Next generation malware: Windows Vista's gadget API

Tim Brown
http://www.portcullis-security.com/

September 11, 2007

Abstract

Windows has had the ability to embed HTML into it's user interface for many years. Right back to and including Windows NT 4.0, it has been possible to embed HTML into the task bar, but the OS has always maintained a sandbox, from which the HTML has been unable to escape. All this changes with Windows Vista. This paper seeks to inform system administrators, users and the wider community on both potential attack vectors using gadgets and the mitigations provided by Windows Vista.

Contents

1 So what are gadgets **2**

2 Gadget development **3**
2.1 Building blocks 3
2.2 Interesting API objects and methods 4
2.3 Distribution 4
2.3.1 CAB archives 4
2.3.2 ZIP archives 4
2.3.3 Directories 4

3 Threat analysis **5**
3.1 Attack vectors 5
3.1.1 Javascript injection 5
3.1.2 Malicious feeds 5
3.1.3 JSON style parser bugs 5
3.1.4 HTML parser bugs 6
3.1.5 Traditional man in the middle attacks 6
3.2 Payloads 6
3.2.1 Network attacks 6
3.2.2 Identify theft 7
3.2.3 Ad fraud 7
3.2.4 Network egress filter bypass 7

4 Mitigation and defence **8**

5 Future research **9**
5.1 Better understanding the gadget API 9
5.1.1 gadget.xml tags 9
5.1.2 The gadget API 9
5.2 Short term targets 9
5.2.1 Silently deployment 9
5.2.2 Identification of vulnerable gadgets 9
5.3 The end game 9
5.3.1 Developing malware frameworks 9

6 Changes **10**

Chapter 1

So what are gadgets

Microsoft describe[1] gadgets as:

> Gadgets are easy-to-use mini programs that give you information at a glance and provide easy access to frequently used tools. Windows Sidebar helps you to organize your gadgets.
>
> As you use your computer to access information, perform tasks, and interact with software, you may at times feel like you're facing information overload. You have to open a web browser just to check the weather, open an application to view your calendar, and open a calculator program just to add up a few numbers. Now, with Windows Sidebar and its associated mini-applications called gadgets, the specific information you need is at your fingertips.

Sound rather harmless don't they? Well they certainly sound interesting[2].

Chapter 2

Gadget development

2.1 Building blocks

Below is an example of the process required to build a typical gadget:

1. Create a HTML text file called `evilgadget.html`:

```
<html>
<body>
<script>
System.Shell.execute("calc.exe");
</script>
</body>
</html>
```

2. Create the gadget manifest, called `gadget.xml`:

```
<?xml version="1.0" encoding="utf-8"?>
<gadget>
<name>EvilGadget</name>
<namespace>Org.Example.EvilGadget</namespace>
<version>1.0</version>
<author name="A.N. Author">
<info url="www.example.org"/>
</author>
<copyright>2007</copyright>
<description>My first evil gadget</description>
<hosts>
<host name="sidebar">
<base type="HTML" apiVersion="1.0.0" src="evilgadget.html"/>
<permissions>full</permissions>
<platform minPlatformVersion="0.3"/>
</host>
</hosts>
</gadget>
```

This is the basic structure of a gadget, that when packaged up and installed will execute `calc.exe`. Note the `permissions` required for this application are set to `full`. This appears to be the only value currently accepted by Windows Vista. We'd prefer to have seen a range of permissions, with prompts to the user before gadgets that require more permissive access to the system are executed, but perhaps this is planned in the future.

2.2 Interesting API objects and methods

The Windows Vista gadget API contains a number of objects and methods which which are interesting from a malware perspective. These include:

- `ActiveXObject`
- `System.Environment`
- `System.Shell`
- `System.Machine`
- `XMLHttpRequest`

2.3 Distribution

2.3.1 CAB archives

CAB archives are the recommended way to distribute gadgets. In this format, it's possible to digitally sign the resulting `.gadget` file, allowing the end user to verify the original distributor.

2.3.2 ZIP archives

Gadgets can also be distributed as ZIP archives again in files ending with the `.gadget` extension. ZIP files can't be cryptographically signed.
By default, attempting to serve one through Apache web server results in IE incorrectly identifying the file as being a standard ZIP archive. However, this rectified using the configuration directive `AddType Windows.gadget .gadget`.
Installing a gadget distributed as a ZIP archive results in a two confirmation requests. UAC confirms that the user wishes to install the downloaded before detailing the untrusted nature of the file. It's worth noting that the user gets one confirmation request if they save the file to disk first (which can happen if the gadget type hasn't been set up), simply detailing the untrusted nature of the file.

