

# IObjectSafety and Internet Explorer

**Source:** <http://www.derkeiler.com/Mailing-Lists/securityfocus/bugtraq/2005-03/0010.html>

---

**From:** Shane Hird ([shanehird\\_at\\_yahoo.com](mailto:shanehird_at_yahoo.com))

**Date:** 03/01/05

Date: Tue, 1 Mar 2005 06:59:35 -0800 (PST)

To: [bugtraq@securityfocus.com](mailto:bugtraq@securityfocus.com)

-----Summary

Problems with ActiveX in Internet Explorer are nothing new. However, I believe there is a design flaw in the way they are implemented in IE which could be easily corrected, but has never been addressed.

The following issues with the use of IObjectSafety in Internet Explorer can be summed up with this excerpt from a Microsoft knowledge base article (PSS ID Number: 216434)

INFO: How Internet Explorer Determines If ActiveX Controls Are Safe  
<http://support.microsoft.com/kb/q216434/> :

"There are two ways to mark a control as safe for scripting and initialization:

Implement the IObjectSafety interface.

Provide the following registry keys for the control's CLSID under the Implemented Categories section:

The following key marks the control safe for scripting:

{7DD95801-9882-11CF-9FA9-00AA006C42C4}

The following key marks the control safe for initialization from persistent data:

{7DD95802-9882-11CF-9FA9-00AA006C42C4}

Microsoft recommends that you implement IObjectSafety to mark a control as safe or unsafe. This prevents other users from repackaging your control and marking it as safe when it is not.

1] The IObjectSafety interface allows a container to retrieve the control's initialization and scripting capabilities through its SetInterfaceSafetyOptions method. First, Internet Explorer checks to see if a control implements the IObjectSafety interface. If it does, Internet Explorer calls SetInterfaceSafetyOptions for the IPersist interfaces to check if the object is safe for initialization. When a control is first scripted, Internet Explorer first calls SetInterfaceSafetyOptions on the IDispatchEx interface of the control. If that fails, it calls

SetInterfaceSafetyOptions on the IDispatch interface.

<snip>

2] If the control does not implement the IObjectSafety interface, Internet Explorer looks under the Implemented Categories section of the control for the keys mentioned above. If these keys are not present, Internet Explorer warns the user according to the security settings."

-----Design flaw

What this article fails to mention is that "checks to see if a control implements the IObjectSafety interface" requires and results in the starting of the COM server process. This is due to the requirement of COM that querying for an interface is done thorough the servers running code, rather than a static lookup for the interface.

This means that, even if the COM server has not been marked as safe, or was even built before the existence of Internet Explorer, it can still be started (at least to the point where IObjectSafety can be queried) by arbitrary web pages on the Internet. (with the default IE "Medium" security settings).

AFAIK, this is also relates to why there was the spate of {1111-1111-11..} codebase=calc.exe type exploits possible in IE.

This poses two problems:

---<1) We have no easy way of determining what COM servers on a given machine can be started and scripted by IE.

Enumerating safe objects using the registry keys is both fast and stable. But with the addition of objects which can only be determined if they are safe by starting the (potentially heavyweight) COM server and querying them, this becomes impractical to do.

---<2) Any COM server can be started, including potentially corrupt or dangerous servers, that were never marked as safe.

Just starting the server and querying for IObjectSafety in 99% of cases isn't going to cause any significant security violation. However this is dependent on the particular components installed on the machine and how they initialise. Components that may never have been intended to be started from remote web pages. It also poses a stability issue for IE.

-----Exploitable "safe" objects

To give an example of a control which has "IObjectSafety" but not marked as safe by keys in the registry, we have the Log Sink class provided by pkmcore.dll (Common Files/Microsoft Shared/Web Folders/).

