Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

Клиент-серверное приложение для обмена сообщениями

Пояснительная записка

RU. 02068048.502900.01.01

# АННОТАЦИЯ

Данный программный документ описывает программу «TAHC-CHAT». В её основе лежит быстрый обмен сообщениями между пользователями. Исходный код разработанного программного обеспечения написан на языке программирования C++. Текст программы «TAHC-CHAT» приведен в Приложении А данного программного документа.

Основной задачей «TAHC-CHAT» является обеспечение обмена текстовыми сообщениями между лицензированными пользователями. Программное обеспечение состоит из двух частей: Клиент, устанавливаемый на машине пользователя и Сервер, находящийся на специально отведенной производительной машине заказчика. С помощью клиентской части, пользователи должны подключиться к серверу для дальнейшего обмена сообщениями с остальными участниками общения. Вход на сервер осуществляется по паролю, который выдается администратором сервера. Правильность пароля проверяется на сервере с помощью особого алгоритма.

Пользователи клиентской части программы могут отправлять на сервер любые текстовые сообщения. После получения сообщения, сервер пересылает его всем остальным клиентам.

Оформление программного документа произведено по требованиям ЕСПД (ГОСТ 19.101-77 [[1]](#footnote-1), ГОСТ 19.103-77 [[2]](#footnote-2), ГОСТ 19.104-78\* [[3]](#footnote-3), ГОСТ 19.105-78\* [[4]](#footnote-4), ГОСТ 19.106-78\* [[5]](#footnote-5), ГОСТ 19.401-78 [[6]](#footnote-6), ГОСТ 19.604-78\* [[7]](#footnote-7)).

# СОДЕРЖАНИЕ

[АННОТАЦИЯ 2](#_Toc515473852)

[СОДЕРЖАНИЕ 3](#_Toc515473853)

[1 СЕРВЕРНАЯ ЧАСТЬ 4](#_Toc515473854)

[2 КЛИЕНТСКАЯ ЧАСТЬ 5](#_Toc515473855)

[3 ОПИСАНИЕ ЗАЩИТЫ ПРОГРАММНОГО ПРОДУКТА 6](#_Toc515473856)

[3.1 Защита от статического анализа. 6](#_Toc515473857)

[3.1.1 Разнообразие выходов из программы. 6](#_Toc515473858)

[3.1.2 Множественные проверки. 6](#_Toc515473859)

[3.2 Защита от динамического анализа. 7](#_Toc515473860)

[3.2.1 Пароль в виде Hash. 7](#_Toc515473861)

[3.2.2 Программы дизассемблирования. 8](#_Toc515473862)

[3.2.3 Таймеры. 9](#_Toc515473863)

[3.3 Защита от патчинга. 9](#_Toc515473864)

[3.3.1 Контроль целостности. 9](#_Toc515473865)

[5 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА 10](#_Toc515473866)

[6 АУДИТ БЕЗОПАСНОСТИ 11](#_Toc515473867)

[ПРИЛОЖЕНИЕ А 20](#_Toc515473868)

[ПРИЛОЖЕНИЕ Б 49](#_Toc515473869)

# 1 СЕРВЕРНАЯ ЧАСТЬ

Сервер представляет собой консольное приложение, которые ждет подключения к нему пользователей. После того, как к нему подключиться пользователь, он выдает ему персональный id (отсчет идет с 0, и прибавляется после подключения нового пользователя). Передача сообщений основана на протоколе TCP. Клиент может воспользоваться несколькими командами. Например, с помощью одной из них можно узнать, сколько пользователей в данный момент онлайн. Сервер может работать более чем с 300 пользователей одновременно, без какой-либо задержки. При более большом количестве пользователей, явно ощущается задержка в передаче сообщений между пользователями. С помощью сервера невозможно отправить сообщения, это реализовано лишь на клиенте.

Все необходимые настройки сервера (IP, PORT) хранятся в файле in.ini, а пароль для запуска сервера хранится в pass.ini. Благодаря этому включение сервера не требует никакого ввода данных.

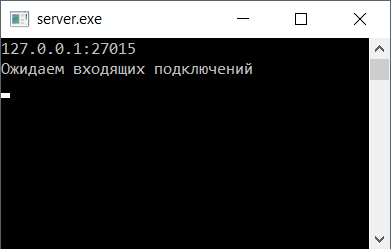


Рисунок 1 – Демонстрация работы сервера

# 2 КЛИЕНТСКАЯ ЧАСТЬ

Клиент также представляет собой консольное приложение. Он предназначен для подключения пользователя к серверу. Необходимые данные для подключения (IP, PORT) также хранятся в файле in.ini, в файле pass.ini хранится пароль.

При удачном соединение с сервером на клиенте нужно ввести свой Nickname. Если пароль подходит, то клиент может общаться в этом чате. Отправляемые пользователем сообщения, кроме специальных команд (например /users), видны всем подключенным в данный момент пользователям. Чтобы выйти из чата, можно просто закрыть консольное приложение, а можно ввести команду “/q”.

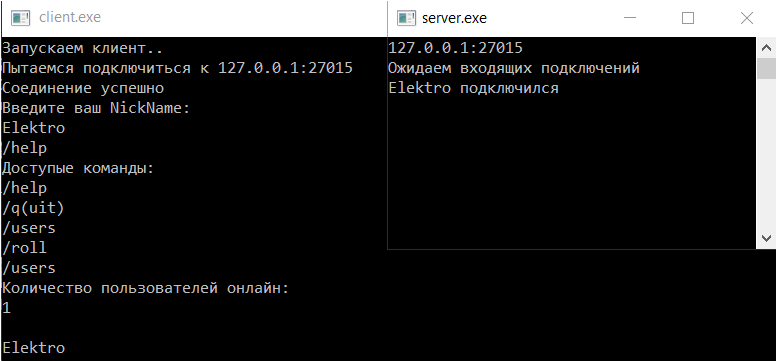
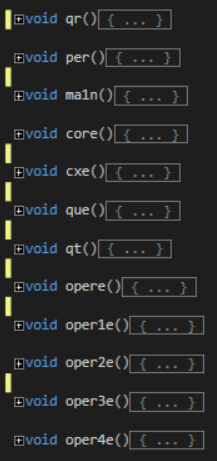


Рисунок 2 – Демонстрация работы «TAHC-CHAT»

# 3 ОПИСАНИЕ ЗАЩИТЫ ПРОГРАММНОГО ПРОДУКТА

## **3.1 Защита от статического анализа.**

### **3.1.1 Разнообразие выходов из программы.**

****

Было решено использовать дюжину функций для выхода из программы (рисунок 3).

При различных ошибках, будь то неправильный пароль, или изменение целостности программы, выполняется одна из этих функций. Все они имеют одинаковую структуру, но имеют небольшие различия в коде. Отличия нужны для того, чтобы компилятор не упрощал выполнение той или иной функции, тем самым усложняя процесс статического и динамического анализа. Также было решено использовать неправильное отображение информации о запуске программы. Это чётко видно на картинке 4, где выводится надпись “Open server on”. На самом же деле вся эта функция ведет прямиком к выходу из программы.

Рисунок 3 – дюжина функций выхода

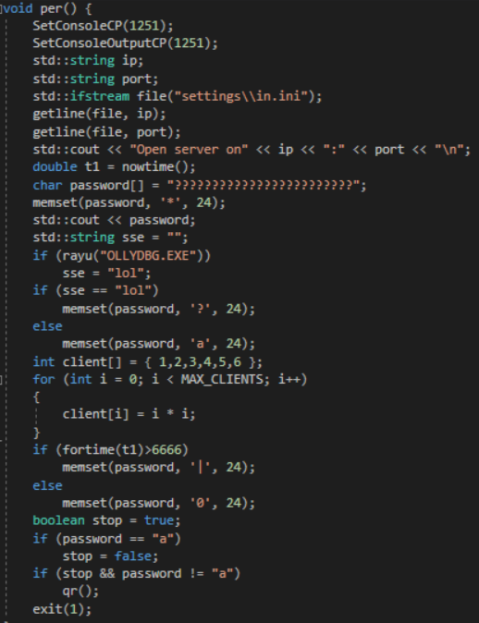


Рисунок 4 – код выхода из программы

### **3.1.2 Множественные проверки.**

Взломщик, даже если и обнаружит участок кода, который отвечает за закрытие приложения, всё равно не сможет добиться работоспособности чата. На протяжении всей программы абсолютно все способы защиты проверяются по два, а то и по три раза. Это явно создаст проблему для взломщика ПО.

## **3.2 Защита от динамического анализа.**

### **3.2.1 Пароль в виде Hash.**

Клиент вводит пароль, он отправляется на сервер, переводится в hash и сверяется, подходит ли он под критерии. Hash создается на основе логина пользователя. Сам алгоритм берет поочередно каждую букву логина+логина, преобразует его в ASCII, а затем умножает на одно из простых чисел массива (рисунок 5). К конечному результату добавляется фиксированное число, созданное тем же алгоритмом, оно нужно для усложнения hash суммы.

int gettingHASH(std::string nick) {

std::string forNICK = "";

forNICK = nick + nick;

const char \* massivHASH = forNICK.c\_str();

int resultHASH = 20093; //Ju36IOszf4pppqLL42mv

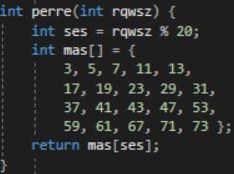
for (int j = 0; j < strlen(massivHASH); j++)

resultHASH = resultHASH + ((int)massivHASH[j] \* perre(j));

resultHASH = resultHASH + 43768;

return resultHASH;

}

На сервере реализован тот же алгоритм, но вместо логина используется IP+port+IP. И всё также добавляется фиксированное число, но оно имеет другое значение.

int gettingHASHserver(std::string ip, std::string port) {

std::string all = "";

all = ip + port;

all = all + ip;

const char \* massivHASH = all.c\_str();

int resultHASH = 0;

resultHASH += 36584; //P9u6Rhb7Dvc67pZQ

for (int j = 0; j < strlen(massivHASH); j++) Рисунок 5 – массив для хэша пароля

resultHASH = resultHASH + ((int)massivHASH[j] \* perre(j));

return resultHASH;

}

Пароль можно получить, используя программу генератор. В основе её реализации лежит две функции, одна служит для создания пароля для сервера, другая для клиента.

