

DOS via issue preview and markdown preview

[HackerOne report #1543718](#) by legit-security on 2022-04-18, assigned to [@kmorrison1](#):

There were two additional reports by the same team which, since this was imported first, are being treated as duplicates due to [sharing the same root cause](#)

- Unauthed via Markdown Preview API: <https://hackerone.com/reports/1544465>
- REDoS via create/update issue: <https://hackerone.com/reports/1543606>

A fourth report, DoS via moving an issue, has a separate cause and separate issue: [DOS via move issue \(#362379 - closed\)](#) / <https://hackerone.com/reports/1543584>

[Report](#) | [How To Reproduce](#)

Report

Summary

Unauthenticated requests to the markdown preview API `/api/v4/markdown` and authenticated previewing of an issue with a specially-crafted description results in high CPU usage for 60 seconds (request timeout).

Multiple requests can be issued in parallel to create a larger impact.

Steps to reproduce

1. Given an authorized user (on GitLab.com - anyone can self-register. On EE - depends on instance configuration).
2. Create an issue with the following description (provided a one-line python script to avoid bloating):
3. Hit the preview button.

Steps 2&3 can be accomplished via the `preview_markdown` API endpoint.

The script:

```
python -c "print('!' * int(1048576 / 3 - 1) + '\n')"
```

Note: this is essentially the maximal description size, but a smaller number of repetitions works too.

Impact

After 60 seconds (timeout) - the request fails.

Meanwhile, on the server end, (a single) CPU is burnt out (verified against a local EE instance).

Issuing multiple requests in parallel results in multiple CPUs burn out.

Using the DockerHub image, the entire server is completely unavailable by repeatedly sending a small number of requests repeatedly.

Examples

The bug is instance-independent, works on latest versions. Since GitLab.com is open-core - it would work on GitLab too.

What is the current *bug* behavior?

The HTTP request fails for timeout while the server is burning CPU.

On the code side:

`texts_and_contexts` is being initialized here:

```
def analyze(text, context = {})  
  [@[texts_and_contexts << { text: text, context: context }  
  end
```

It is then used at `banzai/reference_extractor.rb`:

```
def html_documents  
  ...  
  [@[html_documents ||= Renderer.cache_collection_render([@[texts_and_contexts)  
  ...
```

The CPU utilization is found in the execution of `cache_collection_render`.

What is the expected *correct* behavior?

Fix the implementation of `cache_collection_render`.

Relevant logs and/or screenshots

Output of checks

Results of GitLab environment info

Impact

A complete denial of service of a GitLab EE instance.


As this vulnerability impacts GitLab.com, we assume that this vulnerability opens the door for a DDOS attack.


How To Reproduce

Please add [reproducibility information](#) to this section:


- 1.
- 2.
- 3.

Edited 6 months ago by [Nick Malcolm](#)


 Drag your designs here or [click to upload](#).




Tasks  0


No tasks are currently assigned. Use tasks to break down this issue into smaller parts.



Linked items  2

Relates to


 [DOS via move issue](#)
#362379

 15.4  Sep 26 


 [Add authentication to markdown API](#)
#369369

 15.3 


Activity




[GitLab SecurityBot](#) changed due date to June 26, 2022 [6 months ago](#)



[GitLab SecurityBot](#) added [Weakness](#) [CWE-400](#) [bug](#) [vulnerability](#) [type](#) [bug](#) [priority](#) [2](#) [severity](#) [2](#) scoped labels [6 months ago](#)



[GitLab SecurityBot](#) added [HackerOne](#) [security](#) labels [6 months ago](#)




[GitLab SecurityBot](#) [@gitlab-securitybot](#) · [6 months ago](#)

AuthorReporter

[HackerOne comment](#) by legit-security:

p.s.1. This discovery is issued by Gal Ofri & Tor Beer on behalf of Legit Security Ltd. p.s.2. We want to note that any bounty you find this to be eligible for - Legit Security will donate it.



[GitLab SecurityBot](#) [@gitlab-securitybot](#) · [6 months ago](#)

AuthorReporter

[HackerOne comment](#) by magicmouse:

Hi [[@](#)]legit-security,

Thank you for your submission. I hope you are well. Your report is currently being reviewed and the HackerOne triage team will get back to you once there is additional information to share.

Have a great day!

Kind regards, [[@](#)]magicmouse



GitLab SecurityBot @gitlab-securitybot · 6 months ago

Author

Reporter

[HackerOne comment](#) by magicmouse :

Hi [@]legit-security,

This is a duplicate of your other report.

Best regards, [@]magicmouse



GitLab SecurityBot @gitlab-securitybot · 6 months ago

Author

Reporter

[HackerOne comment](#) by princeofpersia :

Reopening for investigation.



GitLab SecurityBot @gitlab-securitybot · 6 months ago

Author

Reporter

[HackerOne comment](#) by princeofpersia