## SecurityFocus Bugtraq: IObjectSafety and Internet Explorer

This object would allow a remote attacker to write data to any file. I.e..

```
<object id=ctl
classid="clsid:{DE4735F3-7532-4895-93DC-9A10C4257173}"></object>
<script language="vbscript">
ctl.initsink "C:\autoexec.bat"
ctl.addstring "echo Drive formatted? ", ""
ctl.deinitsink
</script>
```

After discovering this object, a search through Microsoft's security bulletins revealed nothing, however the support database revealed this article:

<http://support.microsoft.com/default.aspx?scid=kb;en-us;321780>

One 128mb patch later for a product I never installed (I believe it is installed as a part of Office 2003), and the file is updated to no longer allow log file creation by remote sites. (Although a recent service pack for Office 2003 also claims to update the file to the same corrected version).

Regardless of the current exploitability of this object, it illustrates the point that potentially insecure components can lay dormant on a machine, with no easy way of determining whether IE will make use of them or not.

-----IObjectSafety Enumeration

I have written some very rough code to check for COM servers which implement the IObjectSafety interface. There is also code for "fuzzing" the IDispatch interface of the components, as well as any IDispatch interfaces returned from the methods, by calling every method with garbage values, or overly long BSTRs.

This is by no means production code, but it serves its purpose:

<http://sourceforge.net/projects/axfuzz/>

There is also a modified tscon32 sample which helps to fuzz objects which expect an OLE host.

These programs try to "emulate" Internet Explorer with its ActiveX hosting, by following the steps outlined in the knowledge base article above. If a COM server crashes or formats your drive while using these programs, theoretically it would have also been possible from a remote web page. The exceptions are: it ignores 'Kill bits', doesn't correctly do the OLE 'stuff', and still treats a control as safe if IObjectSafety returns "unsafe" but the registry marks otherwise (uncommon).

Its interesting to note that after running these tools, there are often many processes left hanging and general misbehaviour. It is generally best to reboot after running it. Having said that, there is nothing stopping a web page from doing the exact same thing as it works in the same way that IE does.

## SecurityFocus Bugtraq: IObjectSafety and Internet Explorer

Using this code, I have uncovered the following objects that will crash or otherwise render unusable the Internet Explorer process when hosted on a web page. They are all possible without any user interaction or lowering of security settings.

To generate that list, I simply ran axenum which ran through and tried to instantiate each component on my machine. When an exception occurred, it outputted the CLSID of the component. I then took each CLSID and created a web page and visited it remotely using IE, and logged which ones crashed IE.

There were a lot more components that generated exceptions, however didn't when hosted in IE. I assume this is because the initialisation code makes certain assumptions about its host, which can cause a crash when hosted under something that isn't OLE aware for example, but work fine when they are.

```
**** {0006F071-0000-0000-C000-000000000046}
```

```
**** Outlook Progress Ctl
```

```
**** {0CF32AA1-7571-11D0-93C4-00AA00A3DDEA}
```

```
**** System Monitor Source Properties
```

```
**** {1AA06BA1-0E88-11d1-8391-00C04FBD7C09}
```

```
**** CLSID_CCommAcctImport
```

```
**** {233A9694-667E-11d1-9DFB-006097D50408}
```

```
**** Outlook Express Address Book
```

```
**** {283807B8-2C60-11D0-A31D-00AA00B92C03}
```

```
**** Danim
```

```
**** {2c10a98f-d64f-43b4-bed6-dd0e1bf2074c}
```

```
**** Microsoft Visual Database Tools Query Designer V7.0
```

```
**** {3050f667-98b5-11cf-bb82-00aa00bdce0b}
```

```
**** Microsoft Html Popup Window
```

```
**** {549B5CC4-4A86-11D7-A4DF-000874180BB3}
```

```
**** SmartConnect Class
```

```
**** {8E26BFC1-AFD6-11CF-BFFC-00AA003CFDFC}
```

```
**** Helper Object for Java
```

```
**** {8EE42293-C315-11D0-8D6F-00A0C9A06E1F}
```

```
**** CLSID_ApprenticeICW
```

```
**** {8FE7E181-BB96-11D2-A1CB-00609778EA66}
```

```
**** Microsoft MS Audio Decompressor Control Property page
```

```
**** {ABBA001B-3075-11D6-88A4-00B0D0200F88}
```

```
**** OpenCable Class
```