2.3.3 Directories

Finally, it's also possible to drop directories ending with the `.gadget` extension directly in to relevant directory, where upon they will be available to be added to a users sidebar. Such locations include:

- `\Program Files\Windows Sidebar\Gadgets`
- `\Program Files\Windows Sidebar\Shared Gadgets`
- `\Users\UserName\AppData\Local\Microsoft\Windows Sidebar\Gadgets`

One good thing to note is that the example gadgets supplied by Microsoft have ACLs applied such that it is not trivially possible to modify them. Only the `TrustedInstaller` group has write access to the relevant files. However, this isn't the case for gadgets added by users.

Chapter 3

Threat analysis

3.1 Attack vectors

3.1.1 Javascript injection

Imagine what happens if an attacker injects malicious Javascript into a gadgets remote dependency. Where might we find such vulnerable code? Having had a look at the Microsoft provided examples, it is clear that some santisation of input occurs, particularly when gadgets make use of the inbuilt feed manager that comes as part of Windows Vista. For example, passing in Javascript embedded in an RSS feed using is possible[3] - kudos to IDefense for spotting this.

3.1.2 Malicious feeds

The feed manager isn't perfect, the sample gadgets themselves mention and mitigate at least one bug that can allegedly cause the feed manager to crash:

```
//////////////////////////////////////////////////////////////////////////////
//
//  Bug #1779555: NULL Bstr can cause MS Feed manager crash because it doesn't
//                accept NULL BSTR. Feed team won't change in their code. So we
//                have to make sure no NULL BSTR passed into GetFeed()
//
//  !!! Always call safeGetFeed, NOT   g_msFeedManager.GetFeed() !!!
//
//////////////////////////////////////////////////////////////////////////////
```

3.1.3 JSON style parser bugs

Another way to attack the gadget API is to look for cases where developers are rolling their own parsers for external data. An example would be where gadgets operate on sites that are known to use JSON as gadgets will need to eval the web sites response in order to access it from within their own Javascript. We have developed proof of concept code which shows how this might work.
In fairness to Microsoft they have already acknowledged that this might be possible. Dave Ross and Michael Howard wrote a nice document[4] on how such an attack might be mitigated. However it isn't widely documented and it relies on developers to do the right thing.

3.1.4 HTML parser bugs

The sidebar is simply an application of Microsoft's own HTML/XML rendering technologies. We've all seen the bypasses documented in the XSS cheat sheet[5]. Go looking for IE 7 specific bypasses and then examine how the gadget rendering engine deals with them?

3.1.5 Traditional man in the middle attacks

Even gadgets that have been written safely aren't immune to attack. Consider attacking existing gadgets using a traditional man in the middle attack. If you can control a network segment between the gadget and the source web site, there's nothing to stop you injecting malicious code of your choosing.

3.2 Payloads

3.2.1 Network attacks

Once a system has been compromised by a malicious gadget, there are numerous ways that malware can take advantage of the compromise. Since gadgets can make cross domain requests, consider the following HTML:

```
<html>
<body>
<script>
HTTPSession = new XMLHttpRequest();
HTTPSession.open("GET", "http://www.example.org/test", false);
HTTPSession.send(null);
</script>
</body>
</html>
```

Below is an example of the log entry generated in Apache's logs for the above cross domain request. Note the the use of `x-gadget` in the referrer field. Gadgets use this special protocol handler to reference other files included within the gadget archive. We can also use this in server side code to specifically target Windows Vista's gadgets.

```
www.example.org - - [05/Apr/2007:01:04:38 +0100] "GET /test HTTP/1.1" 404 293
"x-gadget:///test.html" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0;
SLCC1; .NET CLR 2.0.50727; .NET CLR 3.0.04506)"
```

Using cross domain requests as described above, we were able to develop a proof of concept port scanner. This port scanner draws on techniques first documented by SPI Dynamics[6], but attempts full blown banner grabbing where possible to aid in accurately determining the protocols supported by each open port. It is left to the reader to consider further enhancements which may aid in attacking other systems. Hint, the paper Inter-Protocol Communication by Wade Alcorn[7] might be a good place to start.
Microsoft have attempted to secure the cross domain request model as cookies set by the web server in response to browser requests are not sent by gadgets. That is to say that gadgets treat web browsers cookies much as if they were marked HTTP only.

