Haketilo - Feature #25

stop always using the same script nonce on given https(s) site

07/01/2021 12:52 PM - koszko

Status:	Closed	**Start date:**	07/01/2021
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

Other protocols are of no interest since they're not supported by WebRequest API. For HTTP(s), we could make things more secure by using a different nonce each time a page is loaded (well, that's what nonce is meant for...). This will of course require keeping track of used nonces and passing them though messages to content scripts which makes this quite a challenging feature.

History

#1 - 07/12/2021 03:05 AM - jahoti

- Assignee set to jahoti

#2 - 07/12/2021 07:08 AM - jahoti

- % Done changed from 0 to 80

Patch awaiting acceptance/rejection: testing on Chromium is *critical*, as there is a potential (albeit improbable) race between nonce access by the onHeadersReceived listener and by the content script.

P.S. there are two unapproved commits preceding the main patch, the first totally unrelated to this patch; I am happy to remove it if you want.

#3 - 07/12/2021 07:13 AM - jahoti

- Status changed from New to Feedback

#4 - 07/12/2021 02:35 PM - koszko

- Status changed from Feedback to In Progress

Merged into master. Honestly, I am neutral towards that unrelated patch.

I think we should also add some way to forget the nonces that are not going to be used anymore (for example because some tab got closed). We could use tabs.onUpdated[1]. This is not critical, though. Nonces are to short to make us run out of memory in a reasonable time.

Also, have you thought about deriving HTTP(s) nonce from url, tab id and frame id? This way we would not need to store any values.

[1] https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/Tabs/onUpdated

#5 - 07/13/2021 12:01 AM - jahoti

> I think we should also add some way to forget the nonces that are not going to be used anymore (for example because some tab got closed). We could use tabs.onUpdated[1]. This is not critical, though. Nonces are to short to make us run out of memory in a reasonable time.

Most definitely- that you for pointing out that API! I will get to work on that immediately.

> Also, have you thought about deriving HTTP(s) nonce from url, tab id and frame id? This way we would not need to store any values.

Admittedly not; would that be sufficiently unpredictable? Getting the same url/tab id/frame id combination is quite common (particularly when reloading a page after changing permissions), if that's a concern. However, it is definitely a simpler and more efficient than the current setup if achieveable.

#6 - 07/13/2021 11:38 AM - koszko

> Also, have you thought about deriving HTTP(s) nonce from url, tab id and frame id? This way we would not need to store any values.

> Admittedly not; would that be sufficiently unpredictable? Getting the same url/tab id/frame id combination is quite common (particularly when reloading a page after changing permissions), if that's a concern. However, it is definitely a simpler and more efficient than the current setup if achieveable.

Yes, it is common. Probably not a good idea after all, forget it.

However, one more thing came to my mind. When rewriting headers, we could also smuggle the random nonce (or better - random salt for it, to use for nonce deriving together with tab id and url) in the url.

#7 - 07/13/2021 11:46 AM - jahoti

> However, one more thing came to my mind. When rewriting headers, we could also smuggle the random nonce (or better - random salt for it, to use for nonce deriving together with tab id and url) in the url.

Good idea! The only question is how to fit it alongside the smuggled whitelisting code; do you have a possible scheme?

#8 - 07/13/2021 12:22 PM - koszko

> only question is how to fit it alongside the smuggled whitelisting code; do you have a possible scheme?

The whitelisting code itself is not really needed for HTTP(s) (but we could include it in order to apply additional blocking in parallel to CSP-based - for better security).

In general, after `#' we can have the unique value used to authenticate the injected string, followed by settings serialized in some way (they should not be visible to the page anyway). It could even be an object expressed in JSON and escaped using encodeURIComponent. In the future, if need arises, we can smuggle even more settings this way. We could also use some more compact serialization than JSON.

Actually, for better security we could make the unique value itself depend on the salt. We could even make the salt special and exclude it from the JSON object smuggled right after, thus avoiding possibly parsing arbitrary JSON in content scripts context. Again, this should not be needed, but will boost security if done :)

#9 - 07/14/2021 07:28 AM - jahoti

> In general, after `#' we can have the unique value used to authenticate the injected string, followed by settings serialized in some way (they should not be visible to the page anyway). It could even be an object expressed in JSON and escaped using encodeURIComponent. In the future, if need arises, we can smuggle even more settings this way. We could also use some more compact serialization than JSON.
>
> Actually, for better security we could make the unique value itself depend on the salt. We could even make the salt special and exclude it from the JSON object smuggled right after, thus avoiding possibly parsing arbitrary JSON in content scripts context. Again, this should not be needed, but will boost security if done :)

Yes! I'll push the code with gc for nonces, and then do a basic implementation of this. While the details should still be discussed before declaring it finalized- I personally like the idea of a more compact serialization- it should be a start.

