WHITE PAPER: SYMANTEC SECURITY RESPONSE

IME as a Possible Keylogger

Masaki Suenaga
Symantec Security Response, Tokyo

IME as a Possible Keylogger

Contents

Abstract

This paper outlines a potential method for using an Input Method Editor (IME) as a keylogger. It will discuss how it is possible, using components of Windows multilingual support, to create a file that will capture keystrokes on a target system while using the OS to protect that file from removal or deletion.

Introduction

The Chinese, Japanese and Korean writing systems use thousands of characters: Hanzi (Chinese characters) in Chinese; Kanji (Chinese characters), Hiragana and Katakana in Japanese; Hangeul and Hanja (Chinese characters) in Korean. To represent these characters, each of these languages has its own multi-byte character code sets. On ASCII code-based Windows operating systems such as Windows 95, the double byte character set or DBCS is used, where each two-byte sequence represents one character. While DBCS is no longer commonly used, it is still used on Windows XP if a program does not call Unicode APIs. Starting with Windows 2000, Microsoft's desktop operating systems have primarily used Unicode for cross-compatibility and ease of use.

If a keyboard had thousands of keys, as was once the case with mechanical typewriters, there would be no need to convert multiple keystrokes to a single character. However, most modern keyboards have only around 100 keys. Therefore, we need something to convert keystrokes to characters before being used in an application. This kind of software is called a front-end processor or FEP, and IME is the standard name for FEPs used in Windows environments.

Figure 1 shows some common IME options when the keyboard icon is clicked. The pop-up list shows all the available IMEs or keyboard layouts for a given language.

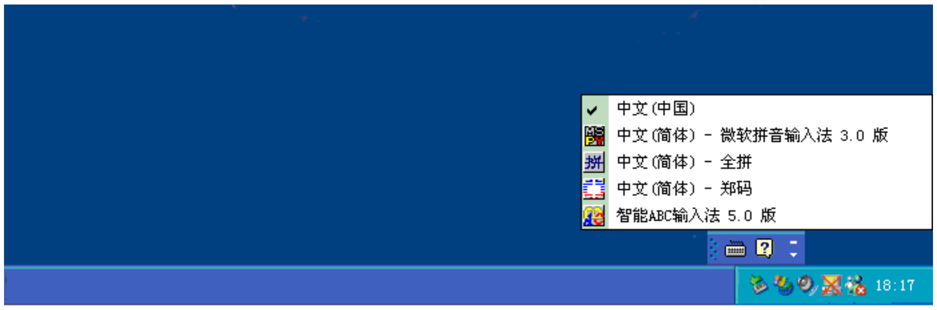


Figure 1: Some common IME options

Figures 2-5 illustrate how a user inputs Chinese characters in Notepad. The IME status bar is shown in the bottom right-hand corner of the Notepad window here, but it can be placed anywhere, and generally is shown either in the bottom right-hand corner of the screen or as part of the Taskbar.

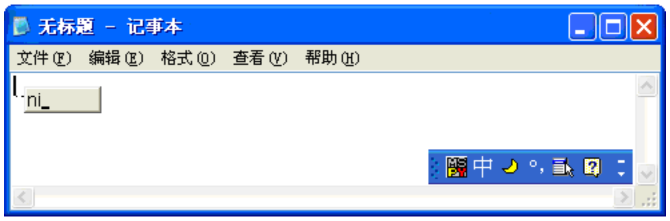


Figure 2: User has entered the composition string ‘ni’

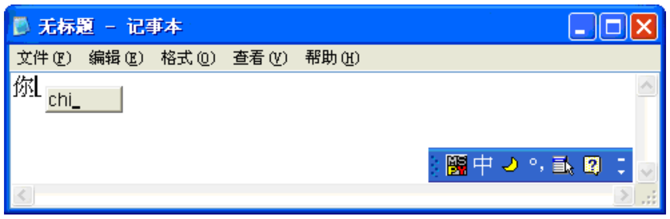


Figure 3: User has entered the composition string ‘nichi’

