Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Курганский государственный университет» (КГУ)

Кафедра «Безопасность информационных и автоматизированных систем»

**РАЗРАБОТКА КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ ДЛЯ** **ПОЛУЧЕНИЯ СПИСКА ИНФОРМАЦИОННЫХ РЕСУРСОВ ПОЛЬЗОВАТЕЛЯ В ОС WINDOWS**

**Курсовая работа**

Дисциплина: Криптографические методы защиты информации

Студент: ИТ–30918 группы…………………………………… /Кандакова А. Н./

подпись

Специальность: 10.05.03 – Информационная безопасность автоматизированных систем

Специализация: Обеспечение информационной безопасности распределенных информационных систем

Руководитель: к.п.н, доцент………………………………./Никифорова Т.А./

подпись

Комиссия: …........................................................................../Никифорова Т. А./

…............................................................................................./Дик Д. И./

Дата защиты: 21.12.2020

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Курган, 2020

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc315364790)

[1 Разработка эскизного и технического проектов клиент-серверного приложения для администрирования ОС WINDOWS 4](#_1.1_Назначение_и)

[1.1 Назначение и область применения клиент-серверного приложения 4](#_Toc315364792)

[1.2 Технические характеристики 4](#_Toc315364793)

[1.2.1 Постановка задачи 5](#_Toc315364794)

[1.2.2 Описание работы приложения 8](#_Toc315364796)

[1.2.3 Организация входных и выходных данных 9](#_Toc315364797)

[1.2.4 Выбор состава технических и программных средств 10](#_Toc315364798)

[2 Разработка рабочего проекта клиент-серверного приложения 12](#_Toc315364800)

[2.1 Конструирование Windows-окна 12](#_Toc315364801)

[2.1.1 Визуальное проектирование приложения 13](#_Toc315364802)

[2.2 Описание кода приложения 15](#_Toc315364803)

[2.3 Спецификация приложения 29](#_Toc315364804)

[2.4 Описание приложения и рекомендации по использованию 30](#_Toc315364805)

[2.5 Тестирование приложения 30](#_Toc315364806)

[Заключение 33](#_Toc315364807)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 34](#_Toc315364808)

[ПРИЛОЖЕНИЕ 35](#_Toc315364809)

# ВВЕДЕНИЕ

Неотъемлемыми частями системного администрирование являются документирование всех произведенных процессов, учёт данных и пользовательских записей. В компаниях с большим штатом сотрудников следить за активностью каждого пользователя, а тем более документирования всех изменений, достаточно проблематично.

Для облегчения работы с пользователями в большинстве случаях используются операционные системы семейства Windows NT. На это есть несколько причин:

1. ОС Windows наиболее распространена (в отличии от Unix-систем), а, следовательно, рядовой пользователь понимает интерфейс.
2. Современные версии Windows не уступают по уровню безопасности Linux [1].

Целью курсовой работы является написание клиент-серверного приложения, которое предоставляло бы возможность получения списка файлов, ключей реестра текущего пользователя Windows. Передача данных должна происходить в закрытом виде: должна использоваться криптографическая функция для шифрования передаваемых данных.

Исходя из цели можно сформулировать следующие задачи курсовой работы:

1. Проанализировать литературу и информационные ресурсы по теме исследования.
2. Изучить структуры реестра и файловой системы ОС Windows.
3. Спроектировать внешний вид клиент-серверного приложения и написать код приложения.
4. Протестировать работу приложения.

Курсовая работа состоит из введения, двух глав, заключения и списка использованных источников (5 источников).

В первой главе описывается область применения клиент-серверного приложения, осуществлена постановка задачи, приведено описание реестра и файловой системы Windows, описаны входные и выходные данные.

Во второй главе рассматривается функции и методы приложения, рекомендации к использованию, описан процесс тестирования.1 **Разработка эскизного и технического проектов Клиент-Серверного приложения для администрирования ОС WINDOWS**

## **1.1 Назначение и область применения клиент-серверного приложения**

Разрабатываемое приложение «Admin» предназначено для получения списка файлов и ключей реестра, доступных для текущего пользователя ОС Windows.

Разрабатываемое приложение может использоваться в сфере информационной безопасности. Клиент-серверное приложение предоставляет возможность удаленной работы, что бывает необходимо при работе с компьютерами, к которым отсутствует прямой физический доступ.

## **1.2 Технические характеристики клиент-серверного приложения**

### *1.2.1 Постановка задачи*

Специалисты информационной безопасностью часто сталкиваются с проблемой администрирования: специалист может не иметь прямого физического доступа компьютеру пользователя. Данную проблему решают программы и приложения удаленного доступа. Приложение удаленного доступа позволяют получит информационные ресурсы пользователя: ключи, данные о дисках, списки файлах.

Следовательно, при разработке такого приложения необходимо разобраться в структурах реестра и файловой системы Windows.

Реестр, по словарю корпорации Майкрософт [2], представляет собой иерархическую центральную базу данных используемы во многих версиях операционной системы Windows. База хранит в себе необходимую информацию о настройках ОС, приложений. Реестр содержит сведения о автозагрузки, параметрах и свойствах папок, приложений, профили пользователей. Так же существует следующее понятие – куст реестра. Куст реестра – группа разделов, подразделов, значений в реестре. Кроме того, куст содержит вспомогательные файлы с резервной копией информации.

*Таблица 1*

*Перечень кустов реестра*

|  |  |
| --- | --- |
| Куст реестра | Вспомогательные файлы |
| HKEY\_LOCAL\_MACHINE\SAM | SAM, SAM. log, SAM. САВ |
| HKEY\_LOCAL\_MACHINE\Security | Безопасность, Security. log, Security. САВ |
| HKEY\_LOCAL\_MACHINE\Software | Программное обеспечение, Software. log, Software. САВ |
| HKEY\_LOCAL\_MACHINE\System | System, System. Alt, System. log, System. САВ |
| HKEY\_CURRENT\_CONFIG | System, System. Alt, System. log, System. САВ, NTuser. dat, NTuser. dat. Log |
| HKEY\_USERS\DEFAULT | По умолчанию, Default. log, Default. САВ |

В целях безопасности управлять доступом к реестру Windows разрешено учетной записи с правами администратора. В следующей таблице (Таблица 2) перечислены стандартные ключи, используемые операционной системой.