3.2.2 Identify theft

Whilst gadgets can't directly access cookies, they can access the filing system[8]. It should in theory be possible for a gadget to load a raw cookie file from the disk and use this to access previously accessed web sites as the authenticated user. We haven't developed a proof of concept for this attack but we've seen nothing yet that suggests this wouldn't be possible.

3.2.3 Ad fraud

Using the gadget APIs for fetching and parsing RSS feeds, it is also possible to develops botnets which abuse advertising networks such as Google to fraudulently manipulate click through statistics for ads providing a steady stream of income to those who develop such attacks.

3.2.4 Network egress filter bypass

Consider a locked down corporate desktop running Windows Vista which has been configured to prevent the execution of arbitrary executables. As noted above, the gadget API includes the method `System.Shell` which includes an execute method, intended to allow local files to be executed. By making use of `XMLHttpRequest` to bypass traditional egress firewalls which force external access to made via a proxy, in combination with the ActiveX objects responsible for file access, gadgets can download and write files. It is then a simple matter of calling `System.Shell.execute()` in order to have arbitrary executable running under the users current privileges.

Chapter 4

Mitigation and defence

Whilst it is clear that Microsoft have accounted for the corporate market and enhanced the security policy[9] of Windows Vista to include new settings to control the use of gadgets, my question is whether this is enough? It is clear from previous attempts by Microsoft to introduce new methods of enhancing their products such as ActiveX and macros that users simply do not understand the connotations of the decisions they are being asked to make, particularly where the files concerned sound so benign. It is our judgement, that virus scanner and anti malware vendors should start looking at gadgets now.

Chapter 5

Future research

5.1 Better understanding the gadget API

5.1.1 gadget.xml tags

The namespace, host and permissions tags within gadget.xml look particularly interesting candidates for further research. We're intrigued as to whether it is possible to install new gadgets which will then participate in an existing gadget's namespace, what other hosts apart from sidebar may exist and whether it is possible to decrease the rights of gadgets by modifying their permissions tag value.

5.1.2 The gadget API

The gadget API is new and untested and as noted above has at least 1 Microsoft acknowledged bug which can cause it to crash. How many more such bugs exist and are any of them exploitable? We're also intending to take a further look at just what exactly the full API allows us to do?

5.2 Short term targets

5.2.1 Silently deployment

We've looked at how gadgets are exposed to the sidebar, but what files control which gadgets are loaded when the sidebar is started? Gadgets will be installed (after prompting) if you leave them in the usual startup locations.

5.2.2 Identification of vulnerable gadgets

Grep, grep and then grep again, but how exactly can you find exploitable gadgets? We've proved the vulnerability exists from a conceptual perspective, but maybe no such gadgets exist in the wild?

5.3 The end game

5.3.1 Developing malware frameworks

Based on our and other organisations research, we're theorising it should be possible to develop a complete framework allowing compromise, control and update of compromised systems, along with further exploitation of other systems on the compromised network. We could even proxy our own web browsing through a malicious gadget. Something like Metasploit meets XSSShell[10]?

Chapter 6

Changes

0.9.9-1.0.0 First published
0.9.0-0.9.9 External peer review
0.0.0-0.9.0 Internal peer review

Bibliography

[1] *http://www.microsoft.com/windows/products/windowsvista/features/details/sidebargadgets.mspx*

[2] *http://www.aavar.org/avar2006/Program/ericchien.html*

[3] *http://www.microsoft.com/technet/security/bulletin/MS07-048.mspx*

[4] *http://msdn2.microsoft.com/en-us/library/bb498012.aspx*

[5] *http://ha.ckers.org/xss.html*

[6] *http://www.spidynamics.com/assets/documents/JSportscan.pdf*

[7] *http://bindshell.net/papers/ipc*

[8] *http://channel9.msdn.com/ShowPost.aspx?PostID=236914*

[9] *http://msdn2.microsoft.com/en-us/library/aa965881.aspx*

[10] *http://www.portcullis-security.com/16.php*