#10 - 07/14/2021 09:40 AM - koszko

> While the details should still be discussed before declaring it finalized

It's still possible to make a proof-of-concept, though, even before we decide on details.

The way I see it is:

1. In onBeforeRequest check if the url has proper settings injected. If not, draw a random salt, then inject&redirect.
2. In onHeadersReceived validate the settings injected into URL. Use the extracted salt to compute a nonce and modify headers to set the required CSP. In general, it should be impossible to encounter a URL without settings injected (or with invalid settings) at this point. But if it happens anyway, we could simply inject CSP header blocking all scripts for it, just to be safe.
3. In content scripts, for HTTP(s) pages, validate the settings injected into URL and use the salt to compute nonce. Also, given that we already have the settings ready (including whitelist info), we can perform additional script blocking just as we do for non-HTTP(s) sites.

The way I see it is to transform

```
https://example.com/a/b?c=d#something
```

into

```
https://example.com/a/b?c=d#<salt><unique-value><other-settings>#something
```

where:

- could be for example base64-encoded salt.
- could be for example base64-encoded unique value derived from salt, url and (optionally) tab/frame id. So far I used hex encoding because it that was what sha256.js produces, but base64 would be better long-term. Validation of the entire injected payload would be computing this value and comparing it with the one injected. This unique value seems to have the same properties we expect from our nonce, so we could make both the same...
- would be compactly encoded whitelist info, possibly with other data. This could actually be enrypted, although I don't consider this worth doing. Using base64 would also reduce the compactness here, so it'd be better to just treat the settings with encodeURIComponent() and decodeURIComponent().

We could put dashes or something in between , and , but the first 2 are going to be fixed-length anyway, so there's no need to.

> I personally like the idea of a more compact serialization- it should be a start.

Really? I thought we could do the JSON approach first and optimize it later.

#11 - 07/14/2021 11:27 AM - jahoti

I agree with doing it as a PoC with JSON-encoded settings; that was the idea I meant to communicate, even if (looking back) the implied meaning is exactly the opposite :(.

Unless you get to it first, I'll try implementing it in the next 24 hours.

#12 - 07/14/2021 12:16 PM - koszko

> Unless you get to it first, I'll try implementing it in the next 24 hours.

Go on. I am doing repo stuff right now (maybe with a break to do sth around the application).

Quoting from the coding conventions thread (where it got off-topic):

> > couldn't scripts on whitelisted pages read it from the URL then, in fact?
>
> No, it gets removed from there by content script at document_start, before they get to execute.
>
> Oh- that's actually a neat trick! (IMO anyway) In hindsight it should've been obvious from the fact that the unique value never shows up in the URL bar, yet it's still just as exciting.

It does actually show up, at least under Chromium, but only for a moment. Or rather it used to show up, before I committed the change that added CSP header injection instead of that

#13 - 07/16/2021 09:03 AM - jahoti

> It does actually show up, at least under Chromium, but only for a moment. Or rather it used to show up, before I committed the change that added CSP header injection instead of that

Now that you mention it, there does seem to be a brief flicking in the URL bar; maybe FF does the same thing.

There's now a proof-of-concept implementation in the (badly named) nonce-PoC branch; it seemed appropriate to have a separate branch for something that diverges one commit ago from my branch (two from master) and still needs significant reworking. Regardless, the process of working on it has brought up some points.

Firstly, is there any point in drawing a random salt? It doesn't prevent a replay attack by itself, and banning reuse of salts creates issues with caching (point 4). While there's room for improvement over the current PoC, which uses the (suddently not so) secure salt, it seems just as reasonable to use a fixed string at the beginning of the smuggled data (which also makes it easy to distinguish our strings from actual fragments).

Secondly, the base URL needs to be encoded unambiguously in the smuggled data to catch redirects. The PoC does this by using get_unique(url) as the "unique value"; an alternative method might be to include it in the settings that get sent.

Thirdly, there doesn't seem to be a way to synchrously get the ID of the current tab from a content script- do you know one?

Finally- and I will make an issue for this as it affects the current version also- caching needs to be broken, as it stops us from updating the CSP headers when a page is reloaded.