## SecurityFocus Bugtraq: IObjectSafety and Internet Explorer

\*\*\*\* {CE292861-FC88-11D0-9E69-00C04FD7C15B}  
\*\*\*\* VideoPort Object

\*\*\*\* {F0975AFE-5C7F-11D2-8B74-00104B2AFB41}  
\*\*\*\* WMI ADSI Extension

//Im not too sure uuidgen was used here...

\*\*\*\* {CAFEEFAC-0014-0002-0003-ABCDEFEDCBA}  
\*\*\*\* Java Plug-in 1.4.2\_03

\*\*\*\* {CAFEEFAC-0014-0002-0003-ABCDEFEDCBB}  
\*\*\*\* Java Plug-in 1.4.2\_03 <applet> redirector

\*\*\*\* {CAFEEFAC-0014-0002-0004-ABCDEFEDCBA}  
\*\*\*\* Java Plug-in 1.4.2\_04

\*\*\*\* {CAFEEFAC-0014-0002-0004-ABCDEFEDCBB}  
\*\*\*\* Java Plug-in 1.4.2\_04 <applet> redirector

Interestingly, most of these problems cannot be reproduced by opening the page in the 'local zone', as Internet Explorer blocks the object with the "yellow information bar". Yet, IE does not do this when viewing the page in the Internet Zone. Also, "Maxthon", an MSHTML host which effectively acts as a different chrome to Internet Explorer, did not appear to be vulnerable to many of the above list either.

-----IMHO

With IE7 in the works, perhaps Microsoft could take the following suggestion into consideration:

To determine if an ActiveX control is safe:

- 1) First check the registry for the "safe keys".
  - 1.1) If the key exists, query IObjectSafety and proceed as usual.
  - 1.2) If the key doesn't exist, ideally IE should proceed no further.

But some objects may have been installed which don't set these keys and assume the IObjectSafety call. For compatibility, if the key doesn't exist, ask the user if they want to try add it as a usable add on (yellow information bar can be used).

- 2) If the user agrees, and it passes the IObjectSafety check, add the "safe" category keys to the registry for that component. Otherwise fail with no option to override – depending on security settings.

---

With this system in place, it becomes a lot easier to enumerate all COM objects on the system with this key set and therefore 'startable' by IE. IE should show all components marked as safe in the manage add ons screen ("safe" objects on a system, effectively are IE add-ons, I believe the user

## SecurityFocus Bugtraq: IObjectSafety and Internet Explorer

should be made aware of them). This screen should have the ability to remove the 'safe' keys, and set kill bits for components that automatically re-add the keys or you don't want to be prompted about.

This can all be done via the 'information bar' with no annoying prompts to the user. There would effectively be no perceivable difference to the end user other than a safer browsing experience.

I can think of few reasons why MS chose to query IObjectSafety first. The only reason I can think of is querying objects with no local registry information, perhaps remote or recently installed? Regardless, the proposed system would still work with such objects, you would simply be prompted about the objects first use, and have the ability to revoke the permission at a later date.

-----"Work around"

You can use the "Administrator approved" option for ActiveX controls and the policy editor to achieve a similar effect. However, it requires manually inputting safe controls rather than making use of the already "safe" category keys available from ActiveX controls.

-----Vendor Response

That's a feature not a bug.

Or to quote:

"COM objects can be called from an object tag and if they are not actually ActiveX controls they can cause IE to terminate."

-----

Shane Hird.

---

Do you Yahoo!?

Yahoo! Mail - now with 250MB free storage. Learn more.

[http://info.mail.yahoo.com/mail\\_250](http://info.mail.yahoo.com/mail_250)