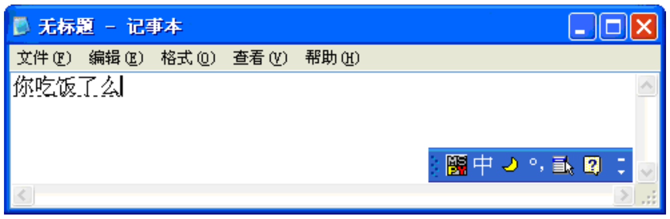


Figure 4: User has entered the phrase ‘nichifanleme’ and pressed Enter

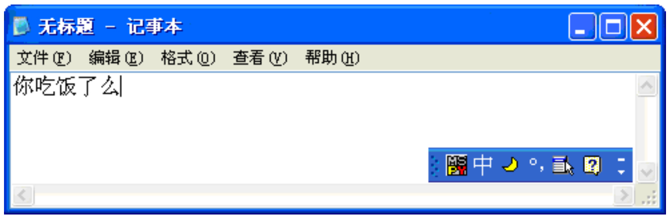


Figure 5: Result string displayed when user presses Enter a second time

In the first screenshot (Figure 2) the user has typed ‘ni’ while IME is ON. The string ‘ni’ here is called the ‘composition string’ in the interface. Next in Figure 3, the user has typed ‘nichi’. You can see that ‘ni’ has disappeared from the little grey box, and the character which is most likely to represent ‘ni’ is underlined with dashes. Both of these strings are called composition strings because they may not match the final text used in the application.

The next screenshot (Figure 4) shows the screen when the user has finished typing the phrase ‘nichifanleme’ and has pressed the Enter key. There is no little grey box, but the five Chinese characters are underlined with dashes. These characters, 10 bytes in DBCS and five words in Unicode, have not been passed to the application yet. At this stage it is still considered a composition string.

Finally, in Figure 5, the user has pressed Enter a second time to confirm the characters in the previous string. This string is called the ‘result string’, since the user has chosen not to alter any of the IME-selected characters for the keystrokes they typed. Each character is sent to the application window with the `WM_IME_CHAR` window message. If the window procedure does not process this message, it will receive a `WM_CHAR` message. So most applications don’t need to be aware of the IME. (Some applications, such as Microsoft Word and Excel are fully aware of IME and display composition strings by themselves.) These complicated tasks are performed by the IME and an Input Method Manager (IMM), which is included in every language version of Windows with multilingual support (2000 and later).

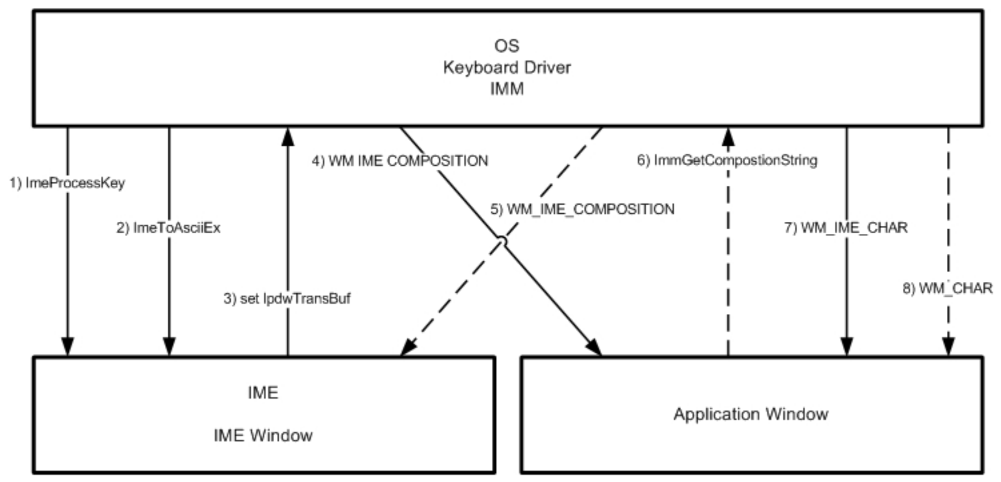


Figure 6: Process diagram for text input process

The process diagram (Figure 6) illustrates how the text input process occurs, and each step is described in detail below:

1) When IMM receives a key stroke from the keyboard driver, it sends the key stroke to the IME through the `ImeProcessKey` entry to ask the IME if the key stroke should be processed by the IME. If the IME returns zero, the key stroke will be processed by the OS and passed to the application as `WM_KEYDOWN` and `WM_KEYUP`, then further processed to `WM_CHAR` or `WM_COMMAND`, and so on. The IME will not receive the key through `ImeToAsciiEx`.

