Haketilo - Bug #53

Interference with existing CSP headers

07/17/2021 09:09 AM - jahoti

Status:	Closed	**Start date:**	07/17/2021
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

Current handling of pre-existing CSP headers needs to be refined:

- Pre-existing http-equiv embeds and actual headers (on whitelisted pages) can block the nonce we use
- Completely stripping headers for blacklisted URLs can remove useful directives (e.g. limitations on stylesheets or multimedia)

Related issues:

Related to Haketilo - Feature #15: make sure page's own csp in <head> doesn't...	**Closed**	**07/01/2021**

History

#1 - 07/21/2021 11:48 PM - jahoti

- Assignee set to jahoti

Currently working on this (albeit somewhat slowly).

#2 - 07/24/2021 12:12 AM - jahoti

- Status changed from New to In Progress

- % Done changed from 0 to 30

A fix is now implemented by parsing CSP headers for direct handling, which also allows removing of directives that report CSP violations (for privacy) and blocking of prefetching/prerendering.

However, testing is still occurring to ensure this everywhere works as intended.

#3 - 07/25/2021 09:26 AM - jahoti

- % Done changed from 30 to 80

Patch committed; awaiting acceptance/rejection from master. While it's difficult to be fully confident it's clear, as there's many moving parts and I don't know the full details of how CSP is implemented, the current implementation seems to cover all possibilities.

#4 - 07/25/2021 09:31 AM - jahoti

- % Done changed from 80 to 70

The patch awaiting merge still doesn't address the CSP we inject *into* the page on Chromium, however. That will require further thought.

#5 - 07/26/2021 12:13 PM - koszko

> The patch awaiting merge still doesn't address the CSP we inject *into* the page on Chromium, however. That will require further thought.

I just notices one possible problem: what if Mozilla caches headers across browser sessions? If so, our "signing" of headers is not going to work, as the secret used changes :/ This needs checking.

As to CSP violation report blocking - should we do that unconditionally? Perhaps there are some legitimate use cases for that? Or no? How about we only block reports when we also block scripts? And maybe later add some switch to give users more fine-grained control?

Until the feature is somewhat complete, I think there's yet no need for merging into master. How about we continue in the csp-PoC branch I created?

You don't have to merge things that appear in origin/koszko if they're not connected to the feature you're working on. You can safely wait until I merge that into master and then only merge master into your baranch(es). In the new PoC branch I excluded these commits from koszko. This way git diff gives much more informative output :)
https://git.koszko.org/browser-extension/diff/?h=csp-PoC&id=97b8e30fadf0f1e1e0aeb9078ac333026d735270&id2=081739e7dcaeed1e0655a49ebdebb02d653d9042

Btw, (Ungoogled) Chromium prints something like this in the console:

```
The Content-Security-Policy directive 'prefetch-src' is implemented behind a flag which is currently disabled.
```

So CSP-based blocking of prefetch might not work here :/

#6 - 07/27/2021 06:45 AM - jahoti

> I just notices one possible problem: what if Mozilla caches headers across browser sessions? If so, our "signing" of headers is not going to work, as the secret used changes :/ This needs checking.

That's a good question; I will see if I can test whether that happens! Hopefully not X.

> As to CSP violation report blocking - should we do that unconditionally? Perhaps there are some legitimate use cases for that? Or no? How about we only block reports when we also block scripts? And maybe later add some switch to give users more fine-grained control?

I forgot I had implemented that! To be honest it's just a guess that it's an antifeature, and unless somebody can offer insight into what actual use cases it's supposed to have, it should probably not be on when scripts aren't blocked. For when they *are*, it remains to be checked whether CSP violation reports pose any danger.

In the next pull request I'll try and put some sensible limits on that, and in future try to keep better records of what is only partially finished :).

> Until the feature is somewhat complete, I think there's yet no need for merging into master. How about we continue in the csp-PoC branch I created?
>
> You don't have to merge things that appear in origin/koszko if they're not connected to the feature you're working on. You can safely wait until I merge that into master and then only merge master into your baranch(es). In the new PoC branch I excluded these commits from koszko. This way git diff gives much more informative output :)
> https://git.koszko.org/browser-extension/diff/?h=csp-PoC&id=97b8e30fadf0f1e1e0aeb9078ac333026d735270&id2=081739e7dcaeed1e0655a49

ebdebb02d653d9042

Thank you for clarifying these things! I'm still very new to Git and collaborative development; hopefully it isn't too annoying for you :/.

> Btw, (Ungoogled) Chromium prints something like this in the console:
>
> The Content-Security-Policy directive 'prefetch-src' is implemented behind a flag which is currently disabled.
>
> So CSP-based blocking of prefetch might not work here :/

We might not even end up needing to block prefetch/prerendering anyway, assuming that extensions get applied properly to prerendered pages. I'll have a look at what flag might be involved in any case.

#7 - 07/27/2021 11:30 AM - koszko

> As to CSP violation report blocking - should we do that unconditionally? Perhaps there are some legitimate use cases for that? Or no? How about we only block reports when we also block scripts? And maybe later add some switch to give users more fine-grained control?

