader.GetString(0);

resultPassword = reader.GetString(1);

}

myConnection.Close(); // закрытие соединения

// Если в БД не нашлось пользователей с таким логином и паролем, то прерываем авторизацию

if (resultLogin == "" || resultPassword == "")

{

return "401";

}

string token = "";

// Проверка сессии аутентифицированного пользователя

if (!CheckToken(login))

{

// Генерация токена

token = GenerateToken();

// получение ip-адреса клиента

OperationContext context = OperationContext.Current;

MessageProperties prop = context.IncomingMessageProperties;

RemoteEndpointMessageProperty endpoint = prop[RemoteEndpointMessageProperty.Name] as RemoteEndpointMessageProperty;

string ipAddr = endpoint.Address;

// Добавление сессии в БД

cmd.CommandText = string.Format("INSERT INTO sessions (Login, Token, CreateTime, IPAddress) VALUES ('{0}', '{1}', NOW(), '{2}')", login, token, ipAddr);

myConnection.Open();

int result = cmd.ExecuteNonQuery();

myConnection.Close();

}

else

{

cmd.CommandText = string.Format("SELECT \* FROM sessions WHERE sessions.Login = '{0}'", login);

myConnection.Open();

// Отправка запроса к БД и возврат ответа

reader = cmd.ExecuteReader();

// Преобразование ответа в строку и обработка результата

string resultToken = "";

while (reader.Read())

{

resultToken = reader.GetString(1);

}

myConnection.Close();

token = resultToken;

}

// Возврат токена

return token;

}

}

}

1. HTML-код страницы Login.aspx

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Login.aspx.cs" Inherits="HomePage" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title></title>

<style type="text/css">

#Text2 {

width: 186px;

margin-left: 16px;

margin-top: 0px;

}

#Text1 {

width: 186px;

margin-left: 42px;

}

</style>

<script src='https://www.google.com/recaptcha/api.js'></script>

</head>

<body>

<form id="form1" runat="server">

<div>

Login:<asp:TextBox ID="TextBoxLogin" runat="server"></asp:TextBox>

</div>

<div>

Password:<asp:TextBox ID="TextBoxPassword" runat="server"></asp:TextBox>

</div>

<div class="g-recaptcha" data-sitekey="6LcmoCMUAAAAAA7NcvkXnXT7zXpz5862jIJnxHA-"></div>

<div>

<asp:Button ID="ButtonLogin" runat="server" Height="25px" OnClick="ButtonLogin\_Click" Text="login" />

</div>

</form>

</body>

</html>

1. Логика страницы Login.aspx.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Diagnostics;

using ServiceAuthorization;

using System.Security.Cryptography;

using System.Text;

using System.Net;

using Newtonsoft.Json;

public class CaptchaResponse

{

[JsonProperty("success")]

public bool Success { get; set; }

[JsonProperty("error-codes")]

public List<string> ErrorCodes { get; set; }

}

public partial class HomePage : System.Web.UI.Page

{

/// <summary>

/// Загрузка страницы

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void Page\_Load(object sender, EventArgs e)

{

}

/// <summary>

/// Зашифрование RSA

/// </summary>

/// <param name="DataToEncrypt"></param>

/// <param name="RSAKeyInfo"></param>

/// <param name="DoOAEPPadding"></param>

/// <returns></returns>

static public byte[] RSAEncrypt(byte[] DataToEncrypt, RSAParameters RSAKeyInfo, bool DoOAEPPadding)

{

RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();

RSA.ImportParameters(RSAKeyInfo);

return RSA.Encrypt(DataToEncrypt, DoOAEPPadding);

}

/// <summary>

/// Нажатие

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void ButtonLogin\_Click(object sender, EventArgs e)

{

ServiceAuthorizationClient client = new ServiceAuthorizationClient();

var response = Request["g-recaptcha-response"];

string secret = "6LcmoCMUAAAAAK4SMWcFs5OwvXAzZZO3i9yKZA9t";

var clientCaptcha = new WebClient();

var reply = clientCaptcha.DownloadString(string.Format("https://www.google.com/recaptcha/api/siteverify?secret={0}&response={1}", secret, response));

var captchaResponse = JsonConvert.DeserializeObject<CaptchaResponse>(reply);

if (!captchaResponse.Success)

{

var error = captchaResponse.ErrorCodes[0].ToLower();

switch (error)

{

case ("missing-input-secret"):

Response.Write("The secret parameter is missing.");

break;

case ("invalid-input-secret"):

Response.Write("The secret parameter is invalid or malformed.");

break;

case ("missing-input-response"):

Response.Write("The response parameter is missing.");

break;

case ("invalid-input-response"):

Response.Write("The response parameter is invalid or malformed.");

break;

default:

Response.Write("Error occured. Please try again");

break;

}

return;

}

string login = TextBoxLogin.Text;

string password = TextBoxPassword.Text;

// Зашифровывание логина и пароля

UnicodeEncoding byteConverter = new UnicodeEncoding();