Соответственно для сервера:

std::string gettingPASSserver(std::string ip, std::string port) {

std::string forPASS = "";

int resultHASH1 = gettingHASHserver(ip, port);

int resultHASH2 = 0;

boolean br = true;

srand(time(0));

int j = 0;

while (br == true) {

forPASS = "";

resultHASH2 = 0;

br = false;

j = -1;

while (true) {

int s = randomi();

j = j + 1;

forPASS = forPASS + ((char)s);

resultHASH2 = resultHASH2 + ((int)((char)s)\*perre(j));

if (resultHASH1 == resultHASH2) {

std::cout << resultHASH2 << "\n";

std::cout << forPASS << " This is Server\n";

break;

}

if (resultHASH2>resultHASH1) {

br = true;

break;

}

}

}

return forPASS;

}

Для клиента:

std::string gettingPASS(std::string nick) {

std::string forPASS = "";

int resultHASH1 = gettingHASH(nick);

int resultHASH2 = 0;

boolean br = true;

srand(time(0));

int j = 0;

while (br == true) {

j = -1;

forPASS = "";

resultHASH2 = 0;

br = false;

while (true) {

int s = randomi();

forPASS = forPASS + ((char)s);

j = j + 1;

resultHASH2 = resultHASH2 + ((int)((char)s)\*perre(j));

if (resultHASH1 == resultHASH2) {

std::cout << resultHASH2 << "\n";

std::cout << forPASS << " this is LOGIN\n";

break;

}

if (resultHASH2>resultHASH1) {

br = true;

break;

}

}

}

return forPASS;

}

Например, конечный пароль для сервера, запущенного на 127.0.0.1:27015 будет выглядеть так: )Hy:+<UkFP8MXRIhg3PhPBTK-P2a,hI  
А пароль, для пользователя с Nickname Elektro:   
d;t2P,YQ\k$NnB:4J8R$-#'\[k[I>]AoVF4}{Y

### **3.2.2 Программы дизассемблирования.**

В чате реализована проверка на то, запущены ли программы дизассемблирования. Сама функция выглядит вот так:

bool isProcessRun(const char \*processName)

{

HANDLE hSnap = NULL;

PROCESSENTRY32 pe32;

pe32.dwSize = sizeof(PROCESSENTRY32);

hSnap = CreateToolhelp32Snapshot(TH32CS\_SNAPPROCESS, 0);

if (hSnap != NULL)

{

if (Process32First(hSnap, &pe32))

{

if (lstrcmp(pe32.szExeFile, processName) == 0)

return TRUE;

while (Process32Next(hSnap, &pe32))

if (lstrcmp(pe32.szExeFile, processName) == 0)

return TRUE;

}

}

CloseHandle(hSnap);

return FALSE;

}

Допустим, нам нужно узнать, запущена ли Ollydbg, нужно просто обратиться к функции isProcessrun(“OLLYDBG.exe”). Если функция находит программу в запущенных, она возвращает значения true.

### **3.2.3 Таймеры.**

Любое проявление динамического анализа контролируется таймерами на всём участке кода. Таймер выглядит следующим образом: первая функция отвечает за начало отсчета, откуда стартует таймер, вторая функция возвращает число в мс (рисунок 7). С помощью второй функции всегда можно выбрать, на сколько секунд поставить таймер. Все таймеры в коде взяты с запасом для того, чтобы слабопроизводительные компьютеры не вызывали ложного срабатывания таймера.

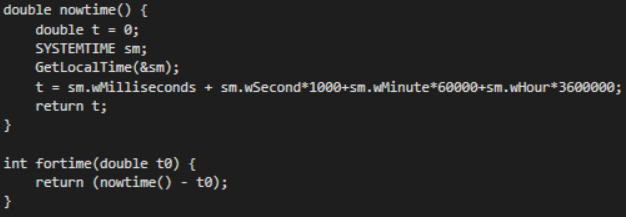


Рисунок 7 – код, нужный для запуска таймера

## **3.3 Защита от патчинга.**

### **3.3.1 Контроль целостности.**

Реализация контроля целостности осуществляется вне основных программ. При запуске сервера (клиента) он даёт команду другой программе, которая в целях усложнения её обнаружения замаскирована под базу данных (database.rar), на запуск. Программа также защищена от взлома, но более слабо. Проверка целостности проходит в несколько этапов. Первый этап – нахождение всех присутствующих дисков на используемом компьютере. Второй этап – запись на найденный диск, в папку temp, файла ultra.dll. В этот файл записываются две строчки: первая - нужный hash файла, вторая – текущий hash файла. В целях того, если вдруг взломщик сможет обнаружить этот файл и решит его изменить – у него это не выйдет. Во-первых, данные операции проходят очень быстро, во-вторых, если он будет использовать точки остановы – то программа закроется благодаря использованию таймера. В-третьих, если он сможет обойти таймер, то ему придется эмулировать запуск файла temp.exe, иначе программа, которая вызвала проверку hash, закрывается. Листинг программы предоставлен в приложении Б.

**5 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА**

Проведем проверку работоспособности чата «TAHC-CHAT».

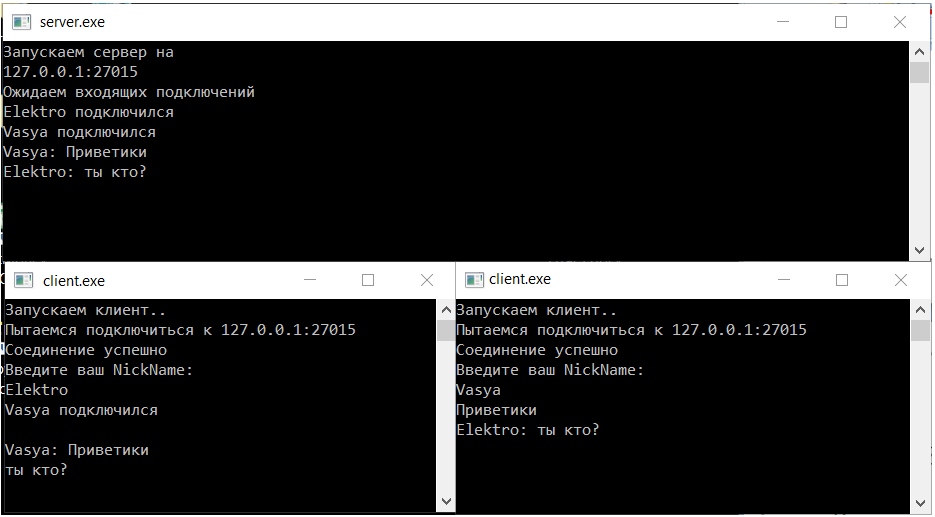


Рисунок 8 – работоспособность чата

Нагрузить чат большим количеством пользователей задача не самая простая. Для каждого нового пользователя нужно сгенерировать новый пароль. На ранних стадиях разработки сервер справлялся с нагрузкой, примерно, в 250 пользователей. При большей нагрузке чувствовалась задержка отправки сообщения всем пользователям.

Программа не будет работать, если не ввести пароль в файле, который лежит в папке settings. При попытке запустить сервер или клиент с использованием неправильного или отсутствующего пароля происходит немедленное закрытие программы.

# 6 АУДИТ БЕЗОПАСНОСТИ

На данном этапе мною был произведен анализ программного продукта MESSLAND студента группы ИБ-116 Клименкова Даниила. Были получены две папки, сервер, клиент, а также в папках находились необходимые файлы для запуска приложения.

При попытке запустить server.exe консоль требовала ввод каких-то данных, скорее всего это пароль.

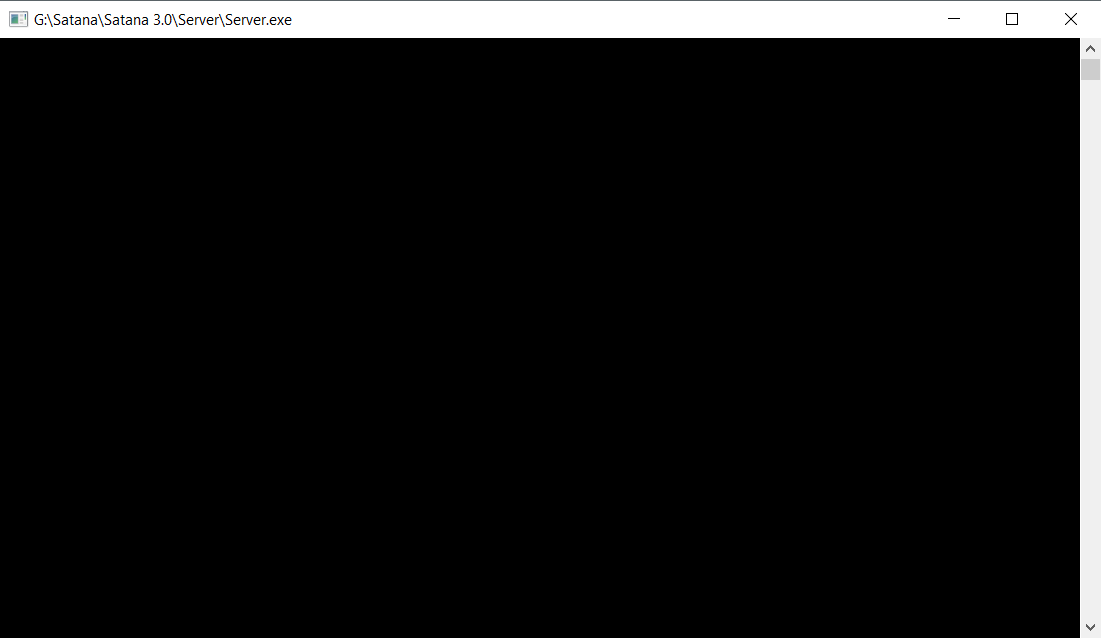


Рисунок 9 – запуск сервера

Прежде, чем приступать непосредственно к анализу дизассемблированного языка, нужно узнать, какие String данные хранит server, самые интересные из них показаны на Рисунке 10.

