XML injection

Name: XML injection

Description: Building an XML document from user-controlled sources is vulnerable to insertion of malicious code by the user.

ID: cs/xml-injection

Kind: problem

Severity: error

Precision: high

The APIs provided by the .NET libraries for XML manipulation allow the insertion of "raw" text at a specified point in an XML document. If user input is passed to this API, it could allow a malicious user to add extra content that could corrupt or supersede existing content, or enable unintended additional functionality.

Recommendation

Avoid using the `WriteRaw` method on `System.Xml.XmlWriter` with user input. If possible, use the high-level APIs to write new XML elements to a document, as these automatically escape user content. If that is not possible, then user input should be escaped before being included in a string that will be used with the `WriteRaw` API.

Example

In this example, user input is provided describing the name of an employee to add to an XML document representing a set of names. The `WriteRaw` API is used to write the new employee record to the XML file.

```
1  using System;
2  using System.Security;
3  using System.Web;
4  using System.Xml;
5
6  public class XMLInjectionHandler : IHttpHandler {
7    public void ProcessRequest(HttpContext ctx) {
8      string employeeName = ctx.Request.QueryString["employeeName"];
9
10     using (XmlWriter writer = XmlWriter.Create("employees.xml"))
11     {
12         writer.WriteStartDocument();
13
14         // BAD: Insert user input directly into XML
15         writer.WriteRaw("<employee><name>" + employeeName + "</name></employee>");
16
17         writer.WriteEndElement();
18         writer.WriteEndDocument();
19     }
20   }
21 }
```

However, if a malicious user were to provide the content `Bobby Pages</name><
/employee><employee><name>Hacker1`, they would be able to add an extra entry into the XML file.

The corrected version demonstrates two ways to avoid this issue. The first is to escape user input before passing it to the `WriteRaw` API, which prevents a malicious user from closing or opening XML tags. The second approach uses the high level XML API to add XML elements, which ensures the content is appropriately escaped.

```
1  using System;
2  using System.Security;
3  using System.Web;
4  using System.Xml;
5
6  public class XMLInjectionHandler : IHttpHandler {
7    public void ProcessRequest(HttpContext ctx) {
8      string employeeName = ctx.Request.QueryString["employeeName"];
9
10     using (XmlWriter writer = XmlWriter.Create("employees.xml"))
11     {
12         writer.WriteStartDocument();
13
14         // GOOD: Escape user input before inserting into string
15         writer.WriteRaw("<employee><name>" + SecurityElement.Escape(employeeName) + "</name><
/employee>");
16
17         // GOOD: Use standard API, which automatically encodes values
18         writer.WriteStartElement("Employee");
19         writer.WriteElementString("Name", employeeName);
20         writer.WriteEndElement();
21
22         writer.WriteEndElement();
23         writer.WriteEndDocument();
24     }
25   }
```

References

- Web Application Security Consortium: XML Injection.
- Microsoft Docs: WriteRaw.
- Common Weakness Enumeration: CWE-91.