Internet Explorer Vulnerabilities

In this tutorial i'm going to write Internet Explorer Vulnerabilities With Exploits and complete information about them .

Internet Explorer Cross-Site Scripting in Unparsable XML Files

Overview:
Internet Explorer automatically attempts to parse any XML file requested individually by the browser. When the parsing process is successful, a dynamic tree of the various XML elements is presented. However, when a parsing error occurs Internet Explorer displays the parse error along with the URL of the requested XML file.

Description:
We have found that in some cases the displayed URL is not filtered appropriately, and may cause HTML that was passed in the querystring of the URL to be rendered by the browser. This creates a classic cross-site scripting attack in almost any XML file that MSXML fails to read. Practically, this means that leaving XML files on your server that can't be parsed correctly by Internet Explorer and MSXML is exposing the site to a global Cross-Site Scripting attack.

We have been able to reproduce this problem in various setups, but we couldn't pinpoint the vulnerable component reliably enough. It is most likely an MSXML issue, and not a flaw in Internet Explorer itself.
Exploit:
This sample shows the basic URL for injecting content:

```
http://host.with.unparsable.xml.file/flaw.xml?<script>alert(document.cookie)</script>
```
Solution:
Microsoft was notified on 20-Feb-2003. They reported that they were able to reproduce this flaw on IE6 Gold, and no other version. Our research showed different, yet inconsistent results .

Microsoft Internet Explorer Buffer Overflow in Processing '.MHT' Web Archives Lets Remote Users Execute Arbitrary Code

A buffer overflow vulnerability was reported in the Microsoft Internet Explorer browser in the processing of malformed '.mht' web archive pages. A remote user can cause arbitrary code to be executed on the target user's system.

Canon System Solutions reported that a remote user can embed executable content within a web archive and, using a malformed MIME header, cause the content to be executed when the archive is loaded on a target user's browser. The code will run with the privileges of the target user.

If the MIME header structure and the accompanying Base64-encoded text is properly crafted (i.e., properly malformed), 4 bytes of memory can be overwritten, according to the report.

A demonstration exploit is provided:

```
MIME-Version: 1.0
------=_NextPart_000_0000_01C2E1F4.0D559EA0
Content-Location:file:///tomatell.exe
Content-Transfer-Encoding: base64
```

Internet Explorer Script Injection to Custom HTTP Errors in Local Zone

Overview:
Internet Explorer ships with various internal HTML resource files. The majority of these files are meant to handle custom HTTP errors in web sites (also called "Friendly HTTP error messages"). They all use the same basic pieces of code, with minor changes to the actual content of each resource.

One of the main functions included in the resources is a method to extract the real URL from the resource URL hash. For example, if "site.com" generated a 404 HTTP error, the following URL will be internally requested by IE:
res://shdoclc.dll/404_HTTP.htm#http://site.com/file.html.

The function takes the part after the # sign and attempts to extract the domain of the site, in order to embed it in the content of the custom message.

Description:
We found that the above-mentioned parsing procedure has a flaw in it that may cause arbitrary script commands to be executed in the Local Zone. Leading to potential arbitrary commands execution, local file reading and other severe consequences.

However, Exploiting this procedure requires user-interaction. The user must click the URL presented to it by the resource for the malicious code to execute.

Here is the vulnerable function, precisely as it appears in the resources:

```
function Homepage(){
// in real bits, urls get returned to our script like this:
// res://shdocvw.dll/http_404.htm#http://www.DocURL.com/bar.htm


//For testing use DocURL =
"res://shdocvw.dll/http_404.htm#https://www.microsoft.com/bar.htm"
DocURL = document.location.href;


//this is where the http or https will be, as found by searching for :// but skipping the res://
protocolIndex=DocURL.indexOf("://",4);


//this finds the ending slash for the domain server
serverIndex=DocURL.indexOf("/",protocolIndex + 3);


//for the href, we need a valid URL to the domain. We search for the # symbol to find the
begining
//of the true URL, and add 1 to skip it - this is the BeginURL value. We use serverIndex as
the end marker.
//urlresult=DocURL.substring(protocolIndex - 4,serverIndex);
BeginURL=DocURL.indexOf("#",1) + 1;
if (protocolIndex - BeginURL > 7)
urlresult=""

urlresult=DocURL.substring(BeginURL,serverIndex);

//for display, we need to skip after http://, and go to the next slash
displayresult=DocURL.substring(protocolIndex + 3 ,serverIndex);

// Security precaution: must filter out "urlResult" and "displayresult"
forbiddenChars = new RegExp("[<>\'\"]", "g"); // Global search/replace
urlresult = urlresult.replace(forbiddenChars, "");
displayresult = displayresult.replace(forbiddenChars, "");

document.write('<A target=_top HREF="' + urlresult + '">' + displayresult + "</a>");

}
```