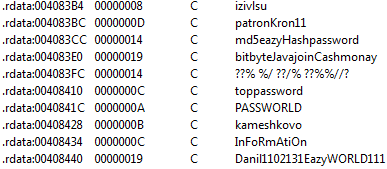


Рисунок 10 – возможные пароли для сервера или клиента

Все эти данные похожи на пароль, но нет, попытка использовать каждую из этих строчек не дала никакого результата, программа сразу закрывалась.

Следующим этапом стал поиск места, непосредственно связанного с паролем.  
Благодаря программе IDA PRO 6.0 найдена строчка, которая называется GetHash (Рисунок 11).

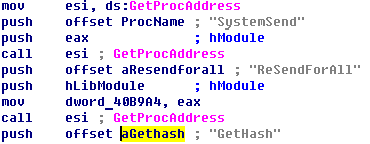


Рисунок 11 – функция GetHash

Нам необходимо либо узнать, как создается пароль, либо как-то обойти его, в качестве динамического анализа была выбрана программа дизассемблирования OLLYDBG.

Так как сервер, при включении, запрашивает какие-то данные, скорее всего пароль, нам необходимо найти это место в дизассемблере. После недолгого дебагинга программы была найдена та самая строчка, которая вызывает функцию GetHash, описанную ранее (Рисунок 12).

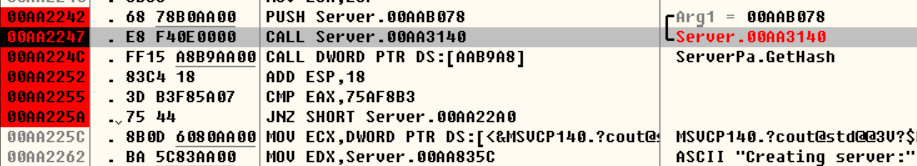


Рисунок 12 – строчка GetHash на сервере

При продолжении программы следующая команда JNZ вызывает выход из программы (Рисунок 13). Но если перехода по JNZ не будет, то нам выдастся сообщение “Creating server”

.



Рисунок 13 – завершение программы

Вбиваем любой пароль и пробуем поменять флаг у JNZ во время работы программы для того, чтобы понять, действительно ли пароль проверяется здесь (Рисунок 14). А затем продолжаем её выполнение (Рисунок 15).



Рисунок 14 – замена флага JNZ



Рисунок 15 – проверка работоспособности сервера

Получилось, сервер запустился.

Теперь необходимо сделать версию сервера, в которой, при вводе любого пароля программа запускалась, для этого удалим переход JNZ простым ассемблеровским nop (Рисунок 16 – до, Рисунок 17 – после).

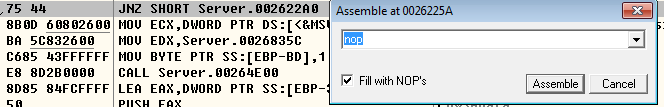


Рисунок 16 – до изменения JNZ

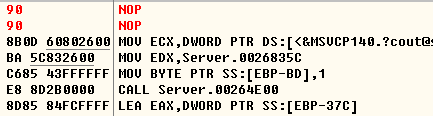
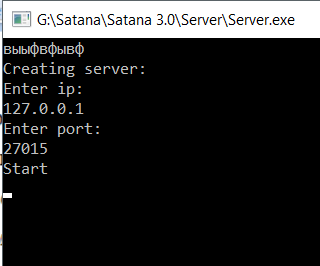


Рисунок 17 – после изменения JNZ

Чтобы убедиться, что программа работает правильно - запускаем её и вводим IP, порт для сервера (Рисунок 18).



(Рисунок 18) – работоспособность программы

Всё работает, сохраняем пропатченный сервер для дальнейшего использования.

В папке с клиентом находятся несколько файлов: Hung.txt, Settings.txt, AddinglLib.dll. Очевидно, что Settings.txt нужен для того, чтобы вписать туда IP и порт сервера, к которому нужно подключиться, сделаем это и попытаемся запустить клиент (Рисунок 19).

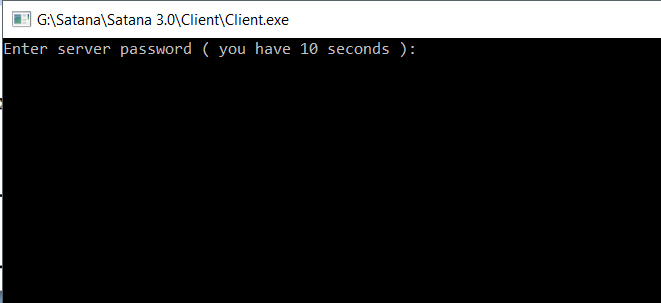


Рисунок 19 – ввод пароля на клиенте

От нас требуют ввода пароля, если пароль не ввести программа закрывается, при вводе спонтанного пароля – программа также закрывается. Если пароль не вводить, то на сервере появляется сообщение “timeout client”, если же пароль ввести, то на сервере появляется сообщение “Error password” (Рисунок 20). Возможно, это как-то поможет нам в дальнейшем.

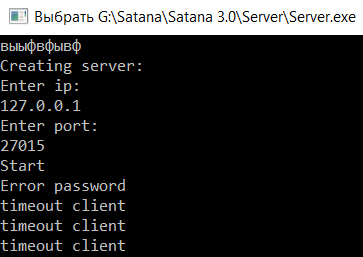


Рисунок 20 – Error password на сервере

Воспользуемся IDA PRO 6.0 для дизассемблирования клиента. Просмотр String данных может лишь запутать, так как по сравнению с сервером их здесь в разы больше (если быть точным, String данных 622 штуки). Естественно, проверять каждый не имеет смысла, поэтому проверим самые подозрительные (Рисунок 21).

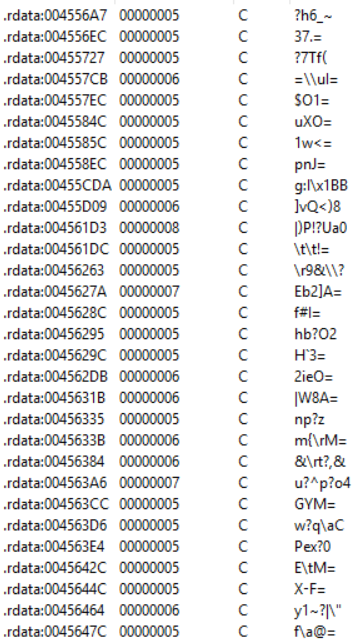


Рисунок 21 – часть списка String данных клиента

Как и в случае с сервером перебор этих вариантов не дал никакого результата, подключиться к серверу все также не удавалось. Однако в String данных можно обнаружить строки, которые показывают, что клиент подключился, скорее всего, это либо “Winner winner chicken dinner!!!”, либо “Successful. Hello,”. Но, во втором варианте присутствует запятая, которая явно указывает на то, что пользователь уже ввел пароль, а вместе с ним и nickname. Всё это означает, что первая фраза является ключевой, именно она отвечает за подключение клиента к серверу и также именно она выводится на экран после ввода пароля с клавиатуры. Пароль же, который должен быть записан в файле, скорее всего, проверяется где-то на клиенте.

Пароль явно отправляется на сервер, там проверяется, а затем пользователю отправляются указания, приняли его или нет. Посмотрим, что отправляет сервер клиенту при подключении пользователя. Для этого находим участок кода на сервере, отвечающий за отправку указаний (Рисунок 22).



Рисунок 22 – команда Send на сервере

Запускаем клиент и пытаемся подключиться к серверу. Мы видим, что сервер отправляет два сообщения пользователю. Первое сообщение отправляет число из стека, а точнее “0” (Рисунок 23).

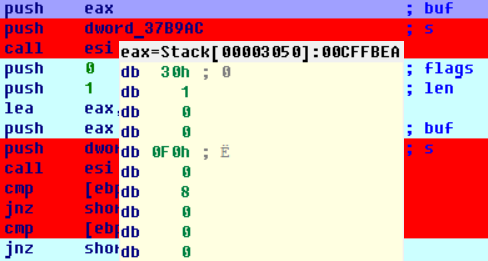


Рисунок 23 – стек с числом 0

Вторая команда send отправляет также число из стека и оно также равно 0 (Рисунок 24).

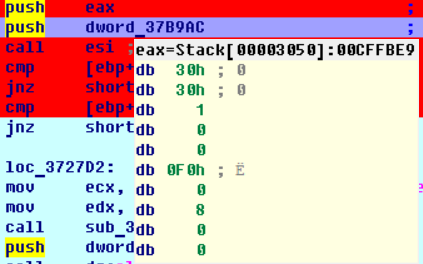


Рисунок 24 – стек с числом 0

Скорее всего именно из-за этих нулей клиент понимает, что его не приняли. Нужно найти участок кода на сервере, на котором происходит получение пароля от клиента. Сделать это просто, в поиске пишем функцию “recv” и находим нужные нам строки (Рисунок 25).



Рисунок 25 – команда Recv на сервере

Как видим, сервер принимает два каких-то значения, скорее всего одно из них это пароль. Если и дальше идти по коду, то мы вернемся к тем самым Рисункам 23 и 24. А также там есть два перехода (Рисунок 26), причем первый, ведет прямиком к надписи “Error password” и закрывает socket для клиента (Рисунок 27).



Рисунок 26 – JNZ переход к Error Password

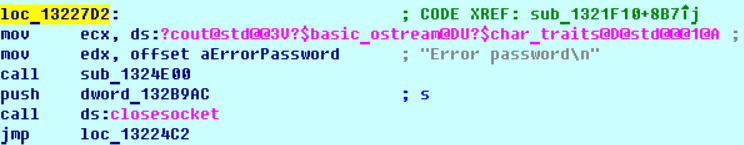


Рисунок 27 – Error Password и закрытие socket

Скорее всего, всё, что нам нужно, это либо поменять местами эти переходы, либо изменить само сравнение, по которому происходит переход. Ну а также изменить 0 числа, которые отсылаются клиенту, оповещая его о том, что он ввел неправильный пароль.