#14 - 07/16/2021 10:06 AM - koszko

> Firstly, is there any point in drawing a random salt? It doesn't prevent a replay attack by itself

Not if someone manages to learn salt, unique value and nonce. But if someone only learns salt and a unique value? Also, this is probably a good example of why my earlier idea to make the unique value equal nonce is a bad one :)

Yet another idea is to also use the current time when deriving nonce and unique value. Sure, if we use, say, current hour, there could be issues when hour changes and one part of the extension ends up using 13 PM and some other one 12 PM. But we can work around that by additionally checking with the neighboring hour.

Btw, we could even sanitize on-page URLs to block tampering... although this already seems like overeagerness...

> it seems just as reasonable to use a fixed string at the beginning of the smuggled data (which also makes it easy to distinguish our strings from actual fragments).

If we need easy distinguishing (and I am not sure we do), we can add such fixed string before the salt.

> Secondly, the base URL needs to be encoded unambiguously in the smuggled data to catch redirects. The PoC does this by using get_unique(url) as the "unique value"; an alternative method might be to include it in the settings that get sent.

Can you give an example of an ambiguity that could cause problems with the current approach? Also, do I understand correctly that you mean putting the base URL in the settings? What would be the point of this? Wouldn't we then need to verify it against the actual URL?

> Thirdly, there doesn't seem to be a way to synchrously get the ID of the current tab from a content script- do you know one?

I think you're right - there probably isn't one. I believe dropping this one value won't harm the security much, though.

> Finally- and I will make an issue for this as it affects the current version also- caching needs to be broken, as it stops us from updating the CSP headers when a page is reloaded.

I've seen the issue. Actually, I think I recall working around this once[1]. Is the current issue something worse that it was back then?

[1] https://git.koszko.org/browser-extension/commit/?id=12fd4fc3a01eb9718a60c8d04860c4e797049b26

#15 - 07/16/2021 11:32 AM - jahoti

> Firstly, is there any point in drawing a random salt? It doesn't prevent a replay attack by itself
> Not if someone manages to learn salt, unique value and nonce. But if someone only learns salt and a unique value?

In the current PoC that would still let them whitelist the page entirely (assuming they can reload the page with the fragment they want- does the http-equiv refresh support that?).

> Yet another idea is to also use the current time when deriving nonce and unique value. Sure, if we use, say, current hour, there could be issues when hour changes and one part of the extension ends up using 13 PM and some other one 12 PM. But we can work around that by additionally checking with the neighboring hour.

That's perfect! I'll cross-check that against the list of edge cases for time first (it was too disconcerting to see just how hard it is to work properly with it in programs), and then add that straight away.

> Btw, we could even sanitize on-page URLs to block tampering... although this already seems like overeagerness...

It would actually be a good addition, albeit not before we even have a release :).

> If we need easy distinguishing (and I am not sure we do), we can add such fixed string before the salt.

I just realized how you've been checking for smuggled URLs, and it's much more sensible.

> Can you give an example of an ambiguity that could cause problems with the current approach? Also, do I understand correctly that you mean putting the base URL in the settings? What would be the point of this? Wouldn't we then need to verify it against the actual URL?

Ambiguity might have been the wrong word. The problem is that pages keep the fragment when they get redirected, and therefore the settings for the source get applied to the destination e.g. if I have https://somega.me/* whitelisted, click on a link that sends me to https://somega.me/ad, and then that page redirects to https://reallyshadycorp.us/spyware, all the scripts on that page will be allowed to run.

> I've seen the issue. Actually, I think I recall working around this once[1]. Is the current issue something worse that it was back then?
> [1] https://git.koszko.org/browser-extension/commit/?id=12fd4fc3a01eb9718a60c8d04860c4e797049b26

That seems to be related! I'll have a look at how you did it and see if it addresses the current situation as well.

#16 - 07/16/2021 12:05 PM - koszko

> > Firstly, is there any point in drawing a random salt? It doesn't prevent a replay attack by itself
> > Not if someone manages to learn salt, unique value and nonce. But if someone only learns salt and a unique value?
>
> In the current PoC that would still let them whitelist the page entirely

Right, I missed that. How about when someone manages to learn only the nonce and not the salt and unique value?

> (assuming they can reload the page with the fragment they want- does the http-equiv refresh support that?).

I assume this is achievable.

