Security Advisory

Microsoft ASP.NET Web Services

Unhandled exception leads to file system disclosure and SQL injection.

Net Square Solutions Pvt. Ltd.
http://www.net-square.com
Shreeraj Shah [shreeraj@net-square.com]
18th May 2005

Advisory ID: NS-052005-ASPNET

Product: IIS running with .Net Framework

Vendor: Microsoft (http://www.microsoft.com)

Platforms:

[Testing has been done on the following:]

1. Windows 2000 + IIS5 + .Net Framework 1.1
2. Windows 2003 + IIS6

Risk: Low (Caution for developers)

Vendor's response: Problem is identified and isolated by Microsoft. An upcoming service pack or next version of this product will address this issue

Guidance URL from Microsoft:

```
http://msdn.microsoft.com/library/default.asp?url=/library/en-
us/vbcon/html/vbtskdisplayingsafeerrormessages.asp
```

Problem domain:

- Web Services running on the ASP.NET framework may disclose an internal system path, if an exception is not handled properly in the source code.

- Web Services running on the ASP.NET framework may disclose possible SQL injection points if an exception is not handled properly in the source code.

Solution:

The solution to this problem can be achieved in following two ways:

- By following secure programming practices and implementing exception handling mechanisms [catching exceptions] at the source code level. This is discussed as part of this document.
- Defending information leakage at the web server level. The Microsoft Security Response Center (MSRC) has taken this issue very seriously and will ensure this issue is addressed in the next servicepack or version of the product.

Problem 1: File system information leakage

Proof of concept:

Here is a simple web services resource on *.Net* which takes a *file name* as input and uses that information to open the file. If the file exists, the code is executed. An exception is raised if the file does not exist.

```
-------------------------- Sample code (sample.asmx) --------------------------
<%@ WebService Language="c#" Class="sample" %>
using System;
using System.Web.Services;
using System.IO;
public class sample
{
[WebMethod]
  public void processFile(string fileinfo)
  {
     String file = "c:\\inetpub\\wwwroot\\"+fileinfo;
     FileStream fs=new FileStream(file,FileMode.Open,FileAccess.Read);
  }
}
-------------------------------------------------------------------------------
```

In the above case, *web services* has one method called *processFile* which accepts a file name from the user and processes it. This is the line where the *FileStream* object gets created with the supplied file name.

FileStream fs=new FileStream(file,FileMode.Open,FileAccess.Read);

Now assume, we are looking for the file ***abc.txt*** which is not in the webroot (c:\inetpub\wwwroot) and the SOAP envelope shown below is sent to the server.

```
------------------------------------ Request -----------------------------------
POST /sample.asmx HTTP/1.0
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
1.0.3705.0)
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://tempuri.org/processFile"
Content-Length: 329
Host: localhost

<?xml version="1.0" encoding="utf-8"?><soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body><processFile
xmlns="http://tempuri.org/"><fileinfo>abc.txt</fileinfo></processFile>
</soap:Body></soap:Envelope>
```

```
------------------------------------ Response ------------------------------------
HTTP/1.1 500 Internal Server Error.
Server: Microsoft-IIS/5.0
Date: Sun, 20 Mar 2005 23:46:00 GMT
X-Powered-By: ASP.NET
X-AspNet-Version: 1.1.4322
Cache-Control: private
Content-Type: text/xml; charset=utf-8
Content-Length: 502

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:Server</faultcode>
      <faultstring>Server was unable to process request. --&gt; Could not find
file &amp;quot;c:\inetpub\wwwroot\abc.txt&amp;quot;.</faultstring>
      <detail />
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

As you can see, we have received the string ***c:\inetpub\wwwroot\abc.txt*** as part of *faultstring*. This discloses the internal path of the system. An attacker can send such malformed requests and obtain this information from the system.

Even though *customError*s is On/RemoteOnly in *web.config*, path information is disclosed in faultstring.(<customErrors mode="On"/>)

Countermeasure:

Since default exceptions sent by IIS disclose an internal path, make sure that exceptions are handled by properly in the source code. Take a look at the code below.

This code will protect an application from path disclosures.

```
Try
{
   String file = "c:\\inetpub\\wwwroot\\"+fileinfo;
   FileStream fs=new FileStream(file,FileMode.Open,FileAccess.Read);
 }
catch
{
   // Error handling routine
}
```

After encompassing *FileStream* with the try/catch block, if you send the same request to server, you get a response such as the one shown below. A path disclosure will not occur.

```
------------------------------------ Request ------------------------------------
POST /sample.asmx HTTP/1.0
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
1.0.3705.0)
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://tempuri.org/processFile"
Content-Length: 329
Host: 192.168.131.3

<?xml version="1.0" encoding="utf-8"?><soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body><processFile
xmlns="http://tempuri.org/"><fileinfo>abc.txt</fileinfo></processFile>
</soap:Body></soap:Envelope>
```

```
------------------------------------ Response ------------------------------------
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Sun, 20 Mar 2005 23:48:37 GMT
X-Powered-By: ASP.NET
X-AspNet-Version: 1.1.4322
Cache-Control: private, max-age=0
Content-Type: text/xml; charset=utf-8
Content-Length: 297

<?xml version="1.0" encoding="utf-8"?><soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body>
<processFileResponse  xmlns="http://tempuri.org/"/>
</soap:Body></soap:Envelope>
```

Guidance URL from Microsoft:

`http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbcon/html/vbtskdisplayingsafeerrormessages.asp`

Problem 2: SQL injection leakage

Proof of concept:

Here is a simple web services resource on .Net which takes a parameter as input and uses that information to access the SQL database. If a SQL query is properly built, then the code is executed. An exception is raised if a character like the double quote (**"**) is injected in the parameter.

---------- **Sample code (db.asmx)** ----------