*Таблица 2*

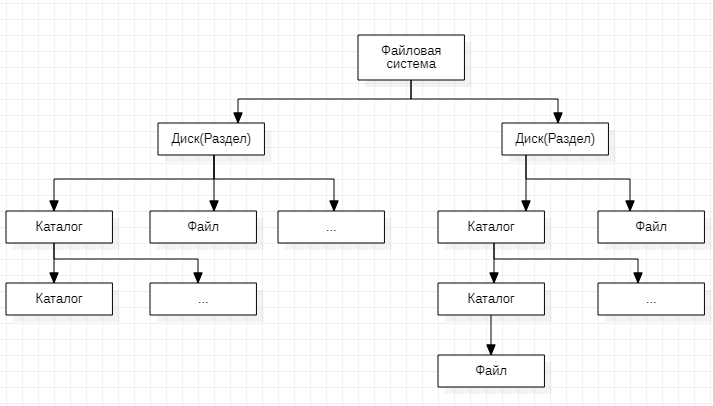
*Стандартные ключи*

|  |  |
| --- | --- |
| Папка/предварительно определенный раздел | Описание |
| HKEY\_CURRENT\_USER | Здесь хранятся папки пользователя, цвета экрана и параметры панели управления. |

*ПРОДОЛЖЕНИЕ Таблицы 2*

|  |  |
| --- | --- |
| HKEY\_USERS | Содержит все активные загруженные профили пользователей на компьютере. |
| HKEY\_LOCAL\_MACHINE | Содержит сведения о конфигурации, определенные для компьютера. |
| HKEY\_CLASSES\_ROOT | Хранящиеся здесь сведения позволят убедиться в том, что при открытии файла с помощью проводника Windows откроется правильная программа. |
| HKEY\_CURRENT\_CONFIG | Содержит сведения о профиле оборудования, используемом локальным компьютером при запуске системы. |

Рассмотрим структуру файловой системы Windows. Система представляет собой древовидную структуру директорий, облегающих поиск и хранение информации. Внутри одной директории может храниться несколько других каталогов. Группировка элементов зависит от пользователя.



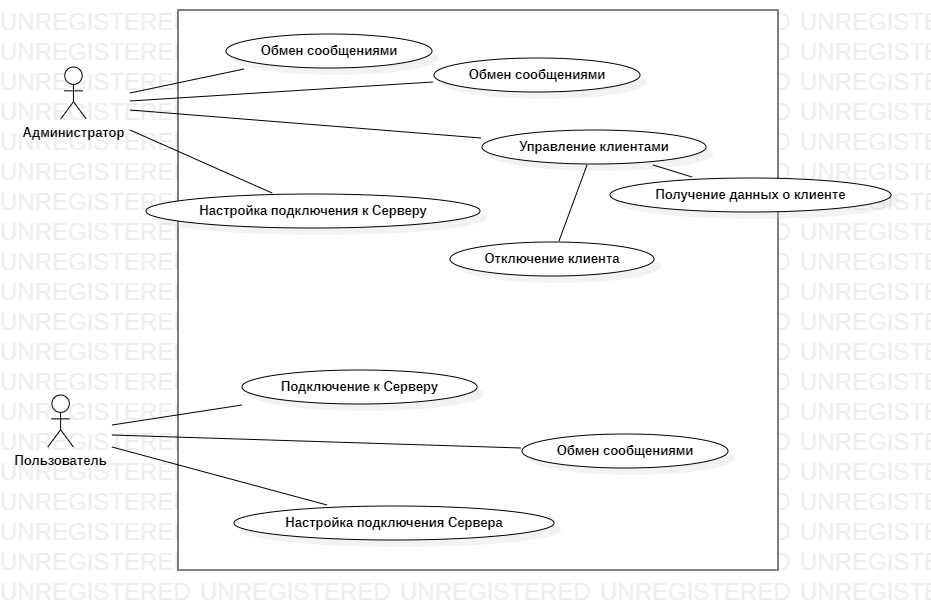
*Рисунок 1 –Структура файловой системы*

Для более простого хранения файлов принято разделять физический диск, как правило HDD накопитель, на логические разделы. Обычно первый логический раздел используется для ОС, второй – хранение документов, программ.

### *1.2.2 Описание работы приложения*

При запуске сервер-приложения будет открыто главное окно с настройками: ip-адрес, порт и ключ доступа. Ключ доступа является не обязательным параметром. Сервер, как и клиент включает все себя аналог чата, который может сообщать об ошибках операций. Будет доступен список подключенных клиентов, из которого администратор – пользователь сервера.

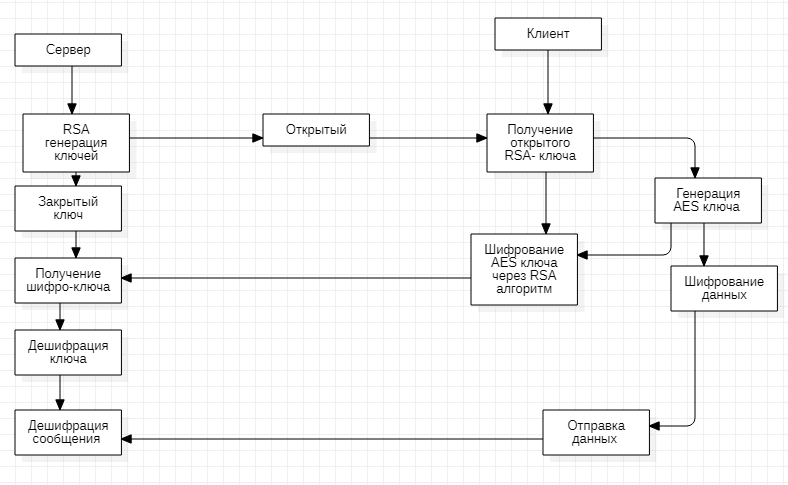
Клиент-приложение включает в свою очередь чат и настройку подключения: IP-адрес, порт, ключ доступа. Имя клиента определяется учетной записью пользователя ОС Windows.



