21 SafeParamsHelper::safe_params is not so safe

Share:

TIMELINE

vakzz submitted a report to GitLab. Jul 29th (2 years ago)

## Summary

GitLab uses SafeParamsHelper to filter out some keys before passing them to `url_for`:

Code 142 Bytes Wrap lines Copy Download

```
def safe_params
  if params.respond_to?(:permit!)
    params.except(:host, :port, :protocol).permit!
  else
    params
  end
end
```

The issue is that there are a lot more dangerous keys:

Code 234 Bytes Wrap lines Copy Download

```
RESERVED_OPTIONS = [:host, :protocol, :port, :subdomain, :domain, :tld_length,
                    :trailing_slash, :anchor, :params, :only_path, :script_name,
                    :original_script_name, :relative_url_root]
```

This means that anywhere `safe_params` is used, the domain could be changed using the `domain` query. Most of the `build_canonical_path` methods call `url_for(safe_params)` which then gets used by RoutableActions:

Code 536 Bytes Wrap lines Copy Download

```
def ensure_canonical_path(routable, requested_full_path)
  return unless request.get?

  canonical_path = routable.full_path
  if canonical_path != requested_full_path
    if !request.xhr? && request.format.html? && canonical_path.casecmp(requested_full_path) != 0
      flash[:notice] = "#{routable.class.to_s.titleize} '#{requested_full_path}' was moved to '#{canonical_path}'. Please update any links and bookmarks
    end

    redirect_to build_canonical_path(routable)
  end
end
```

This creates an open redirect in all of the `RoutableActions` routes by making `canonical_path != requested_full_path` (eg using a capital letter) and adding the `domain` param:

1. Visit https://gitlab.com/vakzz-h1/Redirect1?domain=aw.rs
2. You will be redirected to https://aw.rs/

The other key that can be abused is `script_name`, as this is appended to the start of the url and can be used to fake a protocol such as javascript:

1. Visit https://gitlab.com/vakzz-h1/redirect1/-/issues?script_name=javascript:alert(1)//
2. Look at the RSS Feed link

   Code 400 Bytes Wrap lines Copy Download

   ```
   <a class="btn btn-svg has-tooltip" data-container="body" title=""  href="javascript:alert(1)//vakzz-h1/redirect1/-/issues.atom?feed_token=XXXX&amp;stat
   <svg class="s16 qa-rss-icon" data-testid="rss-icon">
   <use xlink:href="https://gitlab.com/assets/icons-37f758fe6359f04ae912169432d8ddd9dd45a1316d8fa634996c10bd033e9726.svg#rss"></use>
   </svg>
   </a>
   ```

3. On gitlab.com this is blocked by the CSP

There are a bunch of other places that use `safe_params` that could be exploited such as the _viewer.html.haml

Code 264 Bytes Wrap lines Copy Download

```
- viewer_url = local_assigns.fetch(:viewer_url) { url_for(safe_params.merge(viewer: viewer.type, format: :json)) } if load_async
.blob-viewer{ data: { type: viewer.type, rich_type: rich_type, url: viewer_url, path: viewer.blob.path }, class: ('hidden' if hidden) }
```

This allows an attacker to specify the `viewer_url` for the blob url. Since the json returned by the url has an `html` attributes it allows arbitrary html to be inserted. The below uses https://gitlab.com/-/snippets/1999965 as the viewer url and 1 click csp bypass (same as https://hackerone.com/reports/662287#activity-6026826) with

2. See the injected HTML:

**Code** 258 Bytes  Wrap lines  Copy  Download

```
<form>any <b>html</b> can go <button>here<a data-remote="true" data-method="get" data-type="script" href="https://gitlab.com/-/snippets/1999974/raw" cl
<img width="10000" height="10000">
</a></button></form>
```

3. Clicking anywhere will trigger an alert

I've only skimmed the other locations that use `safe_params` but it looks like there are a few more that load data via javascript or could be turned into open redirects. I also haven't looked into the impact of the open redirects to see if they could be escalated to leak sensitive information, I'll update the report if I find anything else.

