Filtering CSP entries to prevent bypassing rules #418

Closed **mvgijssel** opened this issue on Jan 21, 2020 · 8 comments

mvgijssel commented on Jan 21, 2020

Consider the following Rails controller action which overwrites the frame ancestors based on some user input:

```
def show
  user_input_domain1 = URI.parse "https://google.com;script-src"
  user_input_domain2 = URI.parse "https://*;.;"
  user_input_domains = [user_input_domain1, user_input_domain2]

  override_content_security_policy_directives(frame_ancestors: whitelisted_domains)
end
```

This results into the following response header:

```
frame-ancestors: https://google.com;script-src *;
```

This shows unexpected output, because by setting the frame ancestors the user is able to change the `script-src` opening possibilities for XSS.

One solution to this would be to filter out the CSP rules inside of specific CSP rules:

```
frame_ancestor = value.gsub('script-src','').gsub('img-src', '')
```

2

oreoshake commented on Jan 21, 2020 Contributor

Thanks for the report **@mvgijssel**. This is something **@gregose** had brought up a loooong time ago wrt to policy "injection" via semicolons.

I think the better approach would be to escape or raise errors upon seeing a semicolon mid-directive. I believe it would be better to raise an error but that could be breaking change, requiring a deprecation and major version bump.

It's easy to say "don't do that" but what you appear to be doing is absolutely supported. Accepting user input into a policy seems to bypass the whole point of CSP, but sometimes a single non-perfect opt-out can help make the other 99% of the application safer until things can be rearchitected.

I should have fixed this a looooooong time ago, tech debt!

oreoshake commented on Jan 21, 2020 Contributor

This might be worth a security advisory as well.

bwillis commented on Jan 21, 2020

> It's easy to say "don't do that" but what you appear to be doing is absolutely supported. Accepting user input into a policy seems to bypass the whole point of CSP, but sometimes a single non-perfect opt-out can help make the other 99% of the application safer until things can be rearchitected.

In the end your call, but it's surprising to me to be able to set a different CSP directive when you are explicitly stating `frame_ancestors`. I would expect that the library would prevent you from abusing it in this way, regardless of where the input is coming from. Do you have any examples of a legitimate reason to use it in this way?

oreoshake commented on Jan 21, 2020 Contributor

I 💯 agree that this behavior should be fixed. There's no legitimate reason that I can think of.

4

oreoshake added a commit that referenced this issue on Jan 21, 2020

`escape semicolons by replacing them with spaces` ··· `3c4b86e`

This was referenced on Jan 21, 2020

Escape semi colons in directive source lists #419

Merged

Escape semi colons in directive source lists in 3.x releases #420

Merged

oreoshake added a commit that referenced this issue on Jan 21, 2020

`escape semicolons by replacing them with spaces` ··· `ddec695`

oreoshake mentioned this issue on Jan 21, 2020

escape semicolons by replacing them with spaces for 5.x line #421

Merged

oreoshake added a commit that referenced this issue on Jan 21, 2020

`escape semicolons by replacing them with spaces` ··· `e4075d5`

oreoshake commented on Jan 21, 2020 — Contributor

I'm just waiting on the backport to finish building. I'll push the fixed versions for 3.x, 5.x, and 6.x and issue an advisory after.

bwillis commented on Jan 21, 2020

Thanks for jumping on this fix so fast **@oreoshake**! 🎉

oreoshake commented on Jan 21, 2020 — Contributor

I published two advisories today. One for this, and one for newline injections https://github.com/twitter/secure_headers/security/advisories

🎉 2

oreoshake closed this as completed on Jan 21, 2020

---

mvgijssel commented on Jan 22, 2020 — Author

Thanks **@oreoshake**!

oreoshake mentioned this issue on Jan 22, 2020

Move semicolon/newline handling to validation and raise errors #422

Closed

longaraymatheus mentioned this issue on Jan 23, 2020

Bump secure_headers from 3.6.7 to 3.9.0 marcelorcorrea/falae#81

Merged

This was referenced on Aug 17

Semantically parse source expressions. #497

Closed

Semantically parse and deduplicate source expressions #498

Closed

eleabrton added a commit to eleabrton/secure_headers that referenced this issue on Aug 24

`escape semicolons by replacing them with spaces` ··· `d594f18`

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

3 participants