*Рисунок 2 – Варианты использования приложения*

Варианты использования клиент-приложения и сервер- приложения предоставлены на указанной выше диаграмме (Рисунок 2), ней описаны основные действия, которые могут совершать пользователи клиент-серверного приложения.

Для защиты данных используется криптографическая функция, основанная на RSA и AES шифровании (Рисунок 3). При запуске сервера будут RSA ключи. Закрытый ключ, в целях безопасности, будет храниться на сервере, открытый будет отправлен клиенту при запросе сервера на получение данных.



*Рисунок 3 – Криптографическая функция*

Клиент должен получать запрос и RSA ключ. Открытый ключ используется для шифрования сгенерированного AES-ключа, который используется при шифровании данных через AES – алгоритм. Далее происходит отправка шифрованных ключа и данных на сервер. Сервер при получении ключа, будет дешифровать его с помощью закрытого RSA-ключа, тем самым получая AES-ключ для дешифрации данных.

### *1.2.3 Организация входных и выходных данных*

Входными данными для сервера – строка “<GetDate>”, содержащая параметры запроса:

* «-D» -диски
* «-I» - файлы
* «-R» - ключи

Пример отправляемого запроса с сервера на клиент:  
 ”<GetDate –D –I –R>”. Для шифрования данных предусмотрены входные данные для сервера – RSA-ключи в строковом формате: pubKeyString, как открытый ключ, и privKeyString. Открытый ключ отправляется на клиента перед запросом “<GetDate>” виде строки. Клиент в свою очередь имеет входные данные AES-ключ, который позже шифруются открытым RSA-ключом и отправляется на сервер.

В результате выполнения операция с ключами и получения клиент отправляет шифрованные с параметром “<GetDateResult>”, который считывается и дешифруются сервером в несколько файлов формата “.txt” в зависимости от количества полученных пакетов.

### *1.2.4 Выбор состава технических и программных средств*

Средой разработки приложения была выбрана интегрированная среда разработки ПО для Microsoft Windows – Visual Studio 2019. Данная среда разработки поддерживает множество языков, в том числе C# (си шарп) и C++.

Выбор данной среды разработки обусловлен следующими причинами:

1. Поддержка SQL Server и SQL Client и других баз данных– более удобное и быстрое редактирование баз данных.
2. Визуальное построение приложений из программных прототипов, что значительно ускоряет разработку.
3. Наличие стандартных - для Windows Form, и нестандартных - для WPF (Windows Presentation Foundation), компонентов окна.
4. Полная поддержка таких кодировок, как Unicode и ANSI. Применение этих кодировок гарантирует, что приложения будут одинаково выглядеть и функционировать во всех языковых версиях Windows.

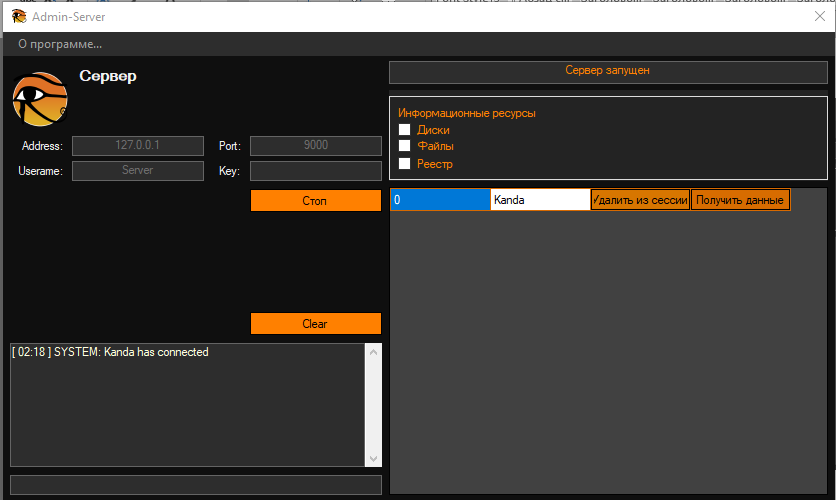
В качестве языка выбран C#, достаточно тонкая настройка интерфейса программы. Использование языка интегрированных LINQ даёт возможность писать структурированные безопасные в отношении типов запросы к локальным коллекциям объектов и удаленным данных.

Для написания криптографических функции использовалось пространство имен System.Security.Cryptography – криптографические службы, которые включают в себя безопасное кодирование и декодирование данных, хеширование и многие другие операции.

# 2 Разработка рабочего проекта КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ «ADMIN»

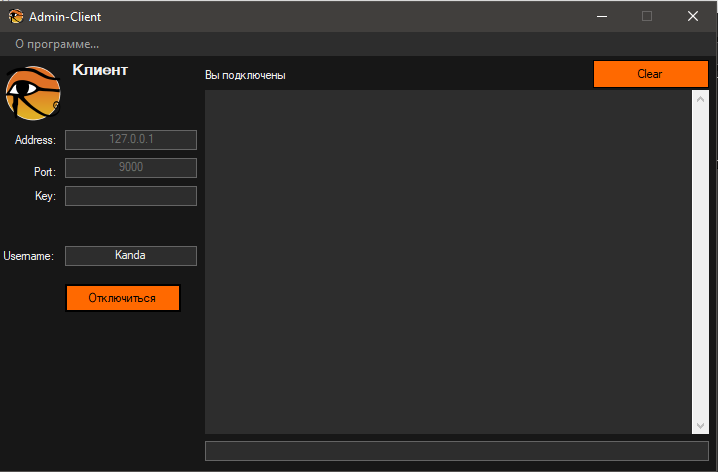
## **2.1 Конструирование Windows-окна**

Интерфейс приложения должен быть наиболее удобным и простым в использовании (Рисунок 4).



