**Федеральное агентство связи**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение**

**Высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра Информационная безопасность

**Отчет по проекту**

По дисциплине «Защита информации от вредоносного программного обеспечения»

Выполнил: студент группы БВТ1801

Митрохин Ярослав Игоревич

Проверил:

Барков Вячеслав Валерьевич

Москва 2021

*Результаты выполнения начального этапа работы:*

Цель работы состоит в написании клиент-серверного приложения с возможностью передачи данных между собой. В качестве программного интерфейса для обмена данными между процессами были выбраны сокеты.

В основе межсетевых взаимодействий по протоколам TCP и UDP лежат сокеты. В .NET сокеты представлены классом System.NET.Sockets.Socket, который предоставляет низкоуровневый интерфейс для приема и отправки сообщений по сети. При работе с сокетами вне зависимости от выбранных протоколов мы будем опираться на методы класса Socket:

* Accept(): создает новый объект Socket для обработки входящего подключения;
* Connect(): устанавливает соединение с удаленным хостом;
* Listen(): начинает прослушивание входящих запросов;
* Flush(): удаляет данные из потока;
* Close(): закрывает сокет.

Далее представлен разработанный интерфейс для сервера и клиента и краткая информация по работе данных приложений.

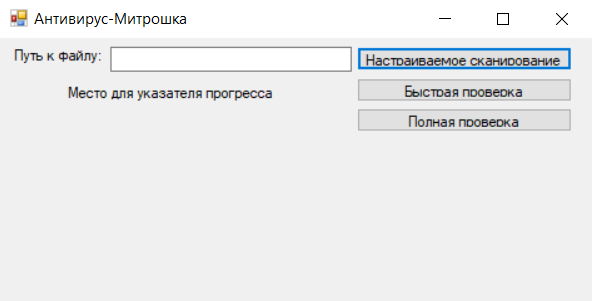


Рисунок 1 - Интерфейс приложения AntiClient

На данном этапе разработки антивируса в качестве действующей кнопки для проверки и диагностики работоспособности программы была выбрана кнопка «Быстрая проверка».

При нажатии кнопки «Быстрая проверка» происходит вызов метода Listeners(), который выполняет функции соединения с сервером, передачи данных и проверки правильности передачи данных. Также в качестве основы реализаций методов необходимых непосредственно для работы антивирусной программы был сделан простой вывод предполагаемых действий в текстовом формате. В данный момент передача данных между двумя процессами реализована в простом выводе текстовой информации на клиенте.

Также на клиенте реализован вывод передаваемой информации для виртуализации работы программы. Для будущей реализации антивирусной программы был выбран следующий вариант ввода информации:

«Путь каталога Путь файла Код выполняемого действия»

В дальнейшем вариант ввода информации может измениться.

В качестве способа подключения сервера к клиенту был выбран класс TcpListener, который позволяет ожидать подключения от TCP-клиентов сети по номеру порта.

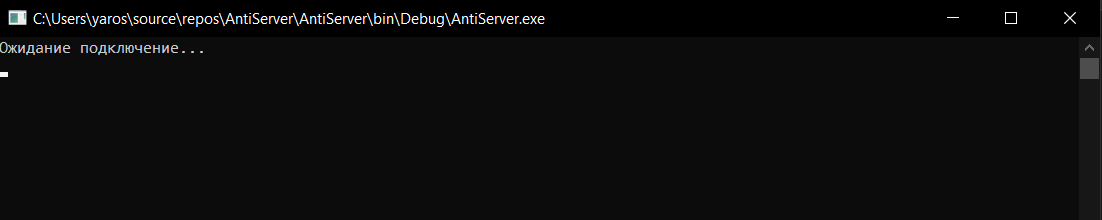
В качестве сервера было выбрано консольное приложение. 

Рисунок 2 - Интерфейс приложения AntiServer

В самом начале работы программы происходит подключение сервера к клиенту с помощью указания номера порта. Далее после считывания данных происходит их обработка и дальнейшее передача их обратно на клиент. На данном этапе разработки антивирусной программы вместо реализации методов, отвечающих за непосредственное сканирование, отправления файлов в карантин и их удаление из системы, был произведен вывод соответствующего смыслового текста в формах программы на клиенте.

*Принцип работы программы:*

При запуске программ сервер постоянно ждет подключения от клиента. Как только будет нажата кнопка «Быстрая проверка» соединение будет установлено.