2) If the IME returns a non-zero value in `ImeProcessKey`, the IMM sends the character to the IME again.

3) The IME receives the `lpdwTransBuf` parameter, which will be set by the IME when the process returns from the IME to the IMM. The `lpdwTransBuf` parameter contains information about window messages to be sent to the application. The IME also receives the `hIMC` parameter, which contains composition strings, such as the composition string itself, the result string, and any reading information or clause information, depending on the language. The IME modifies the content of `hIMC` as it processes characters.

4) Any time the IMM receives `lpdwTransBuf` back from the IME, the IMM checks the buffer to see if it contains a message list. Typically it contains the `WM_IME_COMPOSITION` message, which

should be sent whenever the composition string changes. The IMM sends these messages in the buffer to the application window.

5) If the application is not IME-aware, it will not process the `WM_IME_COMPOSITION` message and thus the user will not see the text within the application. In this case, the message is relayed to the corresponding IME UI window, which is always created if an IME is activated. An IME UI window will show the composition string as it is typed.

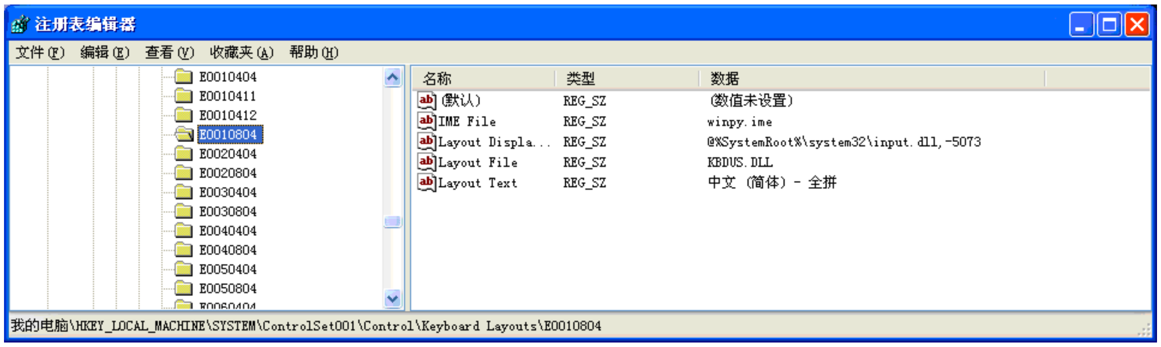
6) If the application is IME-aware, it will process the `WM_IME_COMPOSITION` message. If there is a need to get the contents of composition strings, it calls the `ImmGetCompositionString` API in IMM32.DLL. The `WM_IME_COMPOSITION` message can also notify that the string is determined and the result string has been generated. If the application gets the determined string directly from IMM and pastes the string into its document, it should not call `DefWindowProc` on the `WM_IME_CHAR` message, because further processing will generate the same character twice.

7) If the application is a not IME-aware, it will receive the `WM_IME_CHAR` message. If the application uses the `GetMessageW` API (along with `DispatchMessageW`), it will get one Unicode character in a `WM_IME_CHAR`. If the `GetMessageA` API (along with `DispatchMessageA`) is used, the application receives one DBCS character in the message. If the application does not call `DefWindowProc` on `WM_IME_CHAR`, it will not receive the `WM_CHAR` message later.