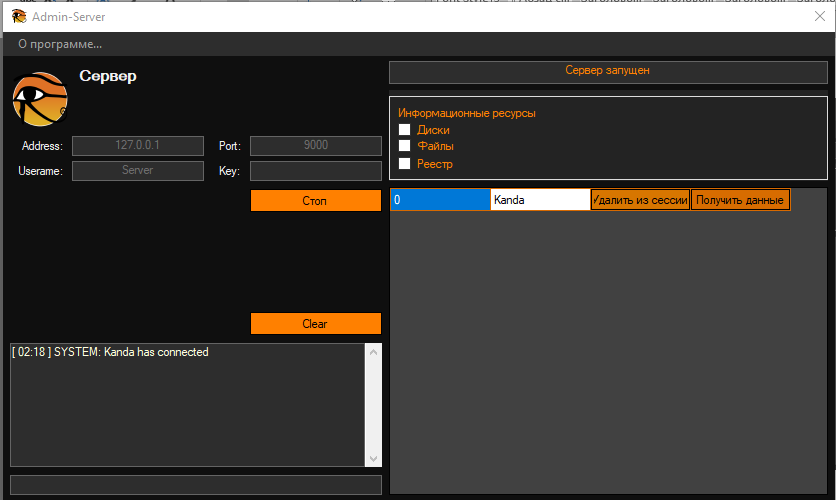
*Рисунок 4 - Внешний вид работающего приложения (Сервер)*

Форма приложение-сервера для удобства имеет выбор параметров для получения информационных ресурсов: «Диски», «Файлы», «Реестр». Приложение сообщит в чат об ошибках подключение или ошибках при получении данных клиента в чат (Рисунок 5).



*Рисунок 5 - Внешний вид работающего приложения (Клиент)*

### 2.1.1 Визуальное проектирование приложения



sendTextBox

clientsDataGridView

DC и GetInfo

statusBox

startButton

*Рисунок 6 - Визуальные и невизуальные компоненты главной формы приложения (Сервер)*

*Таблица 3*

*Назначение компонентов главной формы приложения (Сервер)*

| ***№ п/п*** | ***Компонент*** | ***Назначение*** |
| --- | --- | --- |
| 1 | AdminServerForm | Главная форма приложения |
| 2 | startButton | Запуск/Остановка Сервера |
| 3 | clearButton | Очистка области чата |
| 4 | clientsDataGridView | Таблица подключений |
| 5 | DC | Кнопка отключение текущего пользователя |
| 6 | GetInfo | Получить информацию о клиенте |
| 7 | logTextBox | Чат |
| 8 | sendTextBox | Ввод сообщений |
| 9 | addrTextBox | Ввод IP -адреса |
| 10 | usernameTextBox | Имя сервера (используется только в чате) |
| 11 | portTextBox | Порт |
| 12 | keyTextBox | Ключ доступа к серверу |
| 13 | statusBox | Состояние сервера |

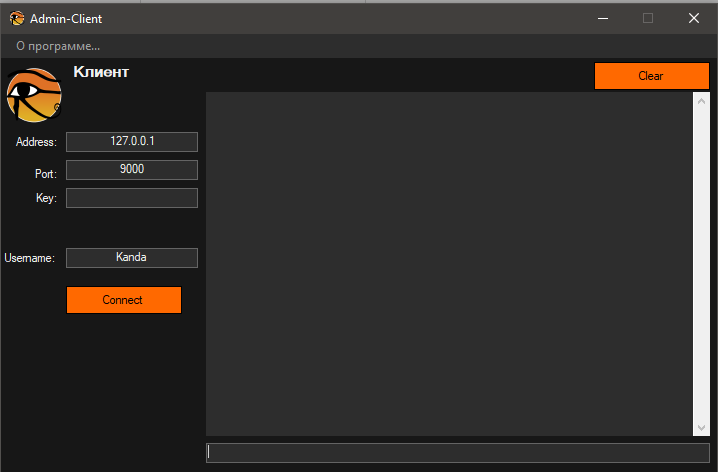
*ПРОДОЛЖЕНИЕ Таблицы 3*

|  |  |  |
| --- | --- | --- |
| 14 | Disks | ComboBox. Выбор информации о дисках |
| 15 | files | ComboBox. Выбор информации о файлах |
| 16 | regedit | ComboBox. Выбор информации о реестре |
| 17 | logTextBox | Чат |
| 18 | sendTextBox | Ввод сообщений |

*Таблица 4*

*Изменённые свойства компонентов главной формы приложения (Сервер)*

| *№ п/п* | *Компонент* | *Свойства* |
| --- | --- | --- |
| 1 | startButton | Text = “Start” |
| 2 | statusBox | Text = “Сервер запущен” |



*Рисунок 7 Визуальные и невизуальные компоненты главной формы приложения (Клиент)*

Ниже описаны основные компоненты (Таблица 5) и функции  
 (Таблица 6) главной формы клиент-приложения.