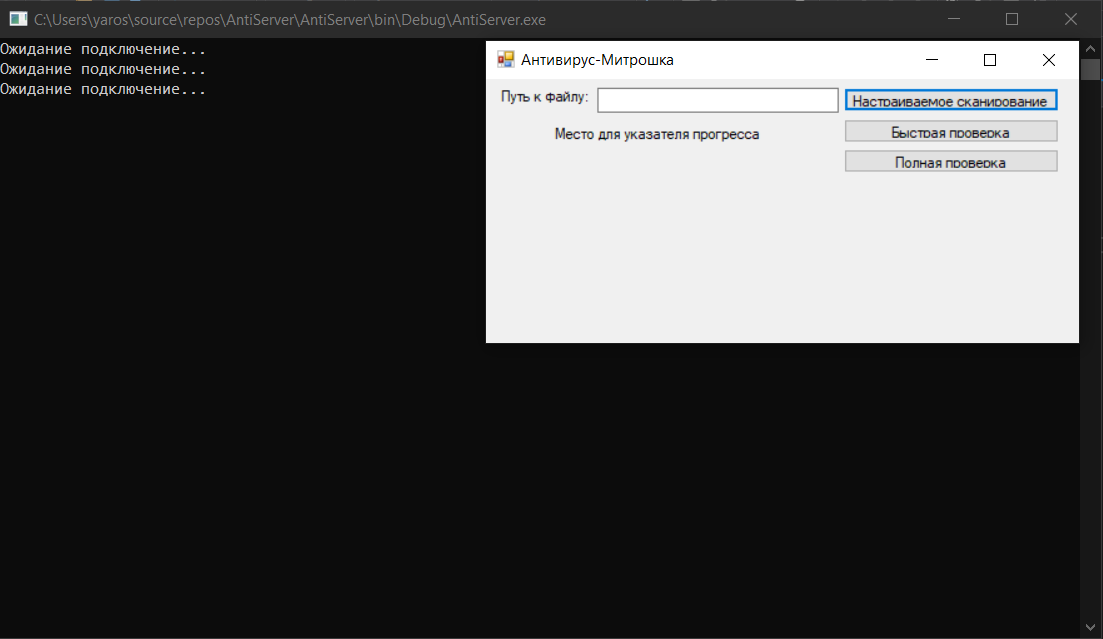


Рисунок 3 - Интерфейс ожидания подключения

После нажатия кнопки «Быстрая проверка» в интерфейсе пользователя было выведена информация о ip-адресе сокета, состояние проверки, выбранные пути и результат работы программы. Также на сервере отражается текущее состояние подключения, действия программы и вывод информации из файла (было сделано для проверки работоспособности передачи данных).

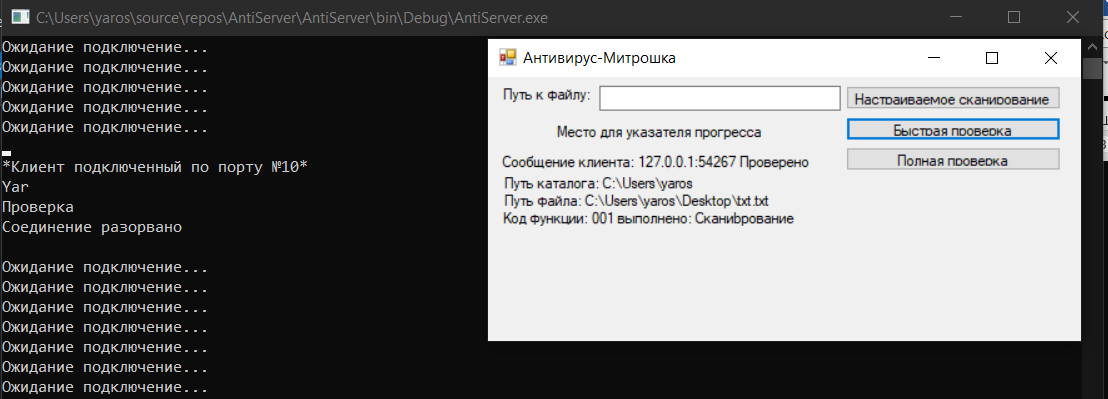


Рисунок 4 - Результат выполнения программы

*Выводы:* в результате реализации клиент-серверного приложения была проведена работа с возможностью подключения клиента и сервера, передачей данных между ними, виртуализацией необходимых для работы антивируса методов и со способами передачи информации.

*Приложение (код программы):*

*Клиент*

namespace AntiCLient

{

public partial class Form1 : Form

{

#region dll

[DllImport("kernel32.dll")]

static extern IntPtr CreateMutex(IntPtr lpMutexAttributes, bool bInitialOwner, string lpName);

[DllImport("kernel32.dll", SetLastError = true)]

static extern UInt32 WaitForSingleObject(IntPtr hHandle, UInt32 dwMilliseconds);

[DllImport("kernel32.dll")]

static extern IntPtr CreateEvent(IntPtr lpEventAttributes, bool bManualReset, bool bInitialState, string lpName);

[DllImport("kernel32.dll")]

static extern bool ResetEvent(IntPtr hEvent);

[DllImport("kernel32.dll")]

static extern bool SetEvent(IntPtr hEvent);

[DllImport("Kernel32.dll", SetLastError = true)]

static extern IntPtr OpenEvent(uint dwDesiredAccess, bool bInheritHandle, string lpName);

[DllImport("kernel32.dll", SetLastError = true)]

[return: MarshalAs(UnmanagedType.Bool)]

static extern bool CloseHandle(IntPtr hObject);

