Haketilo - Feature #92

Replace cookie smuggling with some safer approach

09/11/2021 09:55 PM - koszko

Status:	Closed	**Start date:**	09/11/2021
Priority:	High	**Due date:**	
Assignee:	koszko	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

Yep, we need to find something that works. registerContentScript() might do the job on newer browsers (and under Manifest v3 for Chrome). What about the rest?

History

#1 - 09/11/2021 09:58 PM - koszko

Jahoti, please, remind me. Why aren't we just making a synchronous AJAX call in the content script and redirecting it appropriately using webRequest?

Btw, I tested that redirecting to data: works under Ungoogled Chromium. I seem to be having problems getting anything to work under Mozilla but honestly, I experimented little

#2 - 09/11/2021 09:58 PM - koszko

- Priority changed from Normal to High

#3 - 09/12/2021 12:53 AM - jahoti

> Jahoti, please, remind me. Why aren't we just making a synchronous AJAX call in the content script and redirecting it appropriately using webRequest?

Primarily because you never mentioned it and I never thought of it :). If this works, it will make life much easier!

> Btw, I tested that redirecting to data: works under Ungoogled Chromium. I seem to be having problems getting anything to work under Mozilla but honestly, I experimented little

I will continue the experimenting right now.

#4 - 09/12/2021 03:00 AM - jahoti

It turns out Firefox did once support redirection to data: URLs (prior to v60, it seems), before it was accidentally broken and forgotten about for long enough the devs decided it wasn't worth fixing: https://bugzilla.mozilla.org/show_bug.cgi?id=1475832. It's quite shocking how many useful old features have been left to rot by Mozilla.

That said, there are several options. Apart from the obvious approach of data: URLs for Chromium and contentScripts.register for Firefox, both browsers support tabs.executeScript (and Manifest V3 has the somewhat similar scripting.executeScript), and somebody has somehow written a contentScripts.register polyfill for Chromium.

This might also (not as a priority!) remove the need for bundled secrets on Chromium, in fact, as the secret can be stored in storage.

#5 - 09/13/2021 08:25 AM - koszko

> That said, there are several options. Apart from the obvious approach of data: URLs for Chromium and contentScripts.register for Firefox, both browsers support tabs.executeScript (and Manifest V3 has the somewhat similar scripting.executeScript), and somebody has somehow written a contentScripts.register polyfill for Chromium.

Actually, I thought about simply redirecting to an extension-packaged file. For basic functionality we only need 3 such files, with contents that tell the content script to either allow everything, block scripts or block scripts & sanitize <meta>s. For the third option we would need to find a way content script could deal without an immediately-accessible nonce... but this should be achievable. Although I am still suspecting it might be possible to smuggle arbitrary data somehow

As to tabs.executeScript (and the polyfill that depends on it), I don't thing it gives the document_start guarantees we need :/

> This might also (not as a priority!) remove the need for bundled secrets on Chromium, in fact, as the secret can be stored in storage.

With this we're not going to need any secret at all :)

#6 - 09/14/2021 03:59 AM - jahoti

> Actually, I thought about simply redirecting to an extension-packaged file. For basic functionality we only need 3 such files, with contents that tell the content script to either allow everything, block scripts or block scripts & sanitize <meta>s. For the third option we would need to find a way content script could deal without an immediately-accessible nonce... but this should be achievable. Although I am still suspecting it might be possible to smuggle arbitrary data somehow

That's a good idea! Come to think of it, couldn't we use the query component of the file's URL to smuggle arbitrary data? In fact the actual contents of the file wouldn't even really matter then- the full path itself could provide all the necessary information.

> As to tabs.executeScript (and the polyfill that depends on it), I don't thing it gives the document_start guarantees we need :/

Ah! I forgot about that.

> With this we're not going to need any secret at all :)

Which means we won't need any dependencies either! (Not, of course, that it's a significant problem anyway)

#7 - 10/18/2021 08:24 AM - jahoti

Unfortunately, smuggling via redirection of requests seems to open up the risk of fingerprinting; I can't find a useful way to distinguish requests made by content script from requests made by pages :'(.

#8 - 10/18/2021 10:09 AM - koszko

jahoti wrote:

> Unfortunately, smuggling via redirection of requests seems to open up the risk of fingerprinting; I can't find a useful way to distinguish requests made by content script from requests made by pages :'(.

I thought the "universal" way would be to keep track of what tabs and frames have already made such request and only redirect the first such request in frame's lifetime, blocking subsequent ones.

I did, however, hope there could be some simpler solution. When I tried it seemed that some fields of the details object were different in cases the AJAX call was made by content script. Or have you already found that this is not reliable enough?

#9 - 10/19/2021 07:22 AM - jahoti

> I did, however, hope there could be some simpler solution. When I tried it seemed that some fields of the details object were different in cases the AJAX call was made by content script. Or have you already found that this is not reliable enough?

Not at all- the documentation, insofar as I can tell, doesn't mention such behavior. When there's time I'll test and see if there's any information available on what happens.

#10 - 11/19/2021 06:08 PM - koszko

- Status changed from New to In Progress

- Assignee set to koszko

jahoti wrote:

> > I did, however, hope there could be some simpler solution. When I tried it seemed that some fields of the details object were different in cases the AJAX call was made by content script. Or have you already found that this is not reliable enough?
>
> Not at all- the documentation, insofar as I can tell, doesn't mention such behavior. When there's time I'll test and see if there's any information available on what happens.

I have an even better idea!

When page makes an XmlHttpRequest to our special (chrome-)extension:// URL, we need to block it to avoid fingerprinting. When content script makes the call, we need to somehow return information whether scripts on given page should be blocked and, if applicable, what the nonce is.
We could block the call always when scripts are to be allowed and redirect it in other cases. This will block XmlHttpRequests a script-enabled page makes. When scripts are disallowed, page cannot do XmlHttpRequests, so fingerprinting will be impossible.

There's a loophole: the user might have a page opened with scripts allowed and then change settings to block scripts. Until the page gets reloaded, it will be able to fingerprint.
We could try to make some workarounds here but I'd consider it lower priority - risk is low here and some particular circumstances must occur for fingerprinting to work - i.e. someone who surfs the web with scripts always disabled will not be vulnerable to this. Mozilla users won't be either because of randomization of extensions' ids

EDIT: It won't work after all because the details object available in onBeforeRequest does not provide the full, unspoofable URL of the page that initiates the request...

#11 - 11/19/2021 07:07 PM - koszko

You know what? I'd leave the fingerprinting issue for now. The vunlerability doesn't exist in Mozilla browsers and can be mitigated in Manifest V3 under Chromium which we're going to support anyway. It might be more practical to release more quickly and just warn users of this

#12 - 11/20/2021 06:32 PM - koszko

- % Done changed from 0 to 50

The commit that removes smuggling via cookies and employs synchronous XHR for the job is now on the koszko branch. This temporarily breaks Mozilla port

#13 - 11/30/2021 02:27 AM - jahoti

> I'd leave the fingerprinting issue for now. The vunlerability doesn't exist in Mozilla browsers and can be mitigated in Manifest V3 under Chromium which we're going to support anyway.

Dynamic content scripts can also be used in both cases, which may be necessary (seeing as Mozilla doesn't offer effective synchronous AJAX).

#14 - 02/18/2022 12:35 PM - koszko

- Status changed from In Progress to Closed

- % Done changed from 50 to 100

In master