Для начала отправим клиенту не 0, а что-то другое, находим в OLLYDBG строчку отправки клиенту этого сообщения, Рисунок 28.

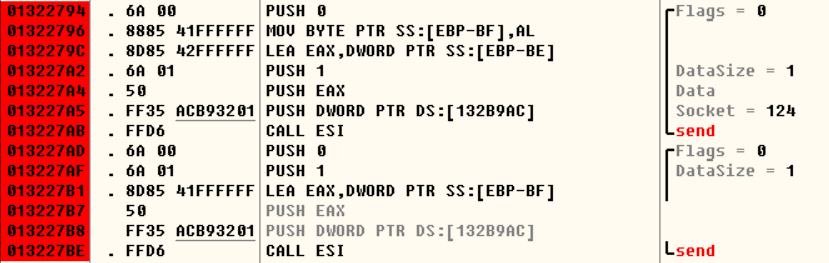


Рисунок 28 – отправка 0 клиенту

Попробуем отправить не число из стека, а допустим единицу (Рисунок 29).

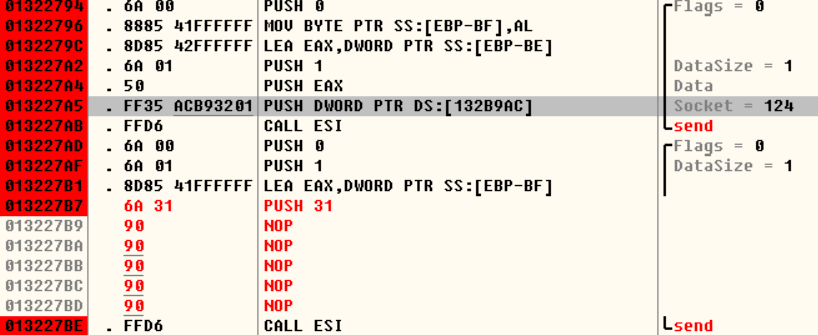


Рисунок 29 – попытка отправить единицу клиенту

А также, как и ранее оговаривало, поменяем прыжок с Error Password, на другой, который находится после. Ещё нужно убрать сравнение перед первым JNZ, а сам JNZ заменить под JE и поменять его на переход нужной нам ветки (Рисунок 30 – до, рисунок 31 – после).

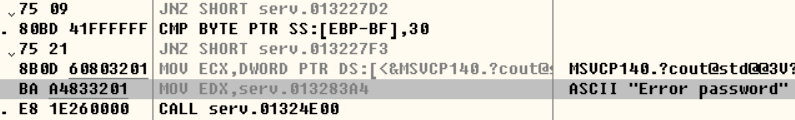


Рисунок 30 – до изменения команд JNZ

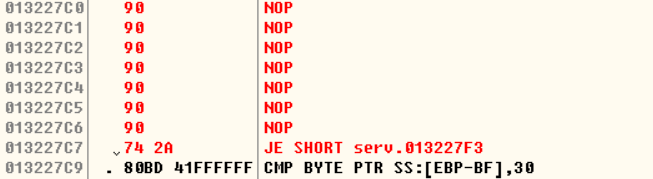


Рисунок 31 – после изменения команды JNZ на команду JE

Попробуем подключиться к серверу. У нас вроде бы выходит, но что-то явно не так, сервер принимает клиента, но сам клиент не может зайти, а потом он упирается в таймер 10 секунд и программа завершается. А сервер пишет, что пользователь подключился, а затем отключился (Рисунок 32).

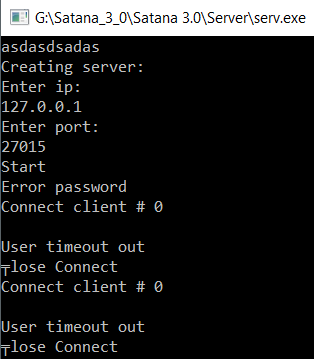


Рисунок 32 – попытка подключения клиента

Нужно убедиться, что клиент принимает нужные сведенья от сервера. Через IDA PRO обнаруживаем, что наше изменение команды Send повлекло ошибку, в результате которой клиент не получает второе сообщение, оно просто не отправляется. Попробуем заменить стек EAX, который можно увидеть на Рисунке 28 стеком EBX (Рисунок 33).

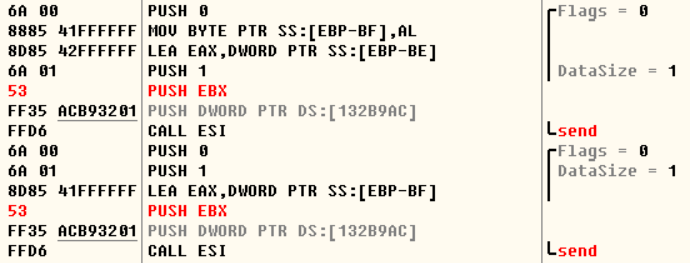


Рисунок 33 – замена стека EAX на стек EBX

Пробуем подключиться к серверу.

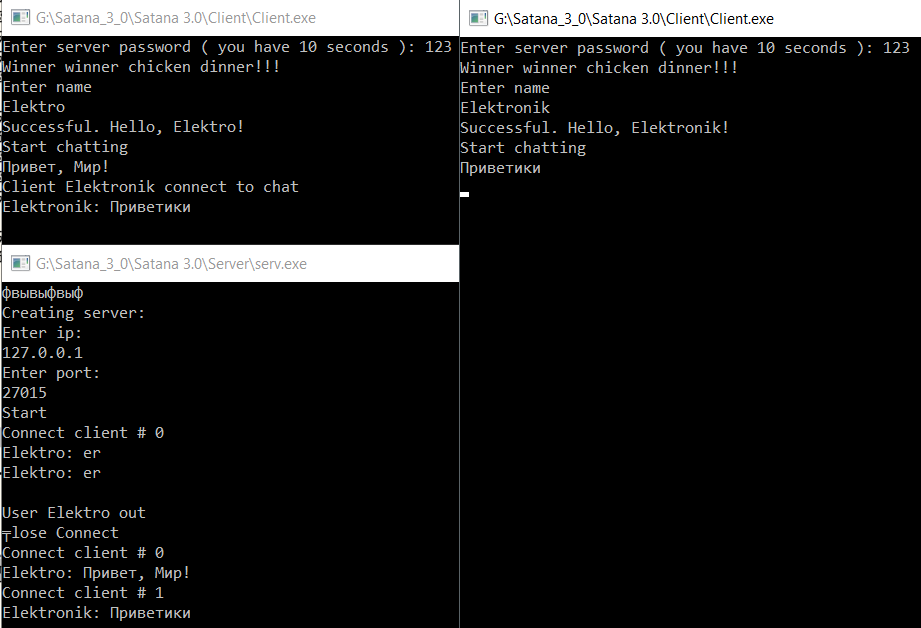


Рисунок 34 – работоспособность чата в программе-отладчике

Сохраняем пропатченный сервер и пробуем запустить его. Попытка запатчить неудачна, как только пользователь пытается подключиться к серверу он закрывается. Скорее всего где-то присутствует ошибка, которая ведет к закрыванию сервера.

Если взглянуть на рисунок 33, то можно заметить, что мы использовали стек EBX вместо EAX, возможно ошибка в том, что к этому стеку лучше не обращаться. Заменим EBX снова на EAX, но изменим строку LEA EAX, DWORD PIR SS:[EBP-BE] и LEA EAX, DWORD PIR SS:[EBP-BF] на строки без BE и BF (Рисунок 35), результат команды LEA должен поменяться.

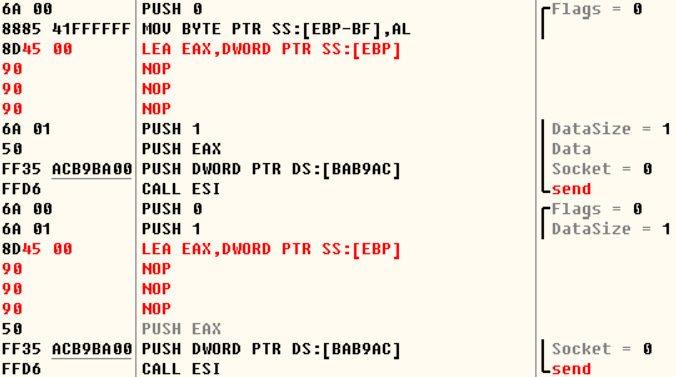


Рисунок 35 – изменение команды LEA

Пробуем снова сохранить полученный сервер и запустить его (Рисунок 36).

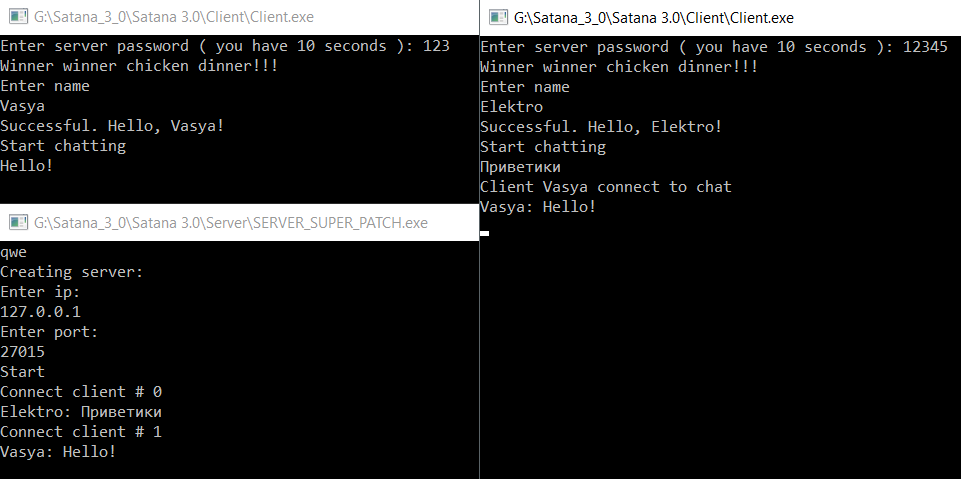


Рисунок 36 – работоспособность чата

Во время аудита безопасности, в отношении полученного программного продукта, были обнаружены несколько методов защиты, такие как: статический (проверка пароля, использование memset, memcopy, множественные неверные пароли в исходном коде, текстовый документ, с паролем, который на самом деле таковым не является), динамический (таймер на ввод паролей). В данном программном продукте была обнаружена уязвимость на сервере, благодаря анализу ассемблерного кода которой удалось взломать.