*Таблицы 5*

*Назначение компонентов главной формы приложения (Клиент)*

| ***№ п/п*** | ***Компонент*** | ***Назначение*** |
| --- | --- | --- |
| 1 | usernameTextBox | Имя сервера (используется только в чате) |

*ПРОДОЛЖЕНИЕ Таблицы 5*

|  |  |  |
| --- | --- | --- |
| 2 | portTextBox | Порт |
| 3 | keyTextBox | Ключ доступа к серверу |
| 4 | connectButton | Кнопка подключения к серверу |
| 5 | statuslabel | Статус клиента |
| 6 | logTextBox | Чат |
| 7 | sendTextBox | Ввод сообщений |
| 8 | clearButton | Очистка области чата |
| 9 | addrTextBox | Ввод IP –адреса сервера |
| 10 | usernameTextBox | Имя пользователя |
| 11 | portTextBox | Порт |
| 12 | keyTextBox | Ключ доступа к серверу |

*Таблица 6*

*Изменённые свойства компонентов главной формы приложения (Клиент)*

| *№ п/п* | *Компонент* | *Свойства* |
| --- | --- | --- |
| 1 | connectButton | Text= “Connect” |
| 2 | statuslabel | Text=”Вы подключены” |

## **2.2** **Описание кода приложения**

При работе программы использованы следующие функции:

*Таблица 7*

*Используемые процедуры и функции в главной форме приложения (Сервер)*

| *Функция* | *Назначение* |
| --- | --- |
| Log(string msg = "") | Используется для вывода в logTextBox ошибок, системные сообщений, сообщений с клиентов. |

*ПРОДОЛЖЕНИЕ Таблицы 7*

| ErrorMsg(string msg) | Конструктор сообщений об ошибке |
| --- | --- |
| SystemMsg(string msg) | Конструктор системных сообщений. |
| Active(bool status) | Отвечает за запуск и остановку сервера. |
| AddToGrid(long id, string name) | Добавляет подключенного клиента в список clientsDataGridView. |
| RemoveFromGrid(long id) | Удаляет выбранного клиента из списка clientsDataGridView. |
| ClientsDataGridView\_CellClick | Обработка и проверка двойных нажатий мышкой в ячейку. |
| ParseMessage(string message) | Проверка сообщения на команду. |
| Read(IAsyncResult result) | Чтение исходящих от клиентов данных. |
| ReadAuth(IAsyncResult result) | Авторизация клиента, читает данные о клиенте. Вызывается из Authorize |
| Authorize(MyClient obj) | Основная функция авторизации, вызывается из Connection. |
| StartButton\_Click | Обработка события StartButton. |
| ShowStatus(string status) | Устанавливает статус сервера |
| ExitToolStripMenuItem\_Click | Выход из программы, при помощи контекстного меню. |
| Connection(MyClient obj) | Главная функция соединения клиентов с сервером. Также добавляет или удаляет клиентов, если у тех произошёл сбой. |
| ShowStatus(string status) | Устанавливает статус сервера |

*ПРОДОЛЖЕНИЕ Таблицы 7*

| Connection(MyClient obj) | Главная функция соединения клиентов с сервером. Также добавляет или удаляет клиентов, если у тех произошёл сбой. |
| --- | --- |
| Listener(IPAddress ip, int port) | Проверка портов. В случает отсутствии Интернета выдаёт ошибку. |
| BeginWrite(string msg, MyClient obj) | Отправка сообщения определенному клиенту. |
| BeginWrite(string msg, long id = -1) | Отправка сообщения всем клиентам. |
| Write(IAsyncResult result) | Функция отправки сообщения используется в методах BeginWrite. |
| Send(string msg, MyClient obj) | Асинхронный метод отправки конкретному пользователю. Вызывает функции BeginWrite. |
| SendTextBox\_KeyDown | Обработка ввода. Если клавиша “Enter”, то отправляет сообщения |
| Disconnect(long id = -1) | Завершение всех потоков. Вызывается при закрытии или остановки сервера. |
| GetInformation(long id) | Создание команды по получению информационных ресурсов. Отправка её клиенту. |

Многие функции клиента повторяют функции сервера. Помимо этого, клиентское приложение содержит класс для работы с файловой системой.

*Таблица 8*

*Используемые процедуры и функции в главной форме приложения (Клиент)*

| *Функция* | *Назначение* |
| --- | --- |
| Log(string msg = "") | Используется для вывода в logTextBox данных: ошибки, системные сообщения, сообщения с клиентов. |
| Active(bool status) | Отвечает за запуск и остановку сервера. |
| AdminClientForm\_Load | Используется для автоматического определения имени учетной записи |
| ParseMessage(string message) | Проверка сообщения на команду. |
| Read(IAsyncResult result) | Чтение исходящих от сервера данных. |
| ReadAuth(IAsyncResult result) | Авторизация клиента, читает данные о клиенте. Вызывается из Authorize. |
| Authorize(MyClient obj) | Основная функция авторизации, вызывается из Connection. |
| Connection(MyClient obj) | Главная функция соединения клиентов с сервером. |
| EncryptStringToBytes\_Aes(string plainText, byte[] Key, byte[] IV) | Шифрование данных через AES. |
| EncryptKey(byte[] keyAes) | Шифрование AES-ключа |
| AESEncrypt(string temp, byte[] keyAes, byte[] IVaes) | Вызывает функцию шифрования данных EncryptStringToBytes\_Aes. |
| ConnectButton\_Click | Отвечает за соединение с сервером. |

*Таблица 9*

*Класс GetDate (Клиент)*

| *Функция* | *Назначение* |
| --- | --- |
| ParseCommand(string command) | Обработка команды, вызов функций работы с информационными ресурсами по параметрам команды. |
| GetKeys() | Получить ключи реестра |
| GetDisk() | Получить информацию о текущих дисках. |
| GetInformation() | Информация о файлах пользователя (выдаёт список). |
| SearchFile(string patch) | Поиск файлов по пути. |

Приведём описание некоторых из вышеописанных методов главной формы приложения «Admin».

Рассмотрим главные и основные функции приложения: функции запроса и получения данных и реализация криптографических методов.

При запуске сервера генерируются RSA-ключи: метод StartButton\_Click вызывает функцию GenerateRSAKeys() перед запуском прослушивания портов с помощью Listener.

private void GenerateRSAKeys()

{

RSA rsa = RSA.Create();

// Получаем из полученного экземпляр RSA открытый ключ

pubKeyString = rsa.ToXmlString(false);

// и закрытый

privKeyString = rsa.ToXmlString(true);

}

Далее сервер ожидает подключение клиентов и действий пользователя.

Пользователь сервера создаёт запрос с выбранными данными из ComboBox (disks, files, regedit), при этом выбрав клиента из списка.