```
<%@ WebService Language="c#" Class="db" %>
using System;
using System.Web.Services;
using System.Data.SqlClient;
using System.IO;
public class db
{
   [WebMethod]
   public string getProductInfo(string id)
   {
      SqlConnection nwindConn = new SqlConnection("Data
      Source=localhost;Initial Catalog=catalog;User ID=sa;Password=junk");
      SqlCommand catCMD = nwindConn.CreateCommand();
      catCMD.CommandText = "SELECT * FROM items
                                 where   product_id =" + id;
      nwindConn.Open();
      SqlDataReader myReader = catCMD.ExecuteReader();
      while (myReader.Read())
      {
         // Source
      }
       myReader.Close();
       nwindConn.Close();
       return "Success";
    }
}
```

In above case, a SQL connection is created and information is fetched from server. In our example, we have **id=1** which exists and we get the data shown below.

```
---------------------------------- Request ----------------------------------
POST /ws/db.asmx HTTP/1.0
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
1.0.3705.0)
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://tempuri.org/getProductInfo"
Content-Length: 317
Expect: 100-continue
Connection: Keep-Alive
Host: 192.168.131.50

<?xml version="1.0" encoding="utf-8"?><soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body><getProductInfo
xmlns="http://tempuri.org/"><id>1</id></getProductInfo>
</soap:Body></soap:Envelope>
```

```
---------------------------------- Response ----------------------------------
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Fri, 15 Apr 2005 18:29:31 GMT
X-Powered-By: ASP.NET
X-AspNet-Version: 1.1.4322
Cache-Control: private, max-age=0
Content-Type: text/xml; charset=utf-8
Content-Length: 375

<?xml version="1.0" encoding="utf-8"?><soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body>
<getProductInfoResponse xmlns="http://tempuri.org/">
  <getProductInfoResult>Success</getProductInfoResult>
</getProductInfoResponse></soap:Body>
</soap:Envelope>
```

As you can see, we received the string "***success***" as expected.

Now if we send a double quote **(")** instead of **1** we get the following response:

```
---------------------------------- Request ----------------------------------
POST /ws/db.asmx HTTP/1.0
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
1.0.3705.0)
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://tempuri.org/getProductInfo"
Content-Length: 317
Expect: 100-continue
Connection: Keep-Alive
Host: 192.168.131.50

<?xml version="1.0" encoding="utf-8"?><soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body><getProductInfo
xmlns="http://tempuri.org/"><id>"</id></getProductInfo>
</soap:Body></soap:Envelope>
```

```
---------------------------------- Response ----------------------------------
HTTP/1.1 500 Internal Server Error.
Server: Microsoft-IIS/5.0
Date: Fri, 15 Apr 2005 18:32:34 GMT
X-Powered-By: ASP.NET
X-AspNet-Version: 1.1.4322
Cache-Control: private
Content-Type: text/xml; charset=utf-8
Content-Length: 508

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:Server</faultcode>
      <faultstring>Server was unable to process request. --&gt; Cannot use
empty object or column names. Use a single space if necessary.</faultstring>
      <detail />
    </soap:Fault>
  </soap:Body>
```

As you can see we get *faultstring* which is mentions ***column*** as a keyword. This is clearly possible opportunity for SQL injection to a hacker or an attacker. It is another example of information disclosure, and that application may or may not be vulnerable to SQL injection but can be start point for future probes.

Even though *customErrors* is On/RemoteOnly in *web.config*, above information is disclosed in *faultstring*.(<customErrors mode="On"/>)

Countermeasure:

Since default exceptions sent by IIS show word "*column*", make sure that exceptions are handled properly in the source code. Take a look at the code below.

This code will protect an application from *faultstring* leakage.

```
[WebMethod]
public string getProductInfo(string id)
{
   try
   {
        string fulltext = "";
        SqlConnection nwindConn = new SqlConnection("Data
             Source=localhost;Initial Catalog=catalog;User ID=sa;Password=junk");
        SqlCommand catCMD = nwindConn.CreateCommand();
        catCMD.CommandText = "SELECT * FROM items
                                  where product_id ="+id;
        nwindConn.Open();
        SqlDataReader myReader = catCMD.ExecuteReader();
        while (myReader.Read())
        {
            //Source
        }
        myReader.Close();
        nwindConn.Close();
        return "Success";
    }
    catch
    {
        return "Error";
    }
}
```

This time we get the string "***Error***" instead of *faultstring,* since the exception is handled properly.

```
------------------------------------ Request ------------------------------------
POST /ws/db.asmx HTTP/1.0
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
1.0.3705.0)
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://tempuri.org/getProductInfo"
Content-Length: 317
Host: 192.168.131.50

<?xml version="1.0" encoding="utf-8"?><soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body><getProductInfo
xmlns="http://tempuri.org/"><id>"</id></getProductInfo></soap:Body>
</soap:Envelope>
```

```
------------------------------------ Response ------------------------------------
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Fri, 15 Apr 2005 18:38:22 GMT
X-Powered-By: ASP.NET
X-AspNet-Version: 1.1.4322
Cache-Control: private, max-age=0
Content-Type: text/xml; charset=utf-8
Content-Length: 373

<?xml version="1.0" encoding="utf-8"?><soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body>
<getProductInfoResponse xmlns="http://tempuri.org/">
  <getProductInfoResult>Error</getProductInfoResult>
</getProductInfoResponse></soap:Body>
</soap:Envelope>
```

Guidance URL from Microsoft:

`http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbcon/html/vbtskdisplayingsafeerrormessages.asp`

DISCLAIMER