> > Can you give an example of an ambiguity that could cause problems with the current approach? Also, do I understand correctly that you mean putting the base URL in the settings? What would be the point of this? Wouldn't we then need to verify it against the actual URL?
>
> Ambiguity might have been the wrong word. The problem is that pages keep the fragment when they get redirected, and therefore the settings

> for the source get applied to the destination

How come? When you for example go to https://www.koszko.org what ends up in the URL bar is https://koszko.org and that's what content script uses, right? Are we talking about some other kind of redirect?

#17 - 07/17/2021 12:25 AM - jahoti

> > In the current PoC that would still let them whitelist the page entirely
> > Right, I missed that. How about when someone manages to learn only the nonce and not the salt and unique value?

That would be OK- the nonce can be (and is) generated randomly for each request, which means that has no repercussions.

> > Ambiguity might have been the wrong word. The problem is that pages keep the fragment when they get redirected, and therefore the settings for the source get applied to the destination
> > How come? When you for example go to https://www.koszko.org what ends up in the URL bar is https://koszko.org and that's what content script uses, right?

Yes; however, the settings are injected by the background script. Acutally, thinking about it we *could* just have the background script load the settings each time and compare them against anything already present; that might be easier too.

#18 - 07/17/2021 08:33 AM - koszko

jahoti wrote:

> > > In the current PoC that would still let them whitelist the page entirely
> > > Right, I missed that. How about when someone manages to learn only the nonce and not the salt and unique value?

> That would be OK- the nonce can be (and is) generated randomly for each request[...]

And we need either salt or in-background-script caching for that, so in the end we want to use salt (and what we were discussing was whether we can leave it), right?

> Ambiguity might have been the wrong word. The problem is that pages keep the fragment when they get redirected, and therefore the settings for the source get applied to the destination
> How come? When you for example go to https://www.koszko.org what ends up in the URL bar is https://koszko.org and that's what content script uses, right?

> Yes; however, the settings are injected by the background script. Acutally, thinking about it we *could* just have the background script load the settings each time and compare them against anything already present; that might be easier too.

Now, I see that including the base URL in the settings could indeed be needed. So, to clarify, it is/should be like this:

1. https://example.com/a/b?c=d#something is supposed to be opened in a browser tab
2. onBeforeRequest callback redirects it to https://example.com/a/b?c=d#<our-injected-stuff-for-https://example.com/a/b>#something
3. onBeforeRequest callback fires again for the request it just redirected, verifies the payload is valid for https://example.com/a/b and leaves is as it is
4. HTTP redirect to https://example.org/e/f?g=h happens and what the browser now wants to open is https://example.org/e/f?g=h#<our-injected-stuff-for-https://example.com/a/b>#something
5. onBeforeRequest callback fires again, sees the injected stuff is valid for https://example.com/a/b and not https://example.org/e/f and redirects the request to https://example.org/e/f?g=h#<our-injected-stuff-for-https://example.org/e/f>#something
6. onBeforeRequest callback fires for the last time for the request it just redirected, verifies the payload is valid for https://example.org/e/f and leaves it as it is.
7. connection is made and server sends a response
8. onHeadersReceived callback fires, sees the injected stuff is valid, takes salt from it, computes nonce and replaces CSP headers
9. Page loads, content script sees https://example.org/e/f?g=h#<our-injected-stuff-for-https://example.org/e/f>#something and verifies the injected stuff is valid. Once it has the settings extracted from URL, it changes it to https://example.org/e/f?g=h#something and that's what page context scripts are going to see.

So in all this, 5. is where the base URL encoded in settings is needed. Without it, we would have no way of verifying that the injected stuff is valid for some other URL. We would then treat it as something not from our extension and would fail to remove it (which was indeed the case with my initial (later removed) code that did this sort of job).

Turns out you were perfectly right :)

Perhaps it would be possible to use requestId to track requests instead, but injecting stuff into URLs allows us to avoid keeping data in background script variables and is therefore preferable.

> That would be OK- the nonce can be (and is) generated randomly for each request[...]
> And we need either salt or in-background-script caching for that, so in the end we want to use salt (and what we were discussing was whether we can leave it), right?

Just to check, are you arguing for drawing one random value or a salt and, separately, a nonce? If it's the former I apologize- I've been misinterpreting you this whole time trying to correct my own originally faulty logic!

Otherwise I'm still confused- doesn't generating a nonce and just putting it in the settings solve the problem?

> Now, I see that including the base URL in the settings could indeed be needed. So, to clarify, it is/should be like this: [...]
> So in all this, 5. is where the base URL encoded in settings is needed. Without it, we would have no way of verifying that the injected stuff is valid for some other URL. > We would then treat it as something not from our extension and would fail to remove it (which was indeed the case with my initial (later removed) code that did this sort of job).