Вызывается событие ClientsDataGridView\_CellClick, которое проверяет какое действие с клиентом выбрал пользователь. Так как пользователь интересует запрос по данным, в событии вызывается функция GetInformation(long id). На вход она получает идентификационный номер выбранного клиента. Так же данная функция отправляет открытый ключ клиенту, позже отправляет сам запрос.

private void GetInformation(long id)

{

clients.TryGetValue(id, out MyClient obj);

// Проверяем, что выбрал пользователь.

//Формируем запрос на основе выбора

if (disks.Checked || files.Checked || regedit.Checked)

{

// Отправляем открытый RSA-ключ клиенту

Send(pubKeyString, obj);

Thread.Sleep(500);

string command = "<GetDate ";

if (disks.Checked)

{

command += "-D ";

}

if (files.Checked)

command += "-I ";

if (regedit.Checked)

command += "-R";

command += "> ";

// Отправляем запрос

Send(command, obj);

}

else

MessageBox.Show("Выберите тип ресурсов");

}

Клиент c помощью функции Read(IAsyncResult result) может получать данные со сервера и других клиентов. При это м каждое сообщение обрабатывается функцией ParseMessage. Данная функция рассматривает сообщения на содержание запросов, ключей, иначе данное сообщение является частью чата.

public bool ParseMessage(string message)

{

if (message.Contains("<GetDate"))

{

GetDate j = new GetDate();

// Собираем информацию по запросу

j.ParseCommand(message);

List<string> report = j.information;

// Начинаем шифрование AES

Aes aes = Aes.Create();

//Отправка ключа (зашифрован)

Send(String.Format("<GK>{0}", EncryptKey((aes.Key))));

Thread.Sleep(1000);

//Отправка вектора AES (зашифрован)

Send(String.Format("<IV>{0}", EncryptKey(aes.IV)));

Thread.Sleep(3000);

foreach (string g in report)

{

// Шифруем и отправляем данные серверу

Send(String.Format("<GetDateResult>{0}", AESEncrypt(g, aes.Key, aes.IV)));

Thread.Sleep(1000);

}

return true;

}

else if (message.Contains("RSAKeyValue"))

{

pubKeyString = message;

return true;

}

else

return false;

}

Обратим внимание на шифрование ключей и вектора инициализации – оба они шифруются открытым RSA-ключом. Функция EncryptKey – использует открытый ключ для шифрования входного массива, в результате получаем строку.

public string EncryptKey(byte[] keyAes)

{

var rsa = new RSACryptoServiceProvider(1024);

rsa.FromXmlString(pubKeyString);

var encryptedData = rsa.Encrypt(keyAes, true);

var base64Encrypted = Convert.ToBase64String(encryptedData);

return base64Encrypted;

}

Рассмотри основные функции шифрования дешифрования данных о клиенте.

AESEncrypt – получает не зашифрованный ключе AES, вектор инициализации, строку данных, которую необходимо зашифровать. Однако она является промежуточным конвертером из байтов в строку. Функция вызывает метод шифрования - EncryptStringToBytes\_Aes.

public string AESEncrypt(string temp, byte[] keyAes, byte[] IVaes)

{

if (temp == null || temp.Length <= 0 || IVaes == null || IVaes.Length <= 0 || keyAes == null || keyAes.Length <= 0)

throw new ArgumentNullException("Checkinputdate");

byte[] encrypted = EncryptAesData(temp, keyAes, IVaes);

return Convert.ToBase64String(encrypted);

}

private static byte[] EncryptAesData(string currentData, byte[] Key, byte[] IV)

{

byte[] byteEncrypt;

using (Aes aes = Aes.Create())

{

aes.Key = Key;

aes.IV = IV;

ICryptoTransform encryptor = aes.CreateEncryptor(aes.Key, aes.IV);

using (MemoryStream ms = new MemoryStream())

{

using (CryptoStream cs = new CryptoStream(ms, encryptor, CryptoStreamMode.Write))

{

using (StreamWriter sw = new StreamWriter(cs))

{

sw.Write(currentData);

}

byteEncrypt = ms.ToArray();}}}

return byteEncrypt;}

Как только данные были зашифрованы они отправляются на сервер. Сервер получил шифрованные ключ и вектор. Так же, как и на клиенте сервер обрабатывает каждое входное сообщение с помощью ParseMessage(string message, long id). При этом функция вызывает методы дешифрования ключа или данных по клиенту.

public bool ParseMessage(string message, long id)

{

clients.TryGetValue(id, out MyClient obj);

// Проверка на содержание ответа на запрос.

if (message.Contains("<GetDateResult>"))

{

//Дешифруем сообщение

string tmp = DecryptAESData(Convert.FromBase64String(message.Replace("<GetDateResult>", "").ToString()), keyDecrypt, IV);

string name = "Temp-" + DateTime.Now.ToString().Replace(":", "-").Replace(".", "-");

using (FileStream fstream = new FileStream($"{name}.txt", FileMode.OpenOrCreate))

{

byte[] array = System.Text.Encoding.Default.GetBytes(tmp);

fstream.Write(array, 0, array.Length);

}

string msg = string.Format("Данные получены");

Log(msg);

Send(msg, obj.id);

return true;

}

else if (message.Contains("<GK>"))

{

// Дешифруем ключ AES

keyDecrypt = DecryptAES(message.Replace("<GK>", ""));

return true;

}

else if (message.Contains("<IV>"))

{

// Дешифруем вектор AES

IV = DecryptAES((message.Replace("<IV>", "")));

return true;

}

else

{

return false;

}

}

Функция DecryptAES дешифрует ключ или вектор с помощью RSA алгоритма. В результате её мы получим необходимые данные для дешифрования основного ответа на запрос – списки информационных ресурсов клиента.

private byte[] DecryptAES(string Aeskey)

{

var rsa = new RSACryptoServiceProvider(1024);

rsa.FromXmlString(privKeyString);

var resultBytes = Convert.FromBase64String(Aeskey);

var decryptedBytes = rsa.Decrypt(resultBytes, true);

return decryptedBytes;

