

Managing content security and samesite cookies

@oreoshake 

@ndm 





What is content security policy

Content-Security-Policy:

```
default-src 'none';  
base-uri 'self';  
block-all-mixed-content;  
connect-src 'self' uploads.github.com...;  
font-src github.githubassets.com;  
form-action 'self' github.com gist.github.com;  
frame-ancestors 'none';  
frame-src render.githubusercontent.com;  
img-src 'self' data: github.githubassets.com ...;  
manifest-src 'self';  
media-src 'none';  
script-src github.githubassets.com;  
style-src 'unsafe-inline' github.githubassets.com;
```





What is content security policy

Content security policy is hard



What is content security policy



CSP can be managed

Important features:

- Globally defined policy
- Dynamic policies
- Familiar, documented APIs

Model View Controller

Model
is the data



Controller
fetches and
manipulates
data



View
Displays
the data



 `application_controller.rb`

Raw

```
1 class ApplicationController < ActionController::Base
2   def add_csp_exceptions
3     append_content_security_policy_directives(self.class::CSP_EXCEPTIONS)
4   end
5 end
```

 `example_controller.rb`

Raw

```
1 class ExampleController < ApplicationController
2   CSP_EXCEPTIONS = {
3     frame_src: ["foo.com"],
4   }
5   before_action :add_csp_exceptions, only: :show
6
7   def index
8     # doesn't get the additional allowance
9   end
10
11  def show
12    # gets foo.com appended to the current frame_src value
13  end
14 end
```

 `other_controller.rb`

Raw

```
1 class OtherController < ApplicationController
2   def index
3     if feature_enabled?
4       append_content_security_policy_directives(connect_src: [THIRD_PARTY])
5     end
6   end
7 end
```



 `_reused_component.html.erb`

Raw

```
1  <%=
2    append_content_security_policy_directives(
3      frame_src: [PAYMENT_PROCESSOR]
4    )
5  %>
6
7  ... code for payment UI
```

 `inline_script.html.erb`

Raw

```
1  <%= nonced_javascript_tag do %>
2    // static JS goes here, please don't use dynamic javascript.
3  <% end %>
```



 security_headers_dependency.rb

Raw

```
1 module SecurityHeadersDependency
2   SecureHeaders::Configuration.override(:static_file_policy) do |config|
3     config.csp = {
4       default_src: [SecureHeaders::CSP::NONE],
5       script_src: [SecureHeaders::CSP::SELF],
6       style_src: [SecureHeaders::CSP::UNSAFE_INLINE],
7       img_src: [SecureHeaders::CSP::SELF, SecureHeaders::CSP::DATA_PROTOCOL],
8       connect_src: [SecureHeaders::CSP::SELF],
9       form_action: [SecureHeaders::CSP::SELF],
10      base_uri: [SecureHeaders::CSP::SELF],
11    }
12  end
13
14  # Set Content-Security-Policy header for static files
15  #
16  # Returns nothing.
17  def set_static_file_csp
18    SecureHeaders.use_secure_headers_override(request, :static_file_policy)
19  end
20 end
```

 the_controller.rb

Raw

```
1 class TheController < ApplicationController
2   def render_404(exception = nil)
3     set_static_file_csp
4     render "error/404"
5   end
6 end
```





Samesite cookies

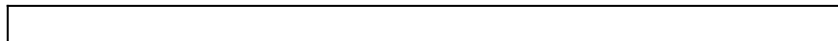
Prevent the browser from sending cookies across sites*,
making CSRF a lot harder





Samesite cookies

```
# Specify cookies SameSite protection level: either :none, :lax, or :strict.  
#  
# This change is not backwards compatible with earlier Rails versions.  
# It's best enabled when your entire app is migrated and stable on 6.1.  
# Rails.application.config.action_dispatch.cookies_same_site_protection = :lax
```





Samesite cookies

Are kinda broken in this weird transition





Samesite cookies

Fairly easy to configure



 controller.rb

Raw

```
1  if current_user&.samesite_lax_enabled?  
2    SecureHeaders.use_secure_headers_override(request, :samesite_lax)  
3  end
```

 security_headers.rb

Raw

```
1  SecureHeaders::Configuration.default do |config|  
2    config.cookies = {  
3      samesite: {  
4        none: { only: ["saml_csrf_token", "saml_return_to"] },  
5      }  
6    }  
7  end  
8  
9  SecureHeaders::Configuration.override(:samesite_lax) do |config|  
10    config.cookies = {  
11      samesite: {  
12        lax: { except: ["saml_csrf_token", "saml_return_to", "saml_csrf_token_legacy", "saml_return_to_legacy"] },  
13        none: { only: ["saml_csrf_token", "saml_return_to"] }  
14      }  
15    }  
16  end
```




```
# persist the relay state digest for future validation
# Cookie attributes are additionally set by app/controller/application_controller/security_headers_dependency.rb
# By default the :saml_csrf_token will have the SameSite=none attribute set from the secure_headers gem.
# The :saml_csrf_legacy cookie does not have the SameSite=none attribute set. This is to accomodate
# browsers that reject or mistreat cookies with the SameSite=none attribute. In particular, there is
# a bug that treats SameSite=None and invalid values as Strict in macOS before 10.15 Catalina and in iOS before 13.
cookies.encrypted[:saml_csrf_token] = saml_csrf_cookie(relay_state.digest)
cookies.encrypted[:saml_csrf_token_legacy] = saml_csrf_cookie(relay_state.digest)
```





May 23, 2019

Maintainer security advisories

We've released maintainer security advisories as a public beta. Maintainer security advisories allow open source maintainers to privately discuss, fix, and publish notices about security vulnerabilities in repositories.

GitHub may additionally create security alerts to all affected downstream repositories as appropriate.

[Learn more about maintainer security advisories](#)

Share

 Twitter

 Facebook

 LinkedIn

Security Advisories

New draft security advisory

Privately discuss, fix, and publish information about security vulnerabilities in your repository's code.

 0 Draft  2 Published  0 Closed

 **Limited header injection when using dynamic overrides with user input**

moderate severity

GHSA-w978-rmpf-qmwg published on Jan 21 by oreoshake

 **Directive injection when using dynamic overrides with user input**

moderate severity

GHSA-xq52-rv6w-397c published on Jan 21 by oreoshake





Questions? Concerns? Comments?

Secure defaults, global configuration, and dynamic overrides are all you need to support rolling out these controls in a **development-friendly**, **reviewable**, and **secure** manner.