Лабораторная работа №2.

Изучение методов использования программных ловушек (хуков).

Цель работы: сформировать практические навыки использования программных ловушек (хуков).

Задачи: изучить принципы использования хуков. Реализоавать функционал программы-шпиона.

Операционные системы семейства Microsoft Windows предлагают встроенные возможности по перехвату функций. Такие ловушки могут использоваться для целей видео/аудио захвата, расширения функционала закрытого ПО, журналирования. В то же время такой мощный механизм может использовать зловредное ПО (трояны, кейлоггеры).

Существует два способа перехвата функций в системе Windows:

1) на основе подмены адреса функции в динамически подключаемых библиотеках
2) на основе внедрения кода безусловного перехода на пользовательскую функцию в заголовке исполняемой программы.

Все остальные способы основываются на той или иной модификации одного из представленных способов.

Для работы с хуками используется вызов системных функций используется системный API.

Существует 14 типов хуков:

1) WH_CALLWNDPROC - хук вызывается при вызове SendMessage.
2) WH_CALLWNDPROCRET - хук вызывается, когда возвращается SendMessage.
3) WH_GETMESSAGE - хук вызывается, когда вызывается GetMessage или PeekMessage.
4) WH_KEYBOARD - хук вызывается, когда GetMessage или PeekMessage получают WM_KEYUP или WM_KEYDOWN из очереди сообщений.

5) WH_MOUSE - хук вызывается, когда GetMessage или PeekMessage получают сообщение от мыши из очереди сообщений.
6) WH_HADRWARE - хук вызывается, когда GetMessage или PeekMessage получают хардварное сообщение, не относящееся к клавиатуре или мыши.
7) WH_MSGFILTER - хук вызывается, когда диалоговое окно, меню или скролбар готовятся к обработке сообщения. Этот хук - локальный. Он создан специально для тех объектов, у которых свой внутренний цикл сообщений.
8) WH_SYSMSGFILTER - то же самое WH_MSGFILTER, но системный.
9) WH_JOURNALRECORD - хук вызывается, когда Windows получает сообщение из очереди хардварных сообщений.
10) WH_JOURNALPLAYBACK - хук вызывается, когда событие затребовывается из очереди хардварных сообщений.
11) WH_SHELL - хук вызывается, когда происходит что-то интересное и связанное с оболочкой, например, когда таскбару нужно перерисовать кнопку.
12) WH_CBN - хук используется специально для CBT.
13) WH_FOREGROUND - такие хуки используются Windows. Обычным приложениям от них пользы немного.
14) WH_DEBUG - хук используется для отладки хук-процедуры.

Установка хука

Для установки хука можно использовать следующую WinAPI функцию (C++):

```
HHOOK WINAPI SetWindowsHookEx(
  _In_ int       idHook,
  _In_ HOOKPROC  lpfn,
  _In_ HINSTANCE hMod,
  _In_ DWORD     dwThreadId
  );
```

где

idHook – тип процедуры для ловушки (из описаных выше)

lpfn — указатель на функцию, обрабатывающую перехваченные сообщения

hMod — дескриптор экземпляра приложения, содержащий обрабатывающую функцию

dwThreadId — идентификатор потока, сообщения которого мы хотим перехватывать. Нужно установить данный параметр в 0, чтобы перехватывать сообщения всех потоков

Возвращаемое значение – дескриптор установленной ловушки.

Прототип функции, обрабатывающей значение нашей ловушки выглядит следущим образом:

```
LRESULT CALLBACK LowLevelKeyboardProc(
  _In_ int    nCode,
  _In_ WPARAM wParam,
  _In_ LPARAM lParam
);
```

где

nCode должен быть равен HC_ACTION, иначе сообщение предоставляется другому процессу.

wParam — это одно из следующих значений WM_KEYDOWN, WM_KEYUP, WM_SYSKEYDOWN, или WM_SYSKEYUP.

lParam — указатель на структуру KBDLLHOOKSTRUCT

Далее функция должна вернуть значение функции CallNextHookEx, в противном же случае, следующая обрабатывающий событие функция, может получить неверные параметры сообщения.

структура KBDLLHOOKSTRUCT

```
typedef struct tagKBDLLHOOKSTRUCT {
  DWORD     vkCode;
  DWORD     scanCode;
  DWORD     flags;
  DWORD     time;
  ULONG_PTR dwExtraInfo;
} KBDLLHOOKSTRUCT, *PKBDLLHOOKSTRUCT, *LPKBDLLHOOKSTRUCT;
```

где нас интересуют только 2 параметра: vkCode(виртуальный код) и scanCode нажатой клавиши.

vkCode – виртуальный код клавиши (от 1 до 254)

scanCode – hardware скан-код клавиши

Для того, тобы транслирвать виртуальный код клавиши в читаемый символ, потребуется функция ToAsciiEx.

```
int WINAPI ToAsciiEx(
  _In_           UINT   uVirtKey,
  _In_           UINT   uScanCode,
  _In_opt_ const BYTE   *lpKeyState,
  _Out_          LPWORD lpChar,
  _In_           UINT   uFlags,
  _In_opt_       HKL    dwhkl
);
```

uVirtKey — виртуальный код клавиши.

uScanCode - скан коды клавиши

lpKeyState — состояние клавиатуры, проверяет какие клавиши нажаты/активны (можно определить нажатый CapsLock)

lpChar — указатель на двойное слово, в которое функция запишет символьное представление клавиши.

uFlags — параметр, указывающий на активность меню(ALT).

dwhkl — (опциональный) идентификатор раскладки клавиатуры

Кейлоггер – программный (программно-аппаратный) комплекс, который может перехватывать и запоминать нажатия на клавиатуру. Зачастую кейлоггеры являются частью другие, более сложных вредоносных програм, таких как трояны.

Существуют аппаратные кейлоггеры, но они мало распространены по причине необходимости физического доступа к машине для их использования.

В общем виде, алгоритм работы кейлоггера можно описать следующим образом:

Устанавливается перехватчик событий нажатия на клавиатуру (хук, ловушка) -> принимается значение нажатой клавиши -> транслируется в символ в соответствии с раскладкой -> символ записывается в файл.

Задание.

Разработать программу-keylogger, фиксирующую нажатия клавиатуры пользователем.

Программа должна удовлетворять следующим требованиям:

1) Работать незаметно для пользователя (не отображать окна работы программы
2) Работать только в окне одного процесса (например, браузер Mozilla Firefox, Google Chrome)
3) При наполнении лога размером 10Кб архивировать его
4) Определять символы в зависимости от текущей раскладки (Ru – En)
5) Определять регистр нажатой клавиши (клавиша Shift, клавиша CapsLock)