}

Дешифрованием основных данных занимается DecryptAESData. После дешефрации у нас имеются необходимые переменные для дешифрования: ключ и вектор.

tatic string DecryptAESData(byte[] cipherText, byte[] Key, byte[] IV)

{

if (cipherText == null || cipherText.Length <= 0 || IV == null || IV.Length <= 0 || Key == null || Key.Length <= 0)

throw new ArgumentNullException("Checkinputdate");

string plaintext = null;

using (Aes aesAlg = Aes.Create())

{

aesAlg.Key = Key;

aesAlg.IV = IV;

ICryptoTransform decryptor = aesAlg.CreateDecryptor(aesAlg.Key, aesAlg.IV);

using (MemoryStream ms = new MemoryStream(cipherText))

{

using (CryptoStream cs = new CryptoStream(ms, decryptor, CryptoStreamMode.Read))

{

using (StreamReader sr = new StreamReader(csDecrypt))

{

plaintext = sr.ReadToEnd();}}}}

return plaintext;}

После дешифрования полученные данные сохраняются в файл.

Теперь рассмотрим функции для получения информационных ресурсов клиента. При обработке ParseMessage в клиенте создаётся экземпляр класса GetDate, отвечающий за работу с информационными ресурсами клиента: получение ключей (GetKeys), данных о дисках (GetDisk), списков файлов(GetInformation).

*Примечание: Пришлось искусственно ограничить получение ключей – большое количество ключей значительно влияло на время обработки данных, вместо это мы получаем данные определённого куста реестра.*

public void GetKeys()

{

string ListKeys = String.Format("<GetDateResult> {0} ", Environment.UserName);

const string REGISTRY\_ROOT = @"SOFTWARE\";

using (RegistryKey rootKey = Registry.CurrentUser.OpenSubKey(REGISTRY\_ROOT, true))

{

if (rootKey != null){

string[] valueNames = rootKey.GetSubKeyNames();

StringBuilder sb = new StringBuilder(valueNames.Length);

foreach (string ch in valueNames)

{

sb.Append("\n");

sb.Append(ch);

}

ListKeys += sb.ToString();

}

rootKey.Close();

}

information.Add(ListKeys);

}

Получение информации о дисках текущего ПК.

public void GetDisk()

{

StringBuilder driveList = new StringBuilder();

foreach (DriveInfo d in DriveInfo.GetDrives())

{

if (d.IsReady) driveList.AppendLine(String.Format("Диск: {0}; метка тома: {1}; файловая система: {2}; тип: {3}; объем: {4} байт; свободно: {5} байт", d.Name, d.VolumeLabel, d.DriveFormat, d.DriveType, d.TotalSize, d.AvailableFreeSpace));

}

string temp = String.Format("<GetDateResult> {0} {1}",Environment.UserName, driveList.ToString());

information.Add(temp);

}

Получение информации о файлах и каталогов каждого диска на ПК. Исключение try-catch получает доступные для пользователя файлы и каталоги.

public void GetInformation()

{

string ListPatch = String.Format("{0} ", Environment.UserName);

try { // некоторые устройства Windows воспринимает как диски

foreach (DriveInfo d in DriveInfo.GetDrives())

{

foreach (FileInfo rootDirFile in d.RootDirectory.GetFiles("\*"))

{

ListPatch += rootDirFile.Name + "\n";

}

string[] S = SearchDirectory(d.Name);

foreach (string folderPatch in S)

{

try

{

string[] F = SearchFile(folderPatch);

foreach (string FF in F)

{

ListPatch += FF + "\n";

}

}

catch{}}}}

catch { }

information.Add(ListPatch);

}

Функции для рекурсивного поиска файлов, каталогов.

static string[] SearchFile(string patch)

{

string[] ReultSearch = Directory.GetFiles(patch, "\*", SearchOption.AllDirectories);

return ReultSearch;

}

static string[] SearchDirectory(string patch){

string[] ReultSearch = Directory.GetDirectories(patch);

return ReultSearch;

}

## **2.3 Спецификация приложения**

Состав проекта:

|  |  |
| --- | --- |
| *Наименование* | *Обозначение* |
| AdminServer.exe | Исполняемый файл приложения (Windows-приложение). Сервер-приложение. |
| Admin.exe | Исполняемый файл приложения (Windows-приложение). Клиент-приложение. |

## **2.4 Описание приложения и рекомендации по использованию приложения**

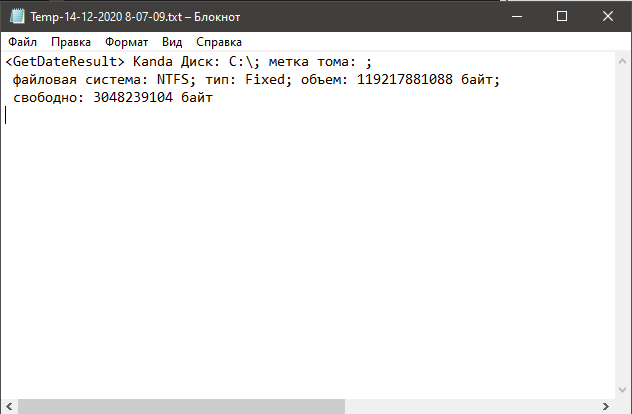
Данное приложение просто в использовании. Для запуска достаточно запустить файлы «AdminServer.exe» на компьютере-сервере и «AdminServer.exe» на компьютере-клиенте.

На приложении-сервер нужно ввести необходимые настройки: ip-адрес, порт, ключ. Затем запустить сервер кнопкой «Start». Приложение-клиент: вводится данные о сервере, затем кнопкой «Connect» выполнить подключение.

Для получение сервером данных о конкретном клиенте нужно указать какие данные мы хотим получить: «диски», «файлы», «ключи». Далее в списке нажать на кнопку «Получить данные». В результате выполнения на сервер придёт сообщение об выполнении запроса, после чего в папке с приложением-сервер будут лежать несколько файлов (количество зависит от количества выбранных параметров).

