

I've always been intrigued by the way Windows handles URI's (Uniform Resource Identifiers) and it's more well-known offspring, the URL (Uniform resource Locator).

URL's are the standard for locating content on the web. The first field in a properly-coded URL is the protocol, which is set apart by a colon and two forward slashes (://). The most common protocol found in a URL is http. A URL beginning with "http://" indicates that the content about to be loaded is formatted in accordance with the Hypertext Transfer Protocol. Lesser-known protocols are used as well, including FTP (FTP://) MAIL (MAILTO://) and Usenet News (NEWS://).

In an attempt to make Windows more Internet-friendly, URI's can be used in many different places within Windows, and by many different native applications. From a security perspective, this can be a bad thing(TM). For example, if a user clicks on a web page link that references the AOL Instant Messenger protocol (AIM://), modern browsers will open the AIM application and pass the URL's data into AIM. If that data has been maliciously crafted, AIM may break, or worse- your system's security may become compromised by a buffer overflow. It is important to state that this particular example is not necessarily a Windows problem, as it may be argued that the browser caused the problem, or more specifically, the application (AIM) that the browser called. However, URI's are recognized throughout the entire Windows **shell**, not just inside Internet-friendly applications. The Eeye advisory by Marc Maiffret entitled "Windows Shell Overflow" was an excellent example of the bad things that can happen as a result of an OS getting too friendly with URIs. This document will begin to expound a bit more on the disjointed URI research I've been doing lately.

This excellent w3C link provides more information about the URI format. Covered in this document is the correct syntax for URI's including delimiters, escape character handling, query strings, unsafe characters, scheme examples, and much more. Well worth the reading time.

Since I was most interested in learning about URI's in my system, I first started by looking at my Windows registry. In order to find the registered URI's on my system, I used "regfind" from the Windows Resource Kit. My syntax was as follows:

```
regfind "URL:"
```

My results: (My comments are added in parenthesis)

```
\Registry
  Machine
    SOFTWARE
      Classes
        aim           = URL: AOL Instant Messenger Protocol
        callto        = URL: CallTo Protocol (MS Netmeeting)
        file          = URL:File Protocol
        ftp           = URL:File Transfer Protocol
        gopher        = URL:Gopher Protocol
        hotline       = URL:hotline Protocol (another chat proto?)
        http          = URL:HyperText Transfer Protocol
        https         = URL:HyperText Transfer Protocol with Privacy
        irc           = URL:IRC Protocol
        LDAP          = URL:LDAP Protocol
        mailto        = URL:MailTo Protocol
        MMS           = URL:mms Protocol (Microsoft Media Server)
        MMST          = URL:mmst Protocol (Microsoft Media Server - TCP)
        MMSU          = URL:mmsu Protocol (Microsoft Media Server - UDP)
        MSBD          = URL:msbd Protocol (Microsoft Media Stream Broadcast Distr.
        news          = URL:News Protocol (news ala usenet)
        nntp          = URL:NNTP Protocol (news ala usenet)
        quicktime     = URL:QuickTimePlayer Protocol
        rlogin        = URL:RLogin Protocol
        snews         = URL:Snews Protocol (nntp over ssl?)
```

```
telnet          = URL:Telnet Protocol
tn3270          = URL:TN3270 Protocol (3270 telnet)
vnd.ms.radio    = URL: Radio Protocol (oops! MS radio Bugtraq ID #861!)
```

These specific URI's had been purposely placed into my registry by the operating system or third-party applications. By and large, these URI's will be honored system-wide. For example, try running

```
mms://localhost
```

at the Start|Run prompt in Windows. If Microsoft Media Player is installed, it will be run.

In addition to the URIs that were loaded up in my registry, there are other "standards" as proposed by the W3C which may apply to your browser. A very thorough document expounds on these URIs: "Addressing Schemes" by the W3c

Now, referring back to the URI's found in my registry, eeye provides a great roadmap for discovering some problematic URI's that are registered on your system. For example, I discovered that the the "hotline" URI pointed to an application that no longer existed on my system. I discovered this by entering

```
hotline://localhost
```

into my IE v5.50.4522 browser. The resulting message box stated "Windows cannot find HOTLIN~1.EXE". Windows had indeed lost (or uninstalled) this file.

Next, I tested eeye's overflow by entering

```
hotline://
```

followed by 330 'a' characters as the URL, again in my IE browser. Just as the eeye advisory predicted, iexplore.exe exploded with

```
"The instruction at 0x70bd3ad8 referenced memory at 0x00610061. The memory could no
```

Most of these other URI's that I found are interesting in and of themselves, perhaps providing other avenues of "exploration." The potential is certainly there to perform other types of nastyness with malicious web links or perhaps the tried and true email-embedded URL...

I'll post more as I continue to play.

johnny