[DllImport("kernel32.dll")]

static public extern bool PulseEvent(IntPtr hEvent);

[DllImport("user32.dll")]

public static extern int GetSystemMetrics(int nIndex);

public const int SM\_MOUSEPRESENT = 19;

public const int SMCMOUSEBUTTONS = 43;

public const int SM\_MOUSEWHEELPRESENT = 75;

#endregion

static TcpListener tcpListener = new TcpListener(10);

public Form1()

{

InitializeComponent();

}

private void button2\_Click(object sender, EventArgs e)

{

Listeners();

tcpListener.Stop();

}

private void Listeners()

{

string txt, pathKatalog, pathFile, code, res;

tcpListener.Start();

Socket socketForClient = tcpListener.AcceptSocket();

if (socketForClient.Connected)

{

int i = 0;

NetworkStream networkStream = new NetworkStream(socketForClient);

System.IO.StreamWriter streamWriter = new System.IO.StreamWriter(networkStream);

System.IO.StreamReader streamReader = new System.IO.StreamReader(networkStream);

string str = "C:\\Users\\yaros C:\\Users\\yaros\\Desktop\\txt.txt 001";

streamWriter.WriteLine(str);

streamWriter.Flush();

Thread.Sleep(1000);

while (i != 1)

{

i++;

string theString = streamReader.ReadLine();

var text = theString.Split(' ');

txt = text[0];

pathKatalog = text[1];

pathFile = text[2];

code = text[3];

label6.Text = "Код функции: " + "001" + " выполнено: " + code;

label3.Text = "Сообщение клиента: " + socketForClient.RemoteEndPoint + " " + txt;

label4.Text = "Путь каталога: " + pathKatalog;

label5.Text = "Путь файла: " + pathFile;

}

streamReader.Close();

networkStream.Close();

streamWriter.Close();

}

socketForClient.Close();

}

}

}

*Сервер*

namespace AntiServer

{

class Program

{

#region dll

[DllImport("kernel32.dll")]

static extern bool ReleaseMutex(IntPtr hMutex);

[DllImport("kernel32.dll")]

static extern IntPtr CreateMutex(IntPtr lpMutexAttributes, bool bInitialOwner, string lpName);

[DllImport("kernel32.dll", SetLastError = true)]

static extern UInt32 WaitForSingleObject(IntPtr hHandle, UInt32 dwMilliseconds);

[DllImport("kernel32.dll")]

static extern IntPtr CreateEvent(IntPtr lpEventAttributes, bool bManualReset, bool bInitialState, string lpName);

[DllImport("kernel32.dll")]

static extern bool ResetEvent(IntPtr hEvent);

[DllImport("kernel32.dll")]

static extern bool SetEvent(IntPtr hEvent);

[DllImport("Kernel32.dll", SetLastError = true)]

static extern IntPtr OpenEvent(uint dwDesiredAccess, bool bInheritHandle, string lpName);

[DllImport("kernel32.dll", SetLastError = true)]

[return: MarshalAs(UnmanagedType.Bool)]

static extern bool CloseHandle(IntPtr hObject);

[DllImport("kernel32.dll")]

static public extern bool PulseEvent(IntPtr hEvent);

#endregion

static void Main(string[] args)

{

TcpClient socketForServer;

while (true)

{

try

{

try

{

Console.WriteLine("Ожидание подключение...");

Thread.Sleep(2000);

socketForServer = new TcpClient("localHost", 10);

}

catch

{

continue;

}

NetworkStream networkStream = socketForServer.GetStream();

System.IO.StreamWriter streamWriter = new System.IO.StreamWriter(networkStream);

System.IO.StreamReader streamReader = new System.IO.StreamReader(networkStream);

Console.WriteLine("\r\n\*Клиент подключенный по порту №10\*");

string strvod = streamReader.ReadLine();

var text = strvod.Split(' ');

string pathKatalog = text[0];

string pathFile = text[1];

string code = text[2];

string result;

System.IO.StreamReader sr = new System.IO.StreamReader(pathFile);

Console.WriteLine(sr.ReadToEnd());

Console.WriteLine("Проверка");

Thread.Sleep(1000);

string str = "Проверено" + " " + pathKatalog + " " + pathFile + " " + SwitchedC(code); ;

streamWriter.WriteLine(str);

streamWriter.Flush();

networkStream.Close();

Thread.Sleep(1000);

Console.WriteLine("Соединение разорвано\r\n");

}

catch

{

Console.WriteLine("Ошибка считывания");

}

}

}

static string SwitchedC(string str)

{

string result = "";

switch (str)

{

case "000":

result = StartScan();

break;

case "001":

result = StopScan();

break;

case "002":

result = MoveToQ();

break;

case "003":

result = RemoveFile();

break;

}

return result;

}

static string StartScan()

{

string res = "Отсканированно";

return res;

}

static string StopScan()

{

string res = "Сканиbрование завершено";

return res;

}

static string MoveToQ()

{

string res = "Карантин";

return res;

}

static string RemoveFile()

{

string res = "Удален";

return res;

}

}

}