# ПРИЛОЖЕНИЕ А

Текст программы «NCHAT» на исходном языке.

**«server.exe»:**

Server.cpp:

#include <iostream>

#include <winsock2.h>

#include <thread>

#include <vector>

#include <myhash>

#include <myhash2>

#include <ShellAPI.h>

#include <windows.h>

#include <cstdio>

#include <tlhelp32.h>

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

TCHAR \*IP\_ADDRESS;

TCHAR \*DEFAULT\_PORT;

#define DEFAULT\_BUFLEN 512

int num\_clients = 0;

const char OPTION\_VALUE = 1;

const int MAX\_CLIENTS = 10000;

struct client\_type{

std::string nickname;

std::string password;

int id;

SOCKET socket;};

int process\_client(client\_type &new\_client, std::vector<client\_type> &client\_array, std::thread &thread){

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string msg = "";

char tempmsg[DEFAULT\_BUFLEN] = "";

while (1) {

srand(time(NULL));

memset(tempmsg, 0, DEFAULT\_BUFLEN);

if (new\_client.socket != 0) {

int iResult = recv(new\_client.socket, tempmsg, DEFAULT\_BUFLEN, 0);

if (iResult != SOCKET\_ERROR && iResult != 0) {

if (!strcmp("/users", tempmsg)) {

msg = "Количество пользователей онлайн: \n" + std::to\_string(num\_clients + 1);

send(new\_client.socket, msg.c\_str(), strlen(msg.c\_str()), 0);

for (int i = 0; i <= num\_clients; i++)

if (client\_array[i].socket != INVALID\_SOCKET) {

msg = "\n" + client\_array[i].nickname;

send(new\_client.socket, msg.c\_str(), strlen(msg.c\_str()), 0);

}}

else

if (!strcmp("/roll", tempmsg)) {

msg = "Выдано случайное число (1-100): \n" + std::to\_string(rand() % 100 + 1);

send(new\_client.socket, msg.c\_str(), strlen(msg.c\_str()), 0);

}

else

if (!strcmp("/help", tempmsg)) {

msg = "Доступые команды: \n/help\n/q(uit)\n/users\n/roll";

send(new\_client.socket, msg.c\_str(), strlen(msg.c\_str()), 0);

}

else {

if (strcmp("", tempmsg))

msg = new\_client.nickname + ": " + tempmsg;

std::cout << msg.c\_str() << std::endl;

for (int i = 0; i < MAX\_CLIENTS; i++)

{

if (client\_array[i].socket != INVALID\_SOCKET)

if (new\_client.id != i)

iResult = send(client\_array[i].socket, msg.c\_str(), strlen(msg.c\_str()), 0);

}}}

else

{

double t1 = nowtime();

msg = new\_client.nickname + " отключился";

num\_clients = num\_clients - 1;

std::cout << msg << std::endl;

closesocket(new\_client.socket);

closesocket(client\_array[new\_client.id].socket);

client\_array[new\_client.id].socket = INVALID\_SOCKET;

for (int i = 0; i < MAX\_CLIENTS; i++)

{

if (client\_array[i].socket != INVALID\_SOCKET)

iResult = send(client\_array[i].socket, msg.c\_str(), strlen(msg.c\_str()), 0);

}

if (fortime(t1) > 2002)

return 0;

break;

}}}

thread.detach();

return 0;

}

bool isProcessRun(const char \*processName)

{

HANDLE hSnap = NULL;

PROCESSENTRY32 pe32;

pe32.dwSize = sizeof(PROCESSENTRY32);

hSnap = CreateToolhelp32Snapshot(TH32CS\_SNAPPROCESS, 0);

if (hSnap != NULL)

{

if (Process32First(hSnap, &pe32))

{

if (lstrcmp(pe32.szExeFile, processName) == 0)

return TRUE;

while (Process32Next(hSnap, &pe32))

if (lstrcmp(pe32.szExeFile, processName) == 0)

return TRUE;

}}

CloseHandle(hSnap);

return FALSE;

}

bool(\*rayu)(const char \*processName) = &isProcessRun;

void qr() {

exit(1);

}

void per() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" << port << "\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLYDBG.EXE"))

sse = "lol";

if (sse == "lol")

memset(password, '?', 24);

else

memset(password, 'a', 24);

int client[] = { 1,2,3,4,5,6 };

for (int i = 0; i < MAX\_CLIENTS; i++) client[i] = i \* i;

if (fortime(t1)>6666)

memset(password, '|', 24);

else memset(password, '0', 24);

boolean stop = true;

if (password == "a")

stop = false;

if (stop && password != "a") qr();

exit(1);

}

void ma1n() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" << port << "\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLYDBG.EXE"))

sse = "lol";

if (sse == "lol") memset(password, '?', 24);

else memset(password, 'a', 24);

int client[] = { 1,2,3,4,5,6 };

for (int i = 0; i < MAX\_CLIENTS; i++) client[i] = i \* i;

if (fortime(t1)>6666) memset(password, '|', 24);

else memset(password, '0', 24);

boolean stop = true;

if (password == "a")

stop = false;

if (stop && password != "a") qr();

exit(1);

}

void core() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" << port << "\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLYDBG.EXE")) sse = "lol";

if (sse == "lol") memset(password, '?', 24);

else memset(password, 'a', 24);

int client[] = { 1,2,3,4,5,6 };

for (int i = 0; i < MAX\_CLIENTS; i++) client[i] = i \* i;

if (fortime(t1)>6666) memset(password, '|', 24);

else memset(password, '0', 24);

boolean stop = true;

if (password == "a") stop = false;

if (stop && password != "a") qr();

exit(1);

}

void cxe() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" << port << "\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLYDBG.EXE")) sse = "lol";

if (sse == "lol") memset(password, '?', 24);

else memset(password, 'a', 24);

int client[] = { 1,2,3,4,5,6 };

for (int i = 0; i < MAX\_CLIENTS; i++)

client[i] = i \* i;

if (fortime(t1)>6666) memset(password, '|', 24);

else memset(password, '0', 24);

boolean stop = true;

if (password == "a") stop = false;

if (stop && password != "a") qr();

exit(1);

}

void que() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" << port << "\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLYDBG.EXE")) sse = "lpol";

if (sse == "lol") memset(password, '?', 24);

else memset(password, 'a', 24);

int client[] = { 1,2,3,4,5,6 };

for (int i = 0; i < MAX\_CLIENTS; i++)

client[i] = i \* i;

if (fortime(t1)>6666) memset(password, '|', 24);

else memset(password, '0', 24);

boolean stop = true;

if (password == "a") stop = false;

if (stop && password != "a") qr();

exit(1);

}

void qt() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" << port << "\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLYDBG.EXE"))

sse = "lol";

if (sse == "lol") memset(password, '?', 24);

else memset(password, 'a', 24);

int client[] = { 1,2,3,4,5,6 };

for (int i = 0; i < MAX\_CLIENTS; i++) client[i] = i \* i;

if (fortime(t1)>6666) memset(password, '|', 24);

else memset(password, '0', 24);

boolean stop = true;

if (password == "a") stop = false;

if (stop && password != "a") qr();

exit(1);

}

void opere() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" << port << "\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLYDBG.EXE")) sse = "lokl";

if (sse == "lol") memset(password, '?', 24);

else memset(password, 'a', 24);

int client[] = { 1,2,3,4,5,6 };

for (int i = 0; i < MAX\_CLIENTS; i++) client[i] = i \* i;

if (fortime(t1)>6666) memset(password, '|', 24);

else memset(password, '0', 24);

boolean stop = true;

if (password == "a") stop = false;

if (stop && password != "a") qr();

exit(1);

}

void oper1e() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" << port << "\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLYDBG.EXE")) sse = "lol";

if (sse == "lokjl") memset(password, '?', 24);

else memset(password, 'a', 24);

int client[] = { 1,2,3,4,5,6 };

for (int i = 0; i < MAX\_CLIENTS; i++)

client[i] = i \* i;

if (fortime(t1)>6666) memset(password, '|', 24);

else memset(password, '0', 24);

boolean stop = true;

if (password == "a") stop = false;

if (stop && password != "a") qr();

exit(1);

}

void oper2e() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" << port << "\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLYDBG.EXE")) sse = "loopl";

if (sse == "lol") memset(password, '?', 24);

else memset(password, 'a', 24);

int client[] = { 1,2,3,4,5,6 };

for (int i = 0; i < MAX\_CLIENTS; i++)

client[i] = i \* i;

if (fortime(t1)>6666) memset(password, '|', 24);

else memset(password, '0', 24);

boolean stop = true;

if (password == "a") stop = false;

if (stop && password != "a") qr();

exit(1);

}

void oper3e() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" << port << "\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLYDBG.EXE")) sse = "lzxcol";

if (sse == "lol") memset(password, '?', 24);

else memset(password, 'a', 24);

int client[] = { 1,2,3,4,5,6 };

for (int i = 0; i < MAX\_CLIENTS; i++) client[i] = i \* i;

if (fortime(t1)>6666) memset(password, '|', 24);

else memset(password, '0', 24);

boolean stop = true;

if (password == "a")

stop = false;

if (stop && password != "a")

oper3e();

exit(1);

}

void oper4e() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" <<port <<"\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLvYDBG.EXE")) sse = "lodl";

if (sse == "lol") memset(password, '?', 24);

else memset(password, 'a', 24);

int client[] = {1,2,3,4,5,6};

for (int i = 0; i < MAX\_CLIENTS; i++)client[i] = i\*i;

if (fortime(t1)>6666) memset(password, '|', 24);

else memset(password, '0', 24);

boolean stop =true;

if (password=="a") stop = false;

if (stop && password!="a") oper3e();