8) If the application is not designed to use IME at all, it will get a `WM_CHAR` message as the result string is generated. If `GetMessageW/DispatchMessageW` is used, it receives a Unicode character, which is exactly the same as when getting `WM_IME_CHAR`. If `GetMessageA/DispatchMessageA` are used, it receives two `WM_CHAR` messages for each DBCS character; the higher byte on the first message, the lower on the second.

An IME is a DLL file, typically with the file extension '.ime', and is usually placed in the Windows System folder. IMEs are registered as keyboard layouts in the registry at the following location: HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\ Keyboard Layouts (Figure 7). If a keyboard layout is not related to an IME, the registry will be similar to that shown in Figure 8.

You'll notice in Figure 7 that this IME has an entry called 'IME file' with a value of 'winpy.ime'. When these registry values are set, a user can choose to enable this IME through the Control Panel. Once enabled, the IME will be shown in the list of keyboard layouts that we saw earlier.

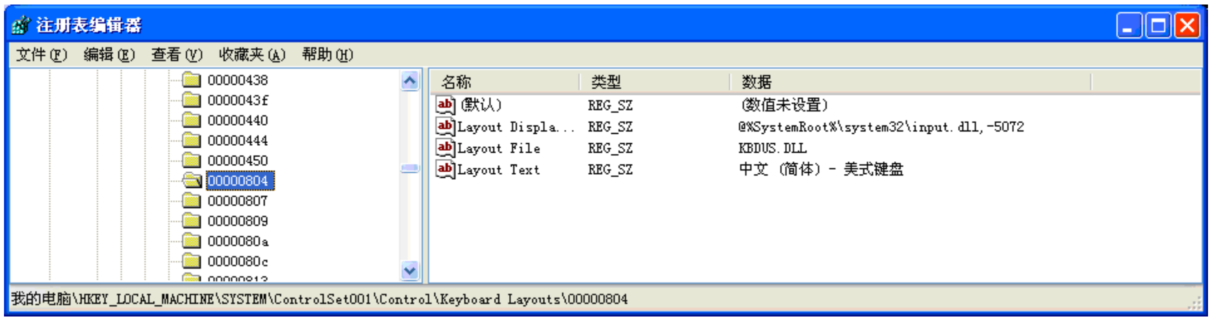


E0010404
E0010411
E0010412
E0010804
E0020404
E0020804
E0030404
E0030804
E0040404
E0040804
E0050404
E0050804

名称	类型	数据
(默认)	REG_SZ	(数值未设置)
IME File	REG_SZ	winpy.ime
Layout Displa...	REG_SZ	@%SystemRoot%\system32\input.dll,-5073
Layout File	REG_SZ	KBDUS.DLL
Layout Text	REG_SZ	中文 (简体) - 全拼

```
我的电脑\HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Keyboard Layouts\E0010804
```

Figure 7: Registry entries for a registered IME



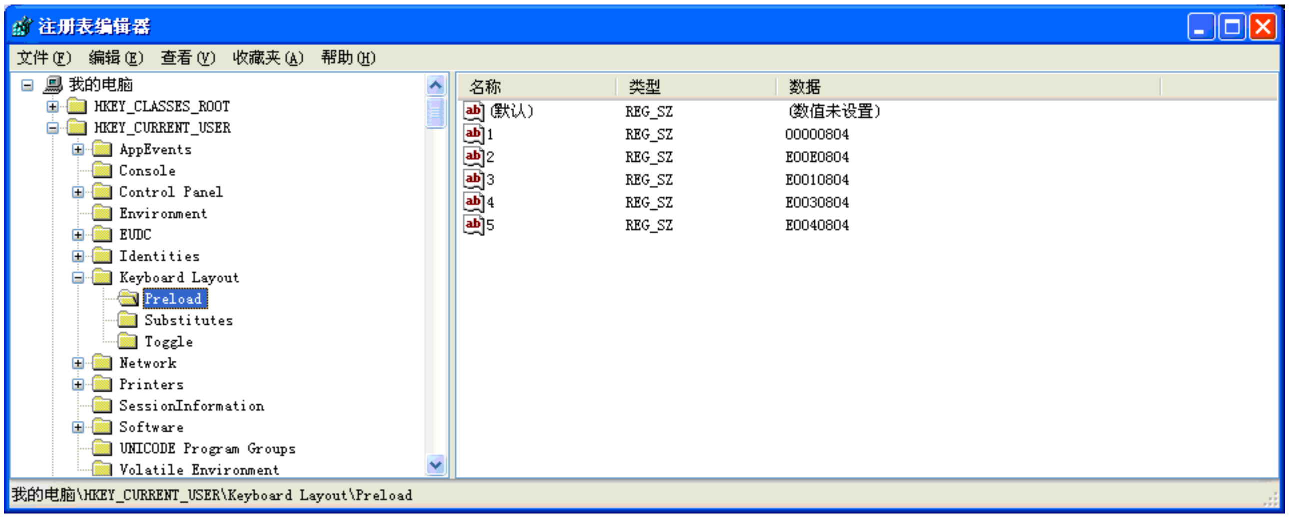
00000438
0000043f
00000440
00000444
00000450
00000804
00000807
00000809
0000080a
0000080c