RSAParameters publicKey = client.MakeCryptoKey();

byte[] input = byteConverter.GetBytes(login);

byte[] output = RSAEncrypt(input, publicKey, false);

login = Convert.ToBase64String(output);

input = byteConverter.GetBytes(password);

output = RSAEncrypt(input, publicKey, false);

password = Convert.ToBase64String(output);

// Получение токена

string token = client.Authentication(login, password);

if (token == "401")

{

Response.Write("Вы ввели неверные данные");

}

else if (token == "404")

{

Response.Write("Ошибка соединения с сервером");

}

else

{

string casheKey = Guid.NewGuid().ToString();

string[] userInfo = { login, password, token };

Cache.Insert(casheKey, userInfo);

Response.Redirect("Home.aspx?cashe=" + casheKey);

}

}

}

1. HTML-код страницы Home.aspx

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Home.aspx.cs" Inherits="\_Default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title></title>

</head>

<body>

<form id="form1" runat="server">

<p>

<asp:Button ID="ButtonForUser" runat="server" Text="User" OnClick="ButtonForUser\_Click" />

</p>

<p>

<asp:Button ID="ButtonForAdmin" runat="server" Text="Admin" OnClick="ButtonForAdmin\_Click" />

</p>

<p>

<asp:Button ID="ButtonBack" runat="server" Text="Вернуться к авторизации" OnClick="ButtonBack\_Click" />

</p>

</form>

</body>

</html>

1. Логика страницы Home.aspx.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using ServiceAuthorization;

public partial class \_Default : System.Web.UI.Page

{

static string userLogin = "";

static string userPassword = "";

static string userToken = "";

/// <summary>

/// Загрузка страницы

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void Page\_Load(object sender, EventArgs e)

{

string casheKey = Request.QueryString["cashe"];

if (!String.IsNullOrEmpty(casheKey))

{

string[] userInfo = (string[])Cache.Remove(casheKey);

userLogin = userInfo[0];

userPassword = userInfo[1];

userToken = userInfo[2];

Response.Redirect(Request.RawUrl.Replace(Request.Url.Query, ""));

}

if (userToken == "")

{

Response.Redirect("Login.aspx");

}

}

/// <summary>

/// Нажатие на кнопку "User"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void ButtonForUser\_Click(object sender, EventArgs e)

{

ServiceAuthorizationClient client = new ServiceAuthorizationClient();

if (!client.AccessAction(userToken, 1))

Response.Write("У вас нет прав быть пользователем!");

else

Response.Write("Доступ получен, пользователь!");

}

/// <summary>

/// Нажатие на кнопку "Admin"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void ButtonForAdmin\_Click(object sender, EventArgs e)

{

ServiceAuthorizationClient client = new ServiceAuthorizationClient();

if (!client.AccessAction(userToken, 2))

Response.Write("У вас нет прав быть администратором!");

else

Response.Write("Доступ получен, администратор!");

}

/// <summary>

/// Нажатие на кнопку "Вернуться к авторизации"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void ButtonBack\_Click(object sender, EventArgs e)

{

Response.Redirect("Login.aspx");

}

}

1. **Примеры работы программы**

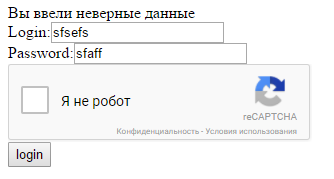


Рисунок 1 – Пример авторизации с ошибкой заполнения полей

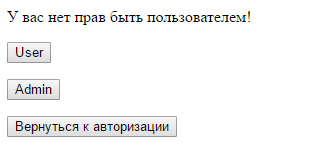
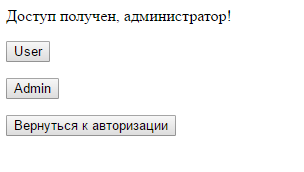


Рисунок 2 – Пример функционала программы (в зависимости от прав доступа при нажатии на кнопку выдаются соответствующие сообщения)

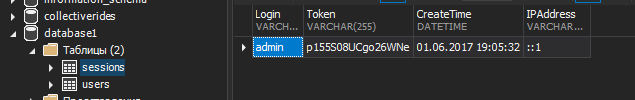


Рисунок 3 – Сессия в базе данных

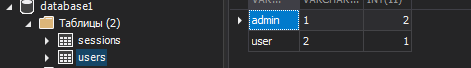


Рисунок 4 – Пользователи в базе данных

1. **Выводы**

В результате проделанной работы были изучены основные методы аутентификации, разобраны различные механизмы аутентификации и авторизации для web-ориентированных приложений, рассмотрены различные модели доступа, особенно модель с токенами, применены меры для защиты от несанкционированного доступа путем шифрования данных методом RSA и привязкой сессии к IP-адресу клиента, применены хеш-функции для обмена данными между различными страницами клиентской части, применены полученные знания на практике и реализовано web-приложение с достаточно безопасной системой аутентификации и авторизации.