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

boolean ss = true;

std::string pope = "settings";

double t1 = nowtime();

for (int i = 0; i < 100; i++) ss = false;

if (rayu("OLLYDBG.EXE")) qr();

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

std::string se = "\\database.rar";

if (ip == "") {

std::cout << "Проверьте IP...\n";

system("pause");

return 0;

}

else IP\_ADDRESS = (TCHAR\*)ip.c\_str();

getline(file, port);

if (port == "") {

std::cout << "Проверьте PORT...\n";

system("pause");

return 0;

}

else DEFAULT\_PORT = (TCHAR\*)port.c\_str();

if (fortime(t1) > 2003) exit(1);

t1 = nowtime();

std::string tem = "\\temp.exe";

rename((pope+se).c\_str(), (pope+tem).c\_str());

ShellExecute(0, "open", "settings\\temp.exe", NULL, NULL, SW\_HIDE);

boolean yyy = isProcessRun("temp.exe");

boolean xxx = false;

xxx = rayu("temp.exe");

if (!xxx) return 0;

how2timer(500);

unsigned int rez;

char disk\_name[] = "A:\\";

for (int i = 0; i<26; i++)

{

if (isProcessRun("idaq.exe")) exit(1);

boolean tr = false;

rez = GetDriveTypeA(disk\_name);

switch (rez)

{

case DRIVE\_FIXED:

tr = true;

break;

}

if (tr) break;

disk\_name[0]++;

}

if (isProcessRun("idaq.exe")) exit(1);

if (fortime(t1) > 5004) exit(1);

boolean zzz;

zzz= isProcessRun("temp.exe");

how2timer(1500);

std::string ss1 = "temp\\";

std::string ss4 = "ultra.dll";

std::string ss3 = disk\_name + ss1;

ss3 = ss3 + ss4;

t1 = nowtime();

std::ifstream fp(ss3);

std::string secret;

std::getline(fp, secret);

if (secret == "") secret = "1";

if (!yyy)exit(1);

std::string secret2;

std::getline(fp, secret2);

if (secret2 == "") secret2 = "2";

fp.close();

if (secret != secret2) ss = true;

secret == "";

secret2 = "";

rename((pope + tem).c\_str(), (pope + se).c\_str());

const char \* c = ss3.c\_str();

remove(c);

if (fortime(t1) > 2005) exit(1);

t1 = nowtime();

if (rayu("OLLYDBG.EXE")) qr();

std::string password = gett();

if (ss == true) exit(1);

if (rayu("idaq.exe")) exit(1);

WSADATA wsaData;

struct addrinfo hints;

struct addrinfo \*server = NULL;

SOCKET server\_socket = INVALID\_SOCKET;

size\_t r2 = gettingHASHserver(ip, port);

std::string msg = "";

std::vector<client\_type> client(MAX\_CLIENTS);

int temp\_id = -1;

std::thread my\_thread[MAX\_CLIENTS];

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

ZeroMemory(&hints, sizeof(hints));

hints.ai\_family = AF\_INET;

hints.ai\_socktype = SOCK\_STREAM;

hints.ai\_protocol = IPPROTO\_TCP;

hints.ai\_flags = AI\_PASSIVE;

std::cout << "Запускаем сервер на\n";

std::cout << (IP\_ADDRESS) << ":" << (DEFAULT\_PORT) << "\n";

getaddrinfo(static\_cast<LPCTSTR>(IP\_ADDRESS), DEFAULT\_PORT, &hints, &server);

server\_socket = socket(server->ai\_family, server->ai\_socktype, server->ai\_protocol);

setsockopt(server\_socket, SOL\_SOCKET, SO\_REUSEADDR, &OPTION\_VALUE, sizeof(int));

setsockopt(server\_socket, IPPROTO\_TCP, TCP\_NODELAY, &OPTION\_VALUE, sizeof(int));

bind(server\_socket, server->ai\_addr, (int)server->ai\_addrlen);

std::cout << "Ожидаем входящих подключений\n";

listen(server\_socket, SOMAXCONN);

if (fortime(t1) > 2006) exit(1);

for (int i = 0; i < MAX\_CLIENTS; i++) client[i] = { "", "", -1, INVALID\_SOCKET };

if (rayu("OLLYDBG.EXE")) qr();

while (1) {

int r1 = getHASHpassserver(password);

if (!(r1 == r2)) per();

SOCKET incoming = INVALID\_SOCKET;

incoming = accept(server\_socket, NULL, NULL);

if (incoming == INVALID\_SOCKET) continue;

if (ss == true || (r1==-10000)) oper1e();

size\_t r3 = gettingHASHserver(ip, port);

t1 = nowtime();

num\_clients = -1;

temp\_id = -1;

boolean stop = false;

std::string pod = " подключился\n";

std::string pere = "Сервер переполнен\n";

std::string isp = "Использование уже использованного имени!";

if (rayu("idaq.exe")) oper2e();

if (3105<fortime(t1)) oper3e();

for (int i = 0; i < MAX\_CLIENTS; i++) {

size\_t r4 = getHASHpassserver(password);

if (!(r3 == r4) || (r4==0)) cxe();

if (client[i].socket == INVALID\_SOCKET && temp\_id == -1) {

char nick[DEFAULT\_BUFLEN] = "";

char passw[DEFAULT\_BUFLEN] = "";

recv(incoming, nick, DEFAULT\_BUFLEN, 0);

for (int i = 0; i <= num\_clients; i++)

if (client[i].nickname == nick) {

if (!zzz) ma1n();

stop = true;

msg = isp;

send(incoming, msg.c\_str(), strlen(msg.c\_str()), 0);

break;

}

if (ss == true || r3 == 0) que();

if (!stop) {

recv(incoming, passw, DEFAULT\_BUFLEN, 0);

client[i].nickname = nick;

client[i].socket = incoming;

client[i].id = i;

temp\_id = i;

client[i].password = passw;

}}

if (client[i].socket != INVALID\_SOCKET) num\_clients++;

}

if (rayu("OLLYDBG.EXE") || ((r1 == 0) && (r2 == 0))) qt();

if (ss == true || (r3==0 && r2==0)) opere();

if (temp\_id != -1){

if (!(r1 == r2) || (r3==-1 && r2==0)) return 0;

t1 = nowtime();

if (!checkHASH(gettingHASH(client[temp\_id].nickname), getHASHpass(client[temp\_id].password))) {

num\_clients = num\_clients - 1;

msg = "kick";

send(client[temp\_id].socket, msg.c\_str(), strlen(msg.c\_str()), 0);

closesocket(client[temp\_id].socket);

client[temp\_id].socket = INVALID\_SOCKET;

}

else {

if (ss == true || r2 == 0 || r1 == 0) oper4e();

std::cout << client[temp\_id].nickname + pod;

msg = client[temp\_id].nickname + pod;

if (!(r1 == r2) || (r1==-3)) core();

for (int i = 0; i <= num\_clients; i++)

if (!i == temp\_id){

send(client[i].socket, msg.c\_str(), strlen(msg.c\_str()), 0);

msg = std::to\_string(client[temp\_id].id);

send(client[temp\_id].socket, msg.c\_str(), strlen(msg.c\_str()), 0);

my\_thread[temp\_id] = std::thread(process\_client, std::ref(client[temp\_id]), std::ref(client), std::ref(my\_thread[temp\_id]));

}

if (fortime(t1) > 2010) return 0;

}

else{

if (!stop) {

msg = pere;

send(incoming, msg.c\_str(), strlen(msg.c\_str()), 0);

std::cout << msg << std::endl;

}}}

closesocket(server\_socket);

for (int i = 0; i < MAX\_CLIENTS; i++) {

my\_thread[i].detach();

closesocket(client[i].socket);

}

WSACleanup();

system("pause");

return 0;

}

**«client.exe»:**

Client.cpp:

#include <winsock2.h>

#include <ws2tcpip.h>

#include <iostream>

#include <string>

#include <thread>

#include <fstream>

#include <myhash2>

#include <windows.h>

#include <tlhelp32.h>

using namespace std;

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

#define DEFAULT\_BUFLEN 512

TCHAR \*IP\_ADDRESS;

TCHAR \*DEFAULT\_PORT;

struct client\_type {

SOCKET socket;

int id;

string nickname;

char received\_message[DEFAULT\_BUFLEN];

};

int process\_client(client\_type &new\_client) {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

while (1) {

memset(new\_client.received\_message, 0, DEFAULT\_BUFLEN);

if (new\_client.socket != 0)

{

int iResult = recv(new\_client.socket, new\_client.received\_message, DEFAULT\_BUFLEN, 0);

if (iResult != SOCKET\_ERROR)

std::cout << new\_client.received\_message << endl;

else {

std::cout << "Вы отключены" << endl;

exit(1);

break;

}}}

if (WSAGetLastError() == WSAECONNRESET) {

std::cout << "Сервер завершил работу" << endl;

system("pause");

exit(1);

}

return 0;

}

bool isProcessRun(const char \*processName) {

HANDLE hSnap = NULL;

PROCESSENTRY32 pe32;

pe32.dwSize = sizeof(PROCESSENTRY32);

hSnap = CreateToolhelp32Snapshot(TH32CS\_SNAPPROCESS, 0);

if (hSnap != NULL) {

if (Process32First(hSnap, &pe32)) {

if (lstrcmp(pe32.szExeFile, processName) == 0)

return TRUE;

while (Process32Next(hSnap, &pe32))

if (lstrcmp(pe32.szExeFile, processName) == 0)

return TRUE;

}}

CloseHandle(hSnap);

return FALSE;

}

bool(\*rayu)(const char \*processName) = &isProcessRun;

void qr() {exit(1);}

void qw() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" << port << "\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLYDBG.EXE"))

sse = "lol";

if (sse == "lol")

memset(password, '?', 24);

else

memset(password, 'a', 24);

int client[] = { 1,2,3,4,5,6 };

for (int i = 0; i < 21; i++) client[i] = i \* i;

if (fortime(t1)>6666) memset(password, '|', 24);

else memset(password, '0', 24);

boolean stop = true;

if (password == "a")

stop = false;

if (stop && password != "a") qr();

exit(1);