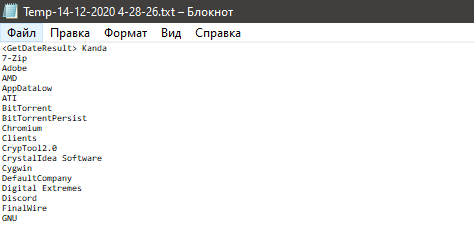
## **2.5 Тестирование приложения**

Приложение прошло тестирование в несколько этапов и было выявлено несколько уязвимостей, связанных с безопасностью. Приложение выполняет функции: получение информационных ресурсов с клиента. Ниже приведены примеры работы:



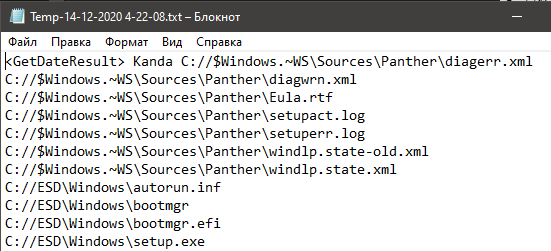
*Рисунок 8 - Информация о дисках*

Проверка получение данных о дисках прошла успешно: получаем размер дисков, тип файловой системы (Рисунок 8).



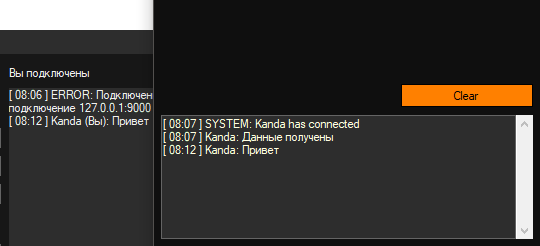
*Рисунок 9 - Ключи*

Функция поиска ключей в реестре выдаёт список определённых ключей – ключи программ (Рисунок 9). Так же работает функция на получения файлов с дисков клиента (Рисунок 10).



*Рисунок 10 - Результат подключения*

Была также проверена работоспособность чата (Рисунок 11). Видны сообщения с клиента, системные сообщения.



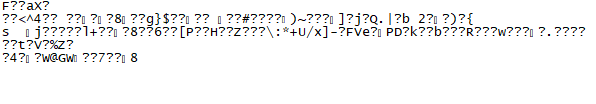
*Рисунок 11 - Проверка работы чата*

При тестировании приложения были выделены следующие уязвимости:

* скорость отправки и создание конечных файлов при большом объеме данных клиента – чем больше информации у клиентов, тем больше создаётся файлов;
* конфиденциальность сервера и клиента – данные связанные с IP –адресом, портом и ключом никак не скрываются.

Способы решения проблем: добавление свойств или элементов для скрытия данных, при отправке команды можно использовать сторонние порты. Также для защиты от добавления зловредного кода следует пресечь попытки с помощью обфускации кода.

Для проверки криптографической функции в отладке программы отключалась дешифрация конечных результатов – это позволило понять работает ли шифрование на клиенте. Тест на проверку шифровании дал положительный результат (Рисунок 11).



*Рисунок 11 – Результат шифрования*

Так как шифрование сработало, был приведён дополнительный тест на шифрование и дешифрование. Программа успешно прошла тест.

# **Заключение**

Безопасная удаленная работа с информационными ресурсами на данный момент является актуальной проблемой системного администрирования и информационной безопасности. Не многие приложения удалённого доступа выполняют свою роль идеально.

В ходе курсовой работы было разработано клиент-серверное приложение для администрирования ОС Windows. Приложение выполняет функции получение данных о дисках и файлах. Однако функция по получение ключей работает на определенном кусте реестра, содержащий сведения и настройки программного обеспечения. При этом большое количество информационных ресурсов клиента может повлиять на передачу данных – часть данных может не отображаться. Цель курсовой была частично достигнута.

При написании курсовой работы были выполнены следующие задачи:

1. Проанализирована литературу по теме исследования.
2. Выявлены основные положительные и отрицательные черты Windows OS.
3. Спроектирован внешний вид приложения и написан код приложения.
4. Протестирована работу приложения.

Поставленные задачи выполнены.

Данное приложение имеет возможность развития: добавление дополнительных функций для системного администрирования.

Практическая значимость данного приложения – возможность получать данные, к примеру, компьютеров офиса удаленно, что может улучшить производительность труда.

# **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Habr.com – блог о технологиях: «Веб-хостинг: Windows или Linux?»: офиц. сайт – URL <https://habr.com/ru/company/ruvds/blog/322650/> (дата обращения: 11.01.2020)
2. Домашняя страница документации и учебных ресурсов Майкрософт для разработчиков и технических специалистов: Сведения о реестре Windows для опытных пользователей: офиц. сайт – URL https://docs.microsoft.com/ru-ru/troubleshoot/windows-server/performance/windows-registry-advanced-users  
   (дата обращения: 5.10.2020)
3. Официальная Документация по Windows 10: офиц. сайт – URL https://docs.microsoft.com/ru-ru/windows/windows-10/ (дата обращения: 9.01.2020)
4. Юрий Белов. Администрирование серверных операционных систем семейства Windows. // Издательство МГТУ им Н.Э. Баумана 2014. - 325 с.   
   (дата обращения 17.01.2020)
5. Петр Шетка. Microsoft Windows server 2003. Практическое руководство по настройки сети // СПб.: Наука и Техника. 2006. – 608 с. (дата обращения: 24.10.2020)

# **ПРИЛОЖЕНИЕ**

Структура диска

Диск содержит 2 проекта Visual Studio: Admin (Клиент-приложение) и Admin-Server (Сервер-приложение). Решение запускается через sln – файл в Admin. Также на диске содержится документ курсовой, презентация и изображения с результатами проверки на antiplagiat.ru.