Haketilo - Feature #45

Feature # 33 (New): Add more possibilities of page URL matching

Add a universal wildcard for URLs

07/02/2021 09:06 AM - jahoti

Status:	New	**Start date:**	07/02/2021
Priority:	Low	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

Potentially something to consider carefully, as it is obviously open to misuse, a way to signal a script should run on every page might be useful in some cases (e.g. a cloudflare e-mail decryptor).

History

#1 - 07/07/2021 12:54 AM - jahoti

- Parent task set to #33

#2 - 11/27/2021 10:37 AM - koszko

- Priority changed from Normal to Low

This is paradoxally a complex issue. We currently have different ways of handling pages that are to

1. have their own scripts allowed
2. have their own scripts blocked
3. have their own scripts blocked and replaced with Haketilo-supplied ones

In case 1 we don't modify CSP at all. In case 2 we only add our own script-blocking CSP rule. In case 3 we remove page's own CSP and replace it with our own which allows our scripts to run.

Merely adding support for a special pattern matches all urls would cause all pages to fall into category "3" which would defeat the purpose of the "block scripts by default" toggle we currently have.

We could instead consider defining a separate kind of script that can run on pages without a specific payload matched as well as *alongside* the main scripts injected to a page (is this what you meant?). However, to be able to reliably inject a script to all pages would require us to also remove pages' CSP in cases 2 and 1 which I consider too bad for security.

When discussing the universal wildcard, I assume we don't want to target case 1 pages anyway, so from now on I shall only consider the issue for case 2. It could be possible to modify case 2 pages' existing CSP rules to also allow for our scripts (and maybe other content we need) to be injected, and we even had something similar in Haketilo's code at some point. The problem is, I removed that feature because it seemed too unreliable.
We could also rely on facilities like userScripts.register() or Chromium's CSP exemptions of elements added by content scritps (would we find a fall-back for older Mozilla browsers?). This, however, doesn't guarantee that the injected script will succeed in modifying the page in a certain way. If it adds an image, for example, the image might still get blocked by CSP.

I think the best action we can take now is to put this idea aside for some time and instead create fixes (like Cloudflare email decryptor or Google reCAPTCHA client) in a way they do nothing on pages that don't use the relevant nonfree tool. Then, we can make one resource that depends on all those and tell people to manually designate it as a payload for sites that don't have any other matching payload yet - with hopes that this catch-all fix will do

#3 - 11/30/2021 04:40 AM - jahoti

> I think the best action we can take now is to put this idea aside for some time and instead create fixes (like Cloudflare email decryptor or Google reCAPTCHA client) in a way they do nothing on pages that don't use the relevant nonfree tool. Then, we can make one resource that depends on all those and tell people to manually designate it as a payload for sites that don't have any other matching payload yet - with hopes that this catch-all fix will do

That should work perfectly well actually; it operates in line with the general system for executing rules, and (ideally) means fix creators will run only the general-purpose fixes actually needed.