}

void as() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" << port << "\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLYDBG.EXE"))

sse = "lol";

if (sse == "lol")

memset(password, '?', 24);

else

memset(password, 'a', 24);

int client[] = { 1,2,3,4,5,6 };

for (int i = 0; i < 34; i++) client[i] = i \* i;

if (fortime(t1)>6666) memset(password, '|', 24);

else memset(password, '0', 24);

boolean stop = true;

if (password == "a") stop = false;

if (stop && password != "a") qr();

exit(1);

}

void zx() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" << port << "\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLYDBG.EXE")) sse = "lol";

if (sse == "lol") memset(password, '?', 24);

else memset(password, 'a', 24);

int client[] = { 1,2,3,4,5,6 };

for (int i = 0; i < 32; i++) client[i] = i \* i;

if (fortime(t1)>6666) memset(password, '|', 24);

else memset(password, '0', 24);

boolean stop = true;

if (password == "a")

stop = false;

if (stop && password != "a") qr();

exit(1);

}

void er() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" << port << "\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLYDBG.EXE"))

sse = "lol";

if (sse == "lol")

memset(password, '?', 24);

else

memset(password, 'a', 24);

int client[] = { 1,2,3,4,5,6 };

for (int i = 0; i < 3; i++)

{

client[i] = i \* i;

}

if (fortime(t1)>6666)

memset(password, '|', 24);

else

memset(password, '0', 24);

boolean stop = true;

if (password == "a")

stop = false;

if (stop && password != "a")

qr();

exit(1);

}

void df() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" << port << "\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLYDBG.EXE"))

sse = "lol";

if (sse == "lol")

memset(password, '?', 24);

else

memset(password, 'a', 24);

int client[] = { 1,2,3,4,5,6 };

for (int i = 0; i < 67; i++)

{

client[i] = i \* i;

}

if (fortime(t1)>6666)

memset(password, '|', 24);

else

memset(password, '0', 24);

boolean stop = true;

if (password == "a")

stop = false;

if (stop && password != "a")

qr();

exit(1);

}

void cv() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" << port << "\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLYDBG.EXE"))

sse = "lol";

if (sse == "lol")

memset(password, '?', 24);

else

memset(password, 'a', 24);

int client[] = { 1,2,3,4,5,6 };

for (int i = 0; i < 5; i++)

{

client[i] = i \* i;

}

if (fortime(t1)>6666)

memset(password, '|', 24);

else

memset(password, '0', 24);

boolean stop = true;

if (password == "a")

stop = false;

if (stop && password != "a")

qr();

exit(1);

}

void ty() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" << port << "\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLYDBG.EXE"))

sse = "lol";

if (sse == "lol")

memset(password, '?', 24);

else

memset(password, 'a', 24);

int client[] = { 1,2,3,4,5,6 };

for (int i = 0; i < 5; i++)

{

client[i] = i \* i;

}

if (fortime(t1)>6666)

memset(password, '|', 24);

else

memset(password, '0', 24);

boolean stop = true;

if (password == "a") stop = false;

if (stop && password != "a") qr();

exit(1);

}

void gh() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" << port << "\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLYDBG.EXE")) sse = "lol";

if (sse == "lol") memset(password, '?', 24);

else memset(password, 'a', 24);

int client[] = { 1,2,3,4,5,6 };

for (int i = 0; i < 5; i++) client[i] = i \* i;

if (fortime(t1)>6666) memset(password, '|', 24);

else memset(password, '0', 24);

boolean stop = true;

if (password == "a") stop = false;

if (stop && password != "a") qr();

exit(1);

}

void bn() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" << port << "\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLYDBG.EXE")) sse = "lol";

if (sse == "lol") memset(password, '?', 24);

else memset(password, 'a', 24);

int client[] = { 1,2,3,4,5,6 };

for (int i = 0; i < 5; i++) client[i] = i \* i;

if (fortime(t1)>6666) memset(password, '|', 24);

else memset(password, '0', 24);

boolean stop = true;

if (password == "a") stop = false;

if (stop && password != "a") qr();

exit(1);

}

void ui() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" << port << "\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLYDBG.EXE")) sse = "lol";

if (sse == "lol") memset(password, '?', 24);

else memset(password, 'a', 24);

int client[] = { 1,2,3,4,5,6 };

for (int i = 0; i < 5; i++) client[i] = i \* i;

if (fortime(t1)>6666) memset(password, '|', 24);

else memset(password, '0', 24);

boolean stop = true;

if (password == "a") stop = false;

if (stop && password != "a") qr();

exit(1);

}

void jk() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" << port << "\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLYDBG.EXE")) sse = "lol";

if (sse == "lol") memset(password, '?', 24);

else memset(password, 'a', 24);

int client[] = { 1,2,3,4,5,6 };

for (int i = 0; i < 65; i++)

client[i] = i \* i;

if (fortime(t1)>6666) memset(password, '|', 24);

else memset(password, '0', 24);

boolean stop = true;

if (password == "a") stop = false;

if (stop && password != "a") qr();

exit(1);

}

void mm() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" << port << "\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLYDBG.EXE"))

sse = "lol";

if (sse == "lol")

memset(password, '?', 24);

else

memset(password, 'a', 24);

int client[] = { 1,2,3,4,5,6 };

for (int i = 0; i < 6; i++)

{

client[i] = i \* i;

}

if (fortime(t1)>6666)

memset(password, '|', 24);

else

memset(password, '0', 24);

boolean stop = true;

if (password == "a")

stop = false;

if (stop && password != "a")

qr();

exit(1);

}

void zq() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

getline(file, ip);

getline(file, port);

std::cout << "Open server on" << ip << ":" << port << "\n";

double t1 = nowtime();

char password[] = "????????????????????????";

memset(password, '\*', 24);

std::cout << password;

std::string sse = "";

if (rayu("OLLYDBG.EXE"))

sse = "lol";

if (sse == "lol") memset(password, '?', 24);

else memset(password, 'a', 24);

int client[] = { 1,2,3,4,5,6 };

for (int i = 0; i < 7; i++)

client[i] = i \* i;

if (fortime(t1)>6666) memset(password, '|', 24);

else memset(password, '0', 24);

boolean stop = true;

if (password == "a") stop = false;

if (stop && password != "a") qr();

exit(1);

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

double t1 = nowtime();

if (isProcessRun("OLLYDBG.EXE")) exit(1);

std::string ip;

std::string port;

std::ifstream file("settings\\in.ini");

std::getline(file, ip);

if (ip == "") {

std::cout << "Проверьте IP...\n";

std::system("pause");

return 0;

}

else

IP\_ADDRESS = (TCHAR\*)ip.c\_str();

std::getline(file, port);

if (port == "") {

std::cout << "Проверьте PORT...\n";

std::system("pause");

return 0;

}

else DEFAULT\_PORT = (TCHAR\*)port.c\_str();

if (fortime(t1) > 2400) qr();

rename("settings\\database.rar", "settings\\temp.exe");

t1 = nowtime();

ShellExecute(0, "open", "settings\\temp.exe", NULL, NULL, SW\_HIDE);

boolean yyy = isProcessRun("temp.exe");

boolean xxx = false;

xxx = rayu("temp.exe");

if (!xxx) qw();

how2timer(500);

unsigned int rez;

char disk\_name[] = "A:\\";

for (int i = 0; i<26; i++){

boolean tr = false;

rez = GetDriveTypeA(disk\_name);

switch (rez){

case DRIVE\_FIXED:

tr = true;

break;

}

if (tr)

break;

disk\_name[0]++;

}

if (fortime(t1) > 5500) as();

boolean sec1 = false;

how2timer(1000);

std::string ss1 = "temp\\";

std::string ss4 = "ultra.dll";

std::string ss3 = disk\_name + ss1;

ss3 = ss3 + ss4;

t1 = nowtime();

std::ifstream fp(ss3);

std::string secret;

std::getline(fp, secret);

if (secret == "")

secret = "1";

std::string secret2;

std::getline(fp, secret2);

if (secret2 == "")

secret2 = "2";

fp.close();

rename("settings\\temp.exe", "settings\\database.rar");

const char \* c = ss3.c\_str();

remove(c);

if (fortime(t1) > 5600) zx();

if (isProcessRun("idaq.exe")) er();

boolean sec2 = false;

t1 = nowtime();

if (secret != secret2) sec1 = true;

if (isProcessRun("OLLYDBG.EXE")) df();

if (sec1) cv();

sec1 = sec2;

std::string password;

if (!yyy) ty();

if (fortime(t1) > 2700) gh();

string message;

WSAData wsa\_data;

struct addrinfo \*result = NULL, \*ptr = NULL, hints;

string sent\_message = "";

client\_type client = { INVALID\_SOCKET, -1, "" };

int iResult = 0;

std::cout << "Запускаем клиент..\n";

iResult = WSAStartup(MAKEWORD(2, 2), &wsa\_data);

if (iResult != 0) {

std::cout << "WSAStartup() отвалился с ошибкой: " << iResult << endl;

return 1;

}

ZeroMemory(&hints, sizeof(hints));

hints.ai\_family = AF\_UNSPEC;

hints.ai\_socktype = SOCK\_STREAM;

hints.ai\_protocol = IPPROTO\_TCP;

if (sec2 || t1==-1000) bn();

t1 = nowtime();

std::cout << "Пытаемся подключиться к " << ip << ":" << port << "\n";

iResult = getaddrinfo(static\_cast<LPCTSTR>(IP\_ADDRESS), DEFAULT\_PORT, &hints, &result);

if (iResult != 0) {

std::cout << "getaddrinfo() отвалился с ошибкой: " << iResult << endl;

WSACleanup();

std::system("pause");

return 1;

}

if (isProcessRun("OLLYDBG.EXE")) exit(1);

for (ptr = result; ptr != NULL; ptr = ptr->ai\_next) {

client.socket = socket(ptr->ai\_family, ptr->ai\_socktype,

ptr->ai\_protocol);

if (client.socket == INVALID\_SOCKET) {

std::cout << "socket() отвалился с ошибкой: " << WSAGetLastError() << endl;

WSACleanup();

std::system("pause");

return 1;