The comments in this function teach us that Microsoft had indeed attempted to protect this resource from being exploited in this way, but unfortunately failed to do so. A specially crafted value appended after the # sign can fool this function to write a "javascript:" URL in the displayed link.
Exploit and Demonstration:
This URL will cause the resource to output a "javascript:" link to the document, which will execute when the user clicks on it:

```
res://shdoclc.dll/HTTP_501.htm#javascript:%2f*://*%2falert(location.href)/
```

Copy and paste the above URL in your browser, then click the red link in order to test it.
Solution:
Microsoft was notified on 20-Feb-2003. They were able to reproduce this on IE6 Gold and all versions below it. We managed to reproduce it on all versions, including IE6 SP1, with no exceptions.

They plan to fix this flaw in a future service pack.

IE - Outlook - MS SHLWAPI Render Vulnerabilities

One:
Malicious htm file can freeze IE with 100% CPU usage:
Construct the file freeze.htm:

```
c:\>perl -e "print qq'\xFF\xFE'; print qq'\r\n' x 30000" > freeze.htm
```

After opening freeze.htm IE will hang with 100% CPU usage until IEXPLORE.EXE process is not killed. Two bytes (0xff 0xfe) at the beginning of the file mean that the encoding is unicode. So the internal unicode representation of the CR LF sequence will look like 0D0A0D0A but not 000D000A (if the file was a plain ASCII).

Tested on IE 6.0 with all fixes, other versions expected to be vulnerable aswell.

Two:
A bug was reported on the 23 of this month where a html file could contain code such as:

```
--------------------
<html>
<form>
<input type crash>
</form>
</html>
--------------------
```

that would crash IE, Outlook, Frontpage and most Microsoft programs with the following error:
"Unhandled exception in iexplore.exe (SHLWAPI.DLL): 0xC0000005: Access Violation"
(It's a null pointer overwrite, so it's not easly exploitable...)

In addition to this, it also seems to crash explorer.exe when the .html file containing the code is copied into any folder !! It may work since windows is trying to create a view in Windows explorer. Indeed, it doesn't work when the file is copied in the desktop.

Internet Explorer Object Type Property Overflow

Description:
The "Object" tag is used to insert objects such as ActiveX components into HTML pages. The "Type" property of the "Object" tag is used to set or retrieve the MIME type of the object. Typical valid MIME types include "plain/text" or "application/hta", "audio/x-mpeg", etc. A buffer overflow has been discovered in the "Type" property of the "Object" tag. While there is buffer checking in place, the buffer checking can be overcome by using a special character. From there, the exploitation is a simple, stack-based overflow that allows the remote attacker to run code of his/her choice on the target system.

This attack may be utilized wherever IE parses HTML, so this vulnerability, affects newsgroups, mailing lists, or websites.

Note:
Due to the popularity and prevalence of ActiveX on the Internet, users running Windows 2003 "Enhanced Security Configuration" Mode may have chosen to re-activate the ability to view active content for all websites instead of continually adding websites to the "Internet" or "Trusted" zones on a per-site basis. These users should be aware that they are at risk for this vulnerability and should apply the necessary patch.

Technical Description:
This example was designed for Windows 2000 with .Net Framework and the latest IE.

Cooler Than Centra Spike

Give or take a few '/' characters depending on the system. The issue is relatively simple and interesting: the '/' character is changed into '_/_' (three characters) after the string is checked for proper buffer size. Because of this expansion, we are able to overrun the bounds of the buffer. This allows us to take control of key registers so as to run code that we specify, which will be available at the EDX register. At this point a JMP EDX is called, and from there the payload can be executed.

This issue was discovered by using the same automated testing tool with which we found the Shockwave, MSN Chat, and PNG issues. Additional time was saved through "eVe", a proprietary vulnerability tracing tool which allows for the viewing of checked and unchecked buffers as they are processed in memory.

Patch :
Microsoft was notified and has released a patch for this vulnerability. The patch is available at:
http://www.microsoft.com/technet/security/bulletin/MS03-020.asp

Internet Explorer Plugin.ocx Heap Overflow

There is an exploitable heap overflow vulnerability in Microsoft's ActiveX control, Plugin.ocx. By default, plugin.ocx is marked safe for scripting, and as such, if an IE user were to visit a malicious web page, the overflow could be triggered allowing for a "remote" compromise of the user's machine. Alternatively, an attacker could send their target a specially crafted e-mail, loaded with an exploit to take advantage of this vulnerability. The problem arises by passing an overly long string to the Load method of the control.

By : Ehsan Omidvar

ehsan_omidvar@mail.com