名称	类型	数据
(默认)	REG_SZ	(数值未设置)
Layout Displa...	REG_SZ	@%SystemRoot%\system32\input.dll,-5072
Layout File	REG_SZ	KBDUS.DLL
Layout Text	REG_SZ	中文 (简体) - 美式键盘

```
我的电脑\HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Keyboard Layouts\00000804
```

Figure 8: Registry entries without a registered IME

Figure 9 shows what the registry looks like when an IME is selected in the Control Panel.



```
我的电脑
  HKEY_CLASSES_ROOT
  HKEY_CURRENT_USER
    AppEvents
    Console
    Control Panel
    Environment
    EUDC
    Identities
    Keyboard Layout
      Preload
      Substitutes
      Toggle
    Network
    Printers
    SessionInformation
    Software
    UNICODE Program Groups
    Volatile Environment
```

名称	类型	数据
(默认)	REG_SZ	(数值未设置)
1	REG_SZ	00000804
2	REG_SZ	E00E0804
3	REG_SZ	E0010804
4	REG_SZ	E0030804
5	REG_SZ	E0040804

```
我的电脑\HKEY_CURRENT_USER\Keyboard Layout\Preload
```

Figure 9: List of keyboard layouts that can be selected by the user

The registry key HKEY_CURRENT_USER\Keyboard Layout\Preload contains the list of keyboard layouts to be selected by the user. The value '1' is the default layout, which is automatically loaded in every process during the user's session. In Japanese and Korean versions of Windows, Microsoft's IME is the default.

An IME Is Loaded In Every Process

Any keyboard layout, including an IME, is loaded in every process. KBDUS.dll, the US keyboard layout DLL, contains only data and has no room to place any extra code. But an IME is a program that has several predetermined entries. No application can reject an IME; an IME acts like a system DLL. Before the instruction enters the entry point of an application, the IME file is loaded, thus DllEntry has been called.

Think of Japanese and Korean OSs. If the genuine Microsoft IME file is replaced with a hacked version using MoveFileEx or another method, the hacked IME will be loaded, even in the System process. This means the hacked IME can run even in the user sessions where the hacked IME is not the default keyboard layout. A removal tool cannot repair or replace the IME files because the correct files differ depending on the OS and the Service Pack installed. If the hacked IME hinders the replacement of the files, it becomes even more difficult to fix the problem.

For those who don't use an IME, a simple IME that does not convert characters can be installed and become the default keyboard layout. There is a slight difference in UI, especially in the language bar, but most users will not understand why it happened unless they have experience using IMEs. The average user would have no idea that the IME was running in his/her English OS. They might even search some commonly known registry load points for the culprit, but would likely find nothing.

To develop a fake IME, which does not convert characters and does not display a window, is far easier than to infect or hack a genuine IME, or than to develop a whole new full functional IME for Chinese, Japanese and Korean users.

An IME Can Be Added

In Korea, there seems to be no need to develop IMEs other than Microsoft's. But in Japan, because many users have become accustomed to their favorite input methods, multiple third-party IMEs are sold.

