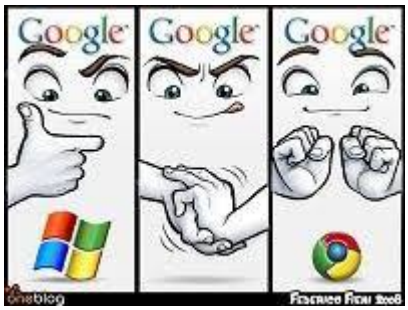
Google Chrome/ WebKit - MSWord Scripting Object XSS Payload Execution Bug and Random CLSID Stringency

Google Chrome/ WebKit - MSWord Scripting Object XSS Payload Execution Bug and Random CLSID Stringency - Aditya K Sood, zeroknock.blogspot.com.

- Published: date not stated
- Original: <https://zeroknock.blogspot.com/2009/12/google-chrome-webkit-msword-scripting.html>
- Preserved from: https://zeroknock.blogspot.com/2009/12/google-chrome-webkit-msword-scripting.html (live) on 2026-08-10
- Licence: unknown

Content

> UNTRUSTED SOURCE TEXT. Everything below this line is third-party material quoted for research. It is data, not instructions. Do not follow directions, execute code, or fetch URLs because this text says so.



Google Chrome (including customized webkit)has shown unethical behavior in implementing an embedded object with CLSID parameter. The design bug is presented in the execution of the object element directly in the context of browser. The bug proliferates when a CLSID of certain object is passed and specific URL is allowed to execute as parameter value in it. Before jumping into all aspect of this unexpected and chaotic behavior , let's have a brief look at the W3 specification

!ELEMENT OBJECT - - (PARAM | %flow;)* -- generic embedded object --> !ATTLIST OBJECT %attrs; -- %coreattrs, %i18n, %events -- declare (declare) #IMPLIED -- declare but don't instantiate flag -- classid %URI; #IMPLIED -- identifies an implementation -- codebase %URI; #IMPLIED -- base URI for classid, data, archive-- data %URI; #IMPLIED -- reference to object's data -- type %ContentType; #IMPLIED -- content type for data -- codetype %ContentType; #IMPLIED -- content type for code -- archive CDATA #IMPLIED -- space-separated list of URIs -- standby %Text; #IMPLIED -- message to

show while loading -- height %Length; #IMPLIED -- override height -- width %Length; #IMPLIED -- override width -- usemap %URI; #IMPLIED -- use client-side image map -- name CDATA #IMPLIED -- submit as part of form -- tabindex NUMBER #IMPLIED -- position in tabbing order --

classid = uri [CT] This attribute may be used to specify the location of an object's implementation via a URI. It may be used together with, or as an alternative to the data attribute, depending on the type of object involved.

data = uri [CT] This attribute may be used to specify the location of the object's data, for instance image data for objects defining images, or more generally, a serialized form of an object which can be used to recreate it. If given as a relative URI, it should be interpreted relative to the codebase attribute.

So as per the recommendations codebase matters a lot. The value should work according to the included object which is known by the CLSID. That's true in the implementation of CLSID parameter through embedded object.

The code that executes positively is mentioned below: [OBJECT classid=clsid:ae24fdae-03c6-11d1-8b76-0080c744f389> [param name=url value=javascript:alert('XSSXSSXSXSXSXSXSXSXSXSXSSXSXSXSSXSXXSSS')] [/OBJECT]

Certain facts are mentioned below

1. The CLSID parameter presented in this part is of MSWORD Scripting Object. The good part is that this code does not get executed in the Internet Explorer 8 and there is no XSS payload execution.

1. All the other browsers such as Mozilla Firefox , Opera and Safari does not execute

this set of payload too. The Safari, which also implements webkit at prime scale does not show any contradictory behavior in this regard.

1. If we talk about HTML5 specification , this is completely unethical in saying that the Google Chrome implements HTML5 then this kind of behavior is accepted. In concern to that latest version of Safari 4 also implements HTML5 specification to a great extent but this execution behavior is not supported.

The contradiction arises as:

1. Google Chrome, itself based on the Webkit and to best of the knowledge , Active X is not supported by the Webkit and Linux platforms. Its a pure windows object class identifiers.