I've put all of these in a single report as the mitigation is the same for all of them, but if you would like me to split them into separate reports I can do that as well. I've also set the severity to high due to the number of places that this is used and relative ease of trigger it, but the individual issues are probably less so might need to be adjusted.

**What is the current *bug* behavior?**

`SafeParamsHelper.safe_params` only filters out the keys `:host, :port, :protocol` but there are other dangerous ones

**What is the expected *correct* behavior?**

`SafeParamsHelper.safe_params` should filter out all of the reserved options:

**Code** 234 Bytes  Wrap lines  Copy  Download

```
RESERVED_OPTIONS = [:host, :protocol, :port, :subdomain, :domain, :tld_length,
                    :trailing_slash, :anchor, :params, :only_path, :script_name,
                    :original_script_name, :relative_url_root]
```

**Output of checks**

This bug happens on GitLab.com

### Impact

- open redirect on quire a few routes
- reflected xss using the `javascript` protocol
- reflected xss with csp bypass using the blob viewer

OT: gitlab-securitybot posted a comment. Jul 29th (2 years ago)

Hi @vakzz,

Thank you for submitting this report! We will investigate the issue as soon as possible, and should get back within a week.

Please do not submit your report or ask about its status through additional channels, as this unnecessarily binds resources in the security team.

Best regards,
GitLab Security Team

dcouture GitLab staff changed the status to **Triaged**. Jul 29th (2 years ago)

Hello @vakzz,

Thank you for submitting this report.

We have verified this finding and have escalated to our engineering team. We will be tracking progress internally at https://gitlab.com/gitlab-org/gitlab/-/issues/232829. This issue will be made public 30 days following the release of a patch.

As you noted we'll need some time to fully asses the severity of this report, but for now I'll reward the initial $1000 on triage for a `High` severity report. Congratulations! We'll come back to you with more details/instructions related to splitting the report if that is needed when we analyze it further.

We will continue to update you via HackerOne as a patch is scheduled for release.

I'm also looking forward to your talk at hacktivitycon. :)

Best regards,
Domninic
GitLab Security Team

itLab rewarded vakzz with a **$1,000** bounty. Jul 29th (2 years ago)

Initial $1000 on triage

OT: gitlab-securitybot posted a comment. Aug 29th (2 years ago)

ETA for fix:

Hi @vakzz,

The issue you reported is currently scheduled to be fixed by 2020-09-30.

Thank you again for contacting us!

vakzz posted a comment. Sep 15th (2 years ago)

Hi @dcouture,

Sorry I haven't looked into this much more since the initial report. There were no routes that immediately jumped out where this could be used to exfiltrate a token or sensitive information, unless you have found anything internally :)

Cheers,
Will

BOT: gitlab-securitybot posted a comment. Sep 29th (2 years ago)

ETA for fix:

Hi @vakzz,

The issue you reported is currently scheduled to be fixed by 2020-10-31.

Thank you again for contacting us!

Best regards,
GitLab Security Team

dcouture GitLab staff closed the report and changed the status to ○ **Resolved**. Oct 6th (2 years ago)

Hi @vakzz,

Thank you again for the report! Your finding has been patched in GitLab version 13.4.2 and we are awarding a bounty. Congratulations!

Please let us know if you find that our patch does not mitigate your finding. Your report will be published in 30 days in GitLab's issue tracker.

We look forward to your next report!

Best regards,
Dominic
GitLab Security Team

GitLab rewarded vakzz with a **$3,000** bounty. Oct 6th (2 years ago)

dcouture GitLab staff requested to disclose this report. Nov 2nd (2 years ago)

vakzz agreed to disclose this report. Nov 2nd (2 years ago)

This report has been disclosed. Nov 2nd (2 years ago)