Figure 10 shows the default display of the IME status bar on Japanese Windows XP. When the keyboard icon is clicked (Figure 11), the list of available IMEs in the current language is shown. If no IME is added, only one IME is displayed (MS-IME2002 here).

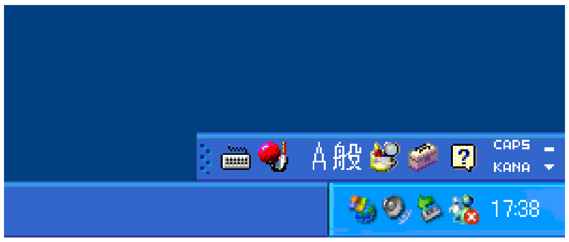


Figure 10: The IME status bar on Japanese Windows XP.

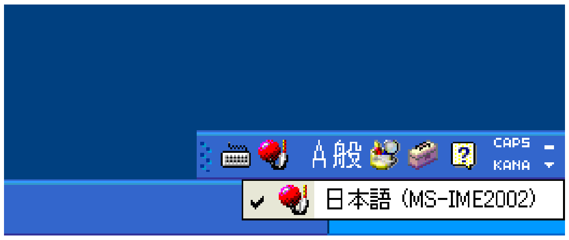


Figure 11: Status bar showing a single IME

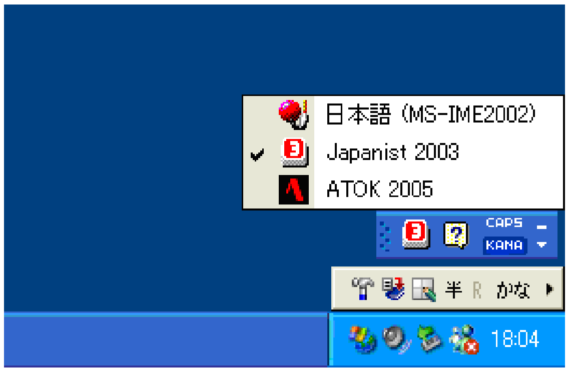
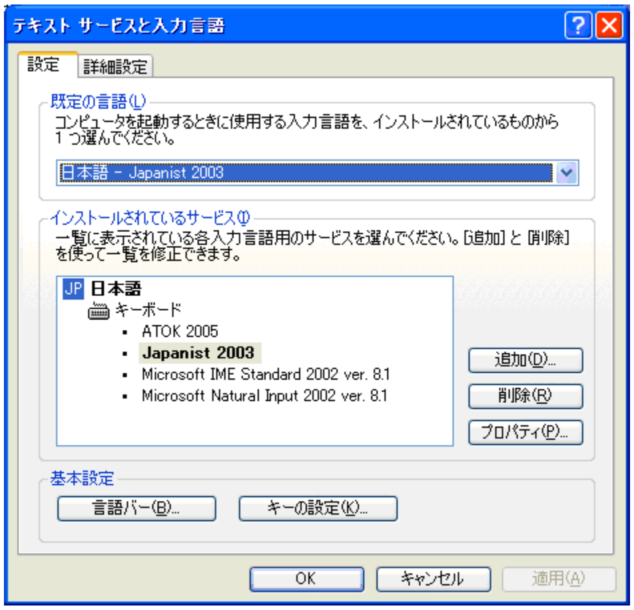


Figure 12: Status bar showing three IMEs

If two IME products are installed on the machine, the list will display (Figure 12) three IMEs and their names. The IME currently in use is marked with a tick (ATOK 2005 and Japanist 2003 are third-party Japanese IME products.)

Figure 13 shows the 'Regional and Language Options' dialog box. The standard language or default language can be selected here. The standard language will be loaded automatically in every process in the session of the user who has selected the language. The user also can remove an IME from the list.