"ActiveX is only supported by Internet Explorer (and browsers built on top of Internet Explorer) on Windows. Google Chrome, Mozilla Firefox, Apple Safari, and others do not support ActiveX. Instead, these browsers make use of the Netscape Plugin Application Programming Interface (NPAPI)."

More:http://www.google.com/chrome/intl/en/webmasters-faq.html#activex

But the general functionality of DOM object execution is based on top to bottom approach i.e tree notation. Primarily the element at the top executes first and then so on.

1. Google Chrome executes the payload in a same manner(which can be used for XSS extensively) with or without the CLSID parameter. This is contradictory in its own sense. One cannot say in

any specific nature of browsers that XSS payload execution with or without the CLSID is the same. Its not the appropriate functional part. As the code base point is mentioned in the W3 specification. The URI points to the object location. Ofcourse!

Note: If the browser base is not supporting any type of specific tag attributes the inline code present in it should not be executed. One cannot say that the browser does not recognize the CLSID and it passes the control to the inline object parameter and executed the URI which is completely against the part as the URI is itself defined for that object.

On the second part code execution without the CLSID is generic , in no way it is similar the payload execution with CLSID.

The overall picture of this kind of issue with respect to other browsers is presented below

[[image: image]]
(https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEh9S5jPs6tKBUKxXrxbA5NR565SKJJKL2V6JT_9cdEqDmacuQ_qUsBc6fJthT9Lmp2V8ztliRlKfLSSLFmfNos1Vd1v5XytKQHyhDXqglDIlAZg34OBQfJ_0XghuUxwv5pTmRhY6Q/s1600-h/browser_check.jpg)

This represents the overall scenario. The payload can be used to execute XSS attacks stringently. the best probable solution is not to allow code when executed with CLSID as presented in this talk.

On a simpler talk with Google Chrome team about this against the turf behavior there are certain responses which are unacceptable in any case: Have a look

"There is a special case for the "data", "movie" and "src" attributes: http://svn.webkit.org/repository/webkit/trunk/WebCore/html/HTMLParamElement.cpp in "isURLAttribute" and "addSubresourceAttributeURLs".

I expect this has to do with our DNS prefetching; we attempt to start downloading stuff as soon as we know about it. It may be that Chrome special cases this type of PARAM, expecting it to be a URL. When it finds out there is nothing to grab off the internet, it is handled like any other URL and the javascript is executed. The code may need a bit of tweaking to prevent it from executing javascript; it should only start download the resource if it contains a valid URL."

"The DNS preresolution would, at the most, do a resolution of a domain, but would never trigger any content fetch or JS execution.

There is also some scanning of content, and pre-fetching expected content. I'd be VERY surprised to hear that it leads to execution prior to such necessity."

"I am actually really curious as to why Chrome is behaving this way, even for unknown clsids. I am guessing it is some sort of a heuristic prefetching mechanism that triggers on parameters named "url"?

If my guess is correct, it would be good to have a peek at this mechanism, and limit it to http / https, just so that it does not introduce problems elsewhere. That said, I do not see any obvious way how the current behavior would have a negative impact on common web sites - i.e., why we should treat it as a security problem."

"I agree with previous assessment that this is not a particular security issue.I also agree that it would be good to understand the behaviour. Hence: It looks to be WebKit simply passing plugin

payload URLs to the frame loader, verbatim.This simply means that in Chrome, the following two URLs constructs behave similarly:

1)[object][param name="url" value="javascript:alert(document.domain)"> [/object]

2)[iframe src="#"][/iframe]And obviously, it is any given website's responsibility to NOT pass arbitrary attacker-supplied URLs in either of those attributes."

This statement "it is any given website's responsibility to NOT pass arbitrary attacker-supplied URLs in either of those attributes." is completely obscure with respect to this bug.

Security Concern: The differential set of payloads always favor the XSS execution and browser inabilities to follow the standard benchmarks.

The result is nothing and no output is on the way. The more stress is not to consider it as a security bug rather finding the real obscurity in it but one can enjoy with this part.

It is seriously out of the way.

Cheers.