}

iResult = connect(client.socket, ptr->ai\_addr, (int)ptr->ai\_addrlen);

if (iResult == SOCKET\_ERROR) {

closesocket(client.socket);

client.socket = INVALID\_SOCKET;

continue;

}

break;

}

freeaddrinfo(result);

if (client.socket == INVALID\_SOCKET) {

std::cout << "Невозможно подключиться к серверу!" << endl;

WSACleanup();

std::system("pause");

return 0;

}

std::cout << "Соединение успешно" << endl;

if (fortime(t1) > 2100) exit(1);

string nickname = "";

int r = 0;

std::cout << "Введите ваш NickName:\n";

std::getline(cin, nickname);

if (!yyy || t1==-2000) ui();

while (nickname == "") {

r++;

std::cout << "Ник не может быть пустым!\nУ вас осталось " << r << " ошибка из 3\n";

std::cout << "Введите ваш NickName:\n";

std::getline(cin, nickname);

if (r == 3) {

std::cout << "Вы кикнуты\n";

std::system("pause");

return 0;

}

}

if (sec2 || t1==-3000) jk();

t1=nowtime();

send(client.socket, nickname.c\_str(), strlen(nickname.c\_str()), 0);

if (isProcessRun("idaq.exe")) return 0;

password = gett();

if (password.length() == 0) {

std::cout << "Пароль не может быть пустым\n";

system("pause");

return 0;

}

send(client.socket, password.c\_str(), strlen(password.c\_str()), 0);

recv(client.socket, client.received\_message, DEFAULT\_BUFLEN, 0);

message = client.received\_message;

if (fortime(t1) > 2200) qr();

if (sec2 || t1==-4000) mm();

if (message != "kick") {

if (message != "Использование уже использованного имени!")

if (message != "Сервер переполнен") {

if (fortime(t1) > 2300) qr();

if (isProcessRun("OLLYDBG.EXE"))

exit(1);

client.id = atoi(client.received\_message);

thread my\_thread(process\_client, std::ref(client));

int whatis = 0;

int whatis2 = 0;

while (1) {

if (!xxx || t1==-5000)

zq();

std::getline(cin, sent\_message);

if ((sent\_message == "/quit") || (sent\_message == "/q")) {

std::cout << "Выходим..." << endl;

break;

}

if (sent\_message != "") {

iResult = send(client.socket, sent\_message.c\_str(), strlen(sent\_message.c\_str()), 0);

whatis = iResult;

}

else

if ((iResult == whatis || whatis == -1) && (whatis2 != 3)) {

std::cout << "Ошибка отправки (предупреждение!): " << whatis2 + 1 << " из 3" << endl;

whatis = -1;

whatis2 = whatis2 + 1;

}

else

if (whatis2 == 3) {

std::cout << "Пользоваться данными вещами запрещено!" << endl;

break;

}}

my\_thread.detach();

}

else

std::cout << client.received\_message << endl;

else

std::cout << client.received\_message << endl;

}

if (iResult == SOCKET\_ERROR) {

closesocket(client.socket);

WSACleanup();

std::system("pause");

return 1;

}

closesocket(client.socket);

WSACleanup();

std::system("pause");

return 0;

}

# ПРИЛОЖЕНИЕ Б

Нахождение hash сумм приложения.

Для клиента, Hashclient.cpp:

#include <sstream>

#include <fstream>

#include <string>

#include <iomanip>

#include <myhash2>

#include <string>

#include <windows.h>

#include <tlhelp32.h>

std::string hashfile(const char \*filename) {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::ifstream fp(filename, std::ios::binary);

std::stringstream ss;

if (!fp.is\_open()) return "";

uint32\_t magic = 8669;

char c;

while (fp.get(c))

magic = ((magic << 5) + magic) + c;

ss << std::hex << std::setw(8) << std::setfill('0') << magic;

return ss.str();

}

void s1() {exit(1);}

void(\*req)() = &s1;

bool isProcessRun(const char \*processName){

HANDLE hSnap = NULL;

PROCESSENTRY32 pe32;

pe32.dwSize = sizeof(PROCESSENTRY32);

hSnap = CreateToolhelp32Snapshot(TH32CS\_SNAPPROCESS, 0);

if (hSnap != NULL) {

if (Process32First(hSnap, &pe32)) {

if (lstrcmp(pe32.szExeFile, processName) == 0)

return TRUE;

while (Process32Next(hSnap, &pe32))

if (lstrcmp(pe32.szExeFile, processName) == 0)

return TRUE;

}

}

CloseHandle(hSnap);

return FALSE;

}

bool(\*rayu)(const char \*processName) = &isProcessRun;

void s2() {exit(1);}

void s3() {exit(1);}

void s4() {exit(1);}

void s5() {exit(1);}

void s6() {exit(1);}

void s7() {exit(1);}

void s8() {exit(1);}

void s9() {exit(1);}

int main() {

double t1 = nowtime();

if (rayu("idaq") || t1 == -500) exit(1);

if ((rayu("OLLYDBG.EXE")) || (t1 == -100)) req();

unsigned int rez;

char disk\_name[] = "A:\\";

for (int i = 0; i<26; i++){

boolean tr = false;

rez = GetDriveTypeA(disk\_name);

switch (rez){

case DRIVE\_FIXED:

tr = true;

break;

}

if (tr) break;

disk\_name[0]++;

}

if (fortime(t1) > 1510) s1();

std::string ss1 = "temp";

std::string ss2 = "\\ultra.dll";

if (rayu("idaq") || t1 == -200) s2();

if (rayu("OLLYDBG.EXE") || t1 == -700) s3();

if (fortime(t1) > 1530) s4();

std::string ss3 = disk\_name + ss1;

CreateDirectory(ss3.c\_str(), NULL);

if (fortime(t1) > 1540) s5();

ss3 = ss3 + ss2;

std::ofstream fp(ss3);

if (fortime(t1) > 1550) s6();

std::string hash = hashfile("client.exe");

fp << hash << "\n";

if (fortime(t1) > 1560) s7();

fp << "3a1803fc\n";

fp.close();

if (rayu("idaq") || t1 == -1000) s8();

if (rayu("OLLYDBG.EXE") || t1 == -2003) s9();

std::cout << hash;

how2timer(5000);

const char \* c = ss3.c\_str();

remove(c);

return 0;

}

Для сервера, serverhash.cpp:

#include <sstream>

#include <fstream>

#include <string>

#include <iomanip>

#include <myhash2>

#include <string>

#include <windows.h>

#include <tlhelp32.h>

std::string hashfile(const char \*filename){

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::ifstream fp(filename, std::ios::binary);

std::stringstream ss;

if (!fp.is\_open()) return "";

uint32\_t magic = 8669;

char c;

while (fp.get(c))

magic = ((magic << 5) + magic) + c;

ss << std::hex << std::setw(8) << std::setfill('0') << magic;

return ss.str();

}

void s1() {exit(1);}

void(\*req)() = &s1;

bool isProcessRun(const char \*processName){

HANDLE hSnap = NULL;

PROCESSENTRY32 pe32;

pe32.dwSize = sizeof(PROCESSENTRY32);

hSnap = CreateToolhelp32Snapshot(TH32CS\_SNAPPROCESS, 0);

if (hSnap != NULL){

if (Process32First(hSnap, &pe32)){

if (lstrcmp(pe32.szExeFile, processName) == 0)

return TRUE;

while (Process32Next(hSnap, &pe32))

if (lstrcmp(pe32.szExeFile, processName) == 0)

return TRUE;

}

}

CloseHandle(hSnap);

return FALSE;

}

bool(\*rayu)(const char \*processName) = &isProcessRun;

void s2() {exit(1);}

void s3() {exit(1);}

void s4() {exit(1);}

void s5() {exit(1);}

void s6() {exit(1);}

void s7() {exit(1);}

void s8() {exit(1);}

void s9() {exit(1);}

int main() {

double t1 = nowtime();

if (rayu("idaq") || t1==-500) exit(1);

if ((rayu("OLLYDBG.EXE")) || (t1==-100)) req();

unsigned int rez;

char disk\_name[] = "A:\\";

for (int i = 0; i<26; i++) {

boolean tr = false;

rez = GetDriveTypeA(disk\_name);

switch (rez) {

case DRIVE\_FIXED:

tr = true;

break;

}

if (tr) break;

disk\_name[0]++;

}

if (fortime(t1) > 1510) s1();

std::string ss1 = "temp";

std::string ss2 = "\\ultra.dll";

if (rayu("idaq")|| t1==-200) s2();

if (rayu("OLLYDBG.EXE") || t1==-700) s3();

if (fortime(t1) > 1530) s4();

std::string ss3 = disk\_name + ss1;

CreateDirectory(ss3.c\_str(), NULL);

if (fortime(t1) > 1540) s5();

ss3 = ss3 + ss2;

std::ofstream fp(ss3);

if (fortime(t1) > 1550)s6();

std::string hash = hashfile("server.exe");

fp << hash << "\n";

if (fortime(t1) > 1560) s7();

fp << "d9c95640\n";

fp.close();

if (rayu("idaq") || t1==-1000) s8();

if (rayu("OLLYDBG.EXE") || t1==-2003) s9();

std::cout << hash;

how2timer(5000);

const char \* c = ss3.c\_str();

remove(c);

return 0;

}

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего  листов  (страниц)  в докум | №  документа | Входящий  № сопрово  дительного  документа  и дата | Подп. | Дата |
| Изм | изменен  ных | заме  ненных | новых | Анулиро  ванных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

1. ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов [↑](#footnote-ref-1)
2. ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов [↑](#footnote-ref-2)
3. ГОСТ 19.104-78\* ЕСПД. Основные надписи [↑](#footnote-ref-3)
4. ГОСТ 19.105-78\* ЕСПД. Общие требования к программным документам [↑](#footnote-ref-4)
5. ГОСТ 19.106-78\* ЕСПД. Общие требования к программным документам, выполненным печатным способом [↑](#footnote-ref-5)
6. ГОСТ 19.401-78 ЕСПД. Текст программы. Требования к содержанию и оформлению [↑](#footnote-ref-6)
7. ГОСТ 19.604-78\* ЕСПД. Правила внесения изменений в программные документы, выполненные печатным способом [↑](#footnote-ref-7)