テキスト サービスと入力言語

設定 詳細設定

既定の言語(L)

コンピュータを起動するときに使用する入力言語を、インストールされているものから 1 つ選んでください。

日本語 - Japanist 2003

インストールされているサービス(I)

一覧に表示されている各入力言語用のサービスを選んでください。[追加] と [削除] を使って一覧を修正できます。

JP 日本語
- キーボード
  - ATOK 2005
  - **Japanist 2003**
  - Microsoft IME Standard 2002 ver. 8.1
  - Microsoft Natural Input 2002 ver. 8.1

Figure 13: The Regional and Language Options dialog box

IMEs Capture Every Keystroke Without Hooking

As already discussed, the OS (IMM) always sends any keystrokes to the currently selected IME through `ImeProcessKey` and `ImeToAsciiEx` entry points in the IME file, which is just a DLL with the file extension '.ime'. It generally exports some mandatory entry points that should be called from the IMM.

The simplest way to log keystrokes is to log them in `ImeProcessKey` and return zero. `ImeProcessKey` receives virtual keys, so that if zero is returned, the IMM will no longer interact with the IME for that keystroke. An IME runs in every process and can act as if it is a part of any program. If the IME sends out packets in Internet Explorer to the system or to monitoring tools, it appears as though Internet Explorer has sent them out, just like injecting such a routine into Internet Explorer.

Unlike the injection technique, however, an IME does not have to hook keyboard-related APIs or messages. Existing security tools would not detect such suspicious behavior performed by the IME.

If `ImeProcessKey` returns a non-zero value, a virtual key is eventually passed to `TranslateMessage` by the application window procedure, then it is passed to the `ImeToAsciiEx` entry point to be converted to a character.

IMEs Are Loaded In Safe Mode

It is not surprising that an IME is loaded even in Safe Mode. This means that the standard (default) IME cannot be deleted, unless a user with a different default IME logs on. The `MoveFileEx` API can be used to rename the IME in use. At the next login, the user will see the second IME in the list become his/her default IME.

More Harmful Actions

Genuine non-infected IME files can easily be removed from the computer. But, if an IME is designed maliciously, code could be added, which may lead to the following scenarios:

- The malicious IME may be able to change the default IME of all the users.
- Even if the user changes their default IME, IMEs that have already been selected cannot be changed. The user must re-login or reboot the computer. A program that forcefully changes the IME in use can be developed, but on Windows NT-based computers, the IME module would remain in memory. If the IME runs a thread, it can keep running. And what if the thread checks the standard IME periodically and changes it back again?
- `MoveFileEx` sets some registry values. If a malicious code deletes the values, it will be difficult to delete the IME file.
- An IME is loaded even in 16-bit applications and the command prompt.
- An IME can load WinSock, enabling it to access the Internet.

Details Of The IME Interface

The following are some important entry points, or exported functions, that any IME should have.

DllMain

This is the start address of the IME DLL file. This is called when an IME file is loaded by the IMM during the initialization of an application if the IME is the default, or is loaded when the user selected the IME manually.

ImeInquire (LPIMEINFO lpInfo, LPTSTR lpszUIClass, DWORD dwSystemInfoFlags)

The IMM would call this entry at least once to retrieve the IME's properties. On Windows NT-based computers, this is called only once and the properties are stored in the global system memory.
An IME can set the member of `lpInfo`, among which `fdwProperty` has `IME_PROP_UNICODE` bit

(0x00080000). If this bit is on, all the IME interfaces will become Unicode-based. If it is off, they become ANSI-based. All the interfaces are called from IMM, which would convert between Unicode and ANSI if the application does not match the code system.

An IME should set a string in `lpszUIClass`, such as ‘`MY_IME_UI_MAIN`’. The IMM would automatically create the window of class ‘`MY_IME_UI_MAIN`’ at the time the application creates the first window. IME-related window messages are passed to this IME UI window. This window becomes one of the child windows of the application window. `RegisterWindow` should be called by the IME.

ImeSelect (HIMC hIMC, BOOL bSelected)
This is called when the IME is selected and unselected. If it is a standard IME, this is called after `DllMain` and before the entry point of the application is called. `hIMC` is a handle to Input Context, in which an IME should store data.