Right!

> Turns out you were perfectly right :)

It *is* a lot easier to see when the steps you describe are listed out in the console rather than just a one-sentence summary :).

> Perhaps it would be possible to use requestId to track requests instead, but injecting stuff into URLs allows us to avoid keeping data in background script variables and is therefore preferable.

It also doesn't need any significant new additions; by using the base URL in the "signature" we already have information for the job, as well as extra security against signature reuse!

#20 - 07/17/2021 11:23 AM - koszko

> Just to check, are you arguing for drawing one random value or a salt and, separately, a nonce?

I was arguing for drawing a salt and deriving the nonce from salt, URL, time and secret.

> If [...] I apologize

You seem to be apologizing to much ever since we started working together on the project :)

> Otherwise I'm still confused- doesn't generating a nonce and just putting it in the settings solve the problem?

Hey, I didn't think about it! It seems like a perfectly valid idea. This means salt is not that much needed after all.

So let's once again sum up how the injected thing is going to look like:

```
#<unique-value><settings>
```

Where:

- unique value is derived from: secret, time and base url
- settings contain at least: base url, random nonce and whitelist boolean

Is that right?

> I was arguing for drawing a salt and deriving the nonce from salt, URL, time and secret.

That makes sense!

> You seem to be apologizing to much ever since we started working together on the project :)

It started well before then :). I do try to be conscious of it and will do so less in future.

```
Otherwise I'm still confused- doesn't generating a nonce and just putting it in the settings solve the problem
?
```

Hey, I didn't think about it! It seems like a perfectly valid idea. This means salt is not that much needed after all.

> So let's once again sum up how the injected thing is going to look like:
>
> #
>
> Where:
>
> - unique value is derived from: secret, time and base url
> - settings contain at least: base url, random nonce and whitelist boolean
>
> Is that right?

Sounds good! The most recent implementation differs in three respects, however, which gives us the choice to pick and choose some aspects:

- The time isn't used (that is being worked on)
- The base URL isn't sent in the settings; instead, if the unique value doesn't match then the listener assumes it changed
- The settings are included in the unique value

Do you think any of those would be an improvement? The time and settings seem useful to me, while omitting the base URL doesn't (it's much more efficient to read it directly than generate the unique value when it's so obviously going to be wrong).

#22 - 07/17/2021 02:50 PM - koszko

- The base URL isn't sent in the settings; instead, if the unique value doesn't match then the listener assumes it changed

If so, how can we tell the difference between a unique value that doesn't match because of changed (redirected) URL and a unique value that was already part of a link and just happend to have the same format our unique values have? Keep in mind a bad web server might deliberately generate a link with a value of that format, trying to tamper with our extension (and maybe detect whether the client is using this extension).

- The settings are included in the unique value

You mean they are one of the inputs used to derive the unique value? That seems like a good idea!

> it's much more efficient to read it directly than generate the unique value when it's so obviously going to be wrong

Do we actually know we are going to have (or already have) efficiency problems? Or is this just a prediction?

#23 - 07/18/2021 02:41 AM - jahoti

> The base URL isn't sent in the settings; instead, if the unique value doesn't match then the listener assumes it changed
> If so, how can we tell the difference between a unique value that doesn't match because of changed (redirected) URL and a unique value that was already part of a link and just happend to have the same format our unique values have? Keep in mind a bad web server might deliberately generate a link with a value of that format, trying to tamper with our extension (and maybe detect whether the client is using this extension).

Good point- in that case I'll definitely add the base URL to the settings!

> The settings are included in the unique value
> You mean they are one of the inputs used to derive the unique value? That seems like a good idea!

Yes- as it stands currently then we're including the secret, time, and settings (which themselves include the base URL, nonce, and whitelisting state).

> it's much more efficient to read it directly than generate the unique value when it's so obviously going to be wrong
> Do we actually know we are going to have (or already have) efficiency problems? Or is this just a prediction?

It's just a prediction- in fact, given that the hashing algorithm isn't an efficiency problem, it's more just a statement of what could happen. The issue you highlighted above with site-provided "bad signatures" is a great reason to include the base URL anyway, however, which makes that irrelevant.

#24 - 07/21/2021 04:21 PM - koszko

- Status changed from In Progress to Closed

- Assignee deleted (jahoti)

- % Done changed from 80 to 100

Ok, this has been merged yesterday