Regional Director Viewpoint

External Inputs Can Be Hazardous

When it comes to developing secure applications, it is important to understand that the only way to ensure 100% security is to not execute the application at all!

Inputs from other users, both known and unknown, pose one of the biggest threats to a running or connected application. In this article, I will show some practical code snippets that you can readily use in your applications to validate or 'clean' the user inputs that you receive.

Filenames in URL

For saving or reading a file, it is common that you prompt the user for a filename. However, in semi-trustworthy scenarios like the Web, it is preferred that you avoid doing this. Instead, your application should automatically decide the filenames. The path and the format for the filename should be configurable from a configuration setting. This way, you prevent users from obtaining access to unwanted files or simply roaming around inside your storage system. For example, you may want to write a generic ASPX index page that is common for all categories in your shopping site. To output specific category titles, you will accept as QueryString in the URL, the path to an XML file and to a XSL file. The ASPX page will then combine both XML and XSL to output the required result. The URL may look like

http://www.venkatarangan.com/shopping.aspx?xml=shampoo.xml&xsl=vr.xsl

Everything seems fine, but if your ASP.NET application is running with elevated- user privileges, then anybody can change your XML filename or XSL filename to access any file in your computer.

In this case, it is better that you only accept the category and the style sheet name. Using the meta data from a configuration file in your application, you should arrive at the XML filename and XSL filename. While doing this, never use the string directly to construct your filenames. Instead, use a hash which contains the list of filenames and a key in the hash to retrieve the final filename for the user input.

Limiting the application scope

In another case, you may want to limit access to files within your application folder. You can use a code-snippet as shown in Figure 1.

Figure 1

```
Try

Dim mappedPath As String =
Request.MapPath(Path2Check,Request.ApplicationPath, False)

Catch ex As Exception  'cross-application mapping attempted

End Try
```

In the code shown in Figure 1, Request.Mappath throws an HTTPException if the requested path is outside the application's directory. This virtually eliminates the possibility of an attacker forcing the application to process a file outside of its directory.

Minimizing Canonicalization Threats

"Canonical" means "simplest or standard form". For example in Windows,

C:\Temp\Orange.txt and C:\Temp\Test\..\Orange.txt

might both be referring to the same file. So if you want to restrict access to a file, you should not be doing so by simple string comparison of the filename. Instead, you should first canonicalize the file name to its standard form and then apply your logic. As an added defense, you should protect all your valuable files by using Windows Access Control Lists (ACL).

You can use the code snippet in Figure 2 to canonicalize a filename/path in Windows.

Figure 2

```
Dim filename As String
Try
    filename = Path.GetFullPath(filename)
    If filename = "C:\boot.ini" Then
        Throw New Exception("No, that's a system file!")
    End If
    Dim sr As StreamReader = New StreamReader(filename)

    outputBox.Text = sr.ReadToEnd()
    sr.Close()
Catch ex As Exception

outputBox.text = "Error: " + ex.Message
End Try
```

SQL Injection

Most web applications accept an input from user and construct dynamic T-SQL to execute a query and then return the results. Here, you have to remember that in T-SQL, you can do nested queries. So while executing Dynamic SQL from user inputs, you might be unknowingly executing a query like this:

```
exec sp_executesql 'select * from customers where Customername='' drop table
deleteme --'
```

Instead of constructing Dynamic SQL, it is recommended that you use stored procedures and parameterized queries available in the .NET framework. An example of the code snippet is shown in Figure 3. As an added defense, the snippet validates the length of the customer name being passed and throws out the exception if it exceeds 40 characters.

Figure 3

```
Private Function GetRowFromName(ByVal customerName As String) As
DataRow
    If customerName.Length > 40 Then
        Throw New Exception("Customer name too long.")
    End If

    'constructing parameterized query
    Dim customerAdapter As SqlDataAdapter = New SqlDataAdapter( _
        "Select CustomerId, CustomerName FROM Customers WHERE " + _
            "CustomerName=@CustomerName",
            ConfigurationSettings.AppSettings ("NorthwindConnec-
tion"))
    Dim customerDataSet As DataSet = New DataSet

    'add parameter to SQLDataAdapter and set its value
    Dim param As SqlParameter =customerAdapter.SelectCommand.Param-
eters.Add( _

        "@CustomerName", SqlDbType.NVarChar, 40)
    param.Value = customerName

    'run query and return the customer record
    customerAdapter.Fill(customerDataSet, "CustomerInfo")
    Dim customersTable As DataTable =customerDataSet.
Tables("CustomerInfo")
    Return customersTable.Rows(0)
End Function
```

If the code is executed and someone tries to input "drop table deleteme -", you will find that the .NET framework will sanitize the value provided for the @CustomerName parameter by automatically replacing the single apostrophe with two apostrophes. This renders the nested query harmless. For better security, RegularExpressionValidator control can be used to restrict inputs to only characters that can appear in CustomerName.

T.N.C. Venkata Rangan
CMD of Vishwak Solutions and
MSDN Regional Director for Chennai.

The Microsoft Journal for Developers

msdn® magazine

India Edition

OFFICE Beyond The Doc

2005·VOL 3

CONTENTS

Publisher's Note
Angela McFeeters page 5

Regional Director Viewpoint
T.N.C. Venkata Rangan page 9

MVP Feature
Saurabh Verma page 11

Inside MSDN
Tim Ewald page 19

Cutting Edge
Dino Esposito page 25

Test Run
Dr. James McCaffrey page 31

ASP.NET
Rob Howard page 57

Smart Tags
Ben Waldron page 63

SQL Server
James Yip page 70

All About Statics
K. Scott Allen page 84

Wicked Code
Jeff Prosise page 91

Web Q&A
Edited by Nancy Michell page 97

msdn

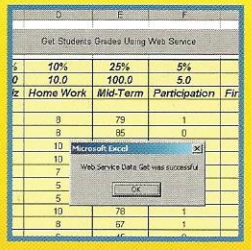
India INR60

8 885002 920011 >

Office

John R. Durant page 38

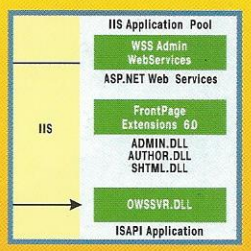
Relive The Moment By Searching Your IM Logs With Custom Research Services



Excel

Alok Mehta page 44

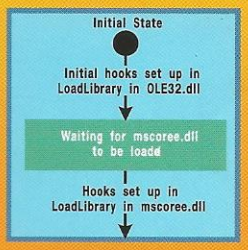
Integrate Far-Flung Data Into Your Spreadsheets With The Help Of Web Services



SharePoint

Maxim V. Karpov and Eric Schoonover page 51

Add A Recycle Bin To Windows SharePoint Services For Easy Document Recovery



Interop

Matt Adamson page 77

Get Seamless .NET Exception Logging From COM Clients Without Modifying Your Code

Plus from closer to home:

Win a year of hosting on a Microsoft Windows Virtual Dedicated Server running Whidbey, Crystal Reports 10 Advanced Developer, Xbox games and much more by entering our writing competition!

The Ultimate Guide to the Microsoft Community page 6

Eyes Wide Open at Microsoft MDC India page 8

Microsoft®