ImeSetActiveContext (HIMC hIMC,BOOL fFlag)
This is called when the application is activated and deactivated.

ImeProcessKey (HIMC hIMC,UINT vKey, LPARAM lKeyData, CONST LPBYTE lpbKeyState)
The IMM calls this first to ask the IME whether this key stroke should be processed by the IME. If the IME returns false (zero), the IMM does not send this key to `ImeToAsciiEx`.

If the MSB of `lKeyData` is on, the key is released, otherwise the key is pressed. `lpbKeyState` is a 256-byte matrix indicating whether each `VKEY` is down or up (including CAPS LOCK state, etc.). So `vKey` is just a virtual key and needs the states of `lpbKeyState` to see what character is input.

ImeToAsciiEx (UINT uVKey, UINT uScanCode, CONST LPBYTE lpbKeyState, LPTRANSMSGLIST lpTransBuf, UINT fuState, HIMC hIMC)
If the IME returns true (non-zero) in `ImeProcessKey`, the virtual key is passed to `ImeToAsciiEx`. This entry is the most important for the genuine IME; in this routine the IME converts alphabet, katakana or Hangeul jamo (parts) to Chinese character, hiragana, kanji, Hangeul or hanja. This routine determines the usability and performance of the IME.

An IME should set the contents of `lpTransBuf` in order for a generated character to be input into the application window, otherwise the key stroke will be lost here. A genuine IME would send `WM_IME_STARTCOMPOSITION`, `WM_IME_COMPOSITION` and `WM_IME_ENDCOMPOSITION` sequentially.

A member `hCompStr` in `hIMC` has the composition string; `hCompStr` is a handle to the `COMPOSITIONSTRING` structure. A genuine IME should set members of `dwCompStrLen`, `dwCompStrOffset`, `dwResultStrLen` and `dwResultStrOffset` in `COMPOSITIONSTRING` and (especially `dwCompStrLen` and `dwResultStrLen`) may change at every key stroke.

If `dwCompStrLen` and `dwCompStrOffset` remain unchanged, the application is highly suspicious as an IME. Now `dwCompStrLen` is the length of the characters which are being converted and shown in a special IME UI window, and `dwResultStrLen` is the length of the character string which is determined and input into the application window. Again, if `dwCompStrLen` is never greater than zero, but `dwResultStrLen` is greater than zero, the IME is highly suspect.

Web-Aware?

An IME that consults the web as to some information related to the input character strings could be developed as a product. But this action should only be initiated when the user performs a specific operation. There would be no legitimate purpose or value for sending all the keystrokes to the web. Similarly, there would be no legitimate reason for sending keystrokes input into a specific window.

If an IME always loads a socket library, there may be conflicts with the application. If the user runs a 16-bit version of an email client, the application would not run properly. Therefore (even though the number of users running a 16-bit Internet application is very low), any quality commercial product should avoid this. Listening on a port should be avoided too, since it is hard to tell what port will be opened by the application. So, if you come across an IME that does this, beware, it could very well be malicious.

About the author

Masaki Suenaga joined Symantec Corporation at Symantec Security Response in Tokyo in 2004 as a virus analyst, and has over ten years experience of working in the Japanese software industry. Masaki graduated from Osaka University of Foreign Studies (soon to be merged with Osaka University), where he studied Russian and other languages while finding the time to hand-assemble Z80 machine code as a hobby.

About Symantec
Symantec is the global leader in information security, providing a broad range of software, appliances, and services designed to help individuals, small and mid-sized businesses, and large enterprises secure and manage their IT infrastructure. Symantec's Norton™ brand of products is the worldwide leader in consumer security and problem-solving solutions. Headquartered in Cupertino, California, Symantec has operations in 35 countries. More information is available at www.symantec.com.

Symantec has worldwide operations in 35 countries. For specific country offices and contact numbers, please visit our Web site. For product information in the U.S., call toll-free 1 800 745 6054.

Symantec Corporation
World Headquarters
20330 Stevens Creek Boulevard
Cupertino, CA 95014 USA
408 517 8000
800 721 3934
www.symantec.com