> I forgot I had implemented that! To be honest it's just a guess that it's an antifeature, and unless somebody can offer insight into what actual use cases it's supposed to have,

To detect cross-site-scripting attack attempts.

> it should probably not be on when scripts aren't blocked. For when they *are*, it remains to be checked whether CSP violation reports pose any danger.

Actually, when scripts are blocked, allowing CSP reports would make no sense because it would be violations of our rules and not site's that would be reported. I think I've even seen some code in NoScript for blocking these reports. It involved using some API to add a listener for an event. Hopefully, we'll be able to achieve the same more simply with HTTP headers...

> Thank you for clarifying these things! I'm still very new to Git and collaborative development; hopefully it isn't too annoying for you :/.

It's OK. I also need to practice interactive rebase (maybe even learn other facilities, too?)

> Btw, (Ungoogled) Chromium prints something like this in the console:
>
> The Content-Security-Policy directive 'prefetch-src' is implemented behind a flag which is currently disabled.

> So CSP-based blocking of prefetch might not work here :/

> We might not even end up needing to block prefetch/prerendering anyway, assuming that extensions get applied properly to prerendered pages.

As to prerendering, that's what I hope for. As to prefetch, it would be good to block it anyway, to avoid needless script downloads. However, it might be possible to stop it by removing its respective tag with MutationObserver under Chromium.

> I'll have a look at what flag might be involved in any case.

I thought it might be something under chrome://flags/. I did not, however, find anything relevant there. Anyway, if we find a solution that would require every user to go to chrome://flags/ and change some setting there, then it's not a solution at all :/

#8 - 07/27/2021 12:03 PM - jahoti

Firstly, header-signing is working OK on Mozilla. While headers are cached across sessions, the secret is too; unless the user somehow manages to clear the cache of extension codes or whatever they are without touching the standard cache, nothing will break.

> and unless somebody can offer insight into what actual use cases [CSP violation reporting]'s supposed to have,
> To detect cross-site-scripting attack attempts.

Ah- thank you! That wouldn't seem to be an antifeature :).

> it should probably not be on when scripts aren't blocked. For when they are, it remains to be checked whether CSP violation reports pose any danger.
> Actually, when scripts are blocked, allowing CSP reports would make no sense because it would be violations of our rules and not site's that would be reported. I think I've even seen some code in NoScript for blocking these reports. It involved using some API to add a listener for an event. Hopefully, we'll be able to achieve the same more simply with HTTP headers...

It could be either ours or the sites- only the relevant directives for our blocking are changed in CSP headers.

That said, by putting injected CSP rules in a separate header and just removing any that conflict from the ones the site provides, I think we might (hopefully) get the perfect result of only having actual violations reported.

> As to prerendering, that's what I hope for. As to prefetch, it would be good to block it anyway, to avoid needless script downloads. However, it might be possible to stop it by removing its respective tag with MutationObserver under Chromium.

That possibility is what I'm counting on, as well as the situation for prerendering. We shall have to see...

> I thought it might be something under chrome://flags/. I did not, however, find anything relevant there. Anyway, if we find a solution that would require every user to go to chrome://flags/ and change some setting there, then it's not a solution at all :/

It's rumoured to be the #enable-experimental-web-platform-features flag, which is somehow even worse.

#9 - 07/27/2021 01:01 PM - koszko

> Actually, when scripts are blocked, allowing CSP reports would make no sense because it would be violations of our rules and not site's that would be reported. I think I've even seen some code in NoScript for blocking these reports. It involved using some API to add a listener for an event. Hopefully, we'll be able to achieve the same more simply with HTTP headers...

> It could be either ours or the sites- only the relevant directives for our blocking are changed in CSP headers.

Ok... However, in that case a violation itself could be a (possibly indirect) result of our actions, so I'd rather we didn't report it anyway.

#10 - 08/13/2021 05:17 PM - koszko

- Related to Feature #15: make sure page's own csp in <head> doesn't block our scripts added

#11 - 08/13/2021 05:23 PM - koszko

From what I tested today and yesterday[1], the experimental code in csp-PoC that's responsible for removing the CSP <meta> tag does not lift the once-imposed rules. How about we revert that change in content/main.js (then test the rest of what's on that branch) and merge the branch to master? I think proper fixes for related issue 15 can wait a little :)

[1] https://hachettebugs.koszko.org/issues/15#note-2

#12 - 08/14/2021 02:25 AM - jahoti

> From what I tested today and yesterday[1], the experimental code in csp-PoC that's responsible for removing the CSP tag does not lift the once-imposed rules. How about we revert that change in content/main.js (then test the rest of what's on that branch) and merge the branch to master? I think proper fixes for related issue 15 can wait a little :)

Fair point- I'll do that now! The rest had already been subject to some light testing, which makes things easier also.

#13 - 08/14/2021 10:10 AM - koszko

- Status changed from In Progress to Closed

- Assignee deleted (jahoti)

- % Done changed from 70 to 100

Merged to master. You no longer need the csp-PoC branch, do you?

#14 - 08/15/2021 09:08 AM - jahoti

No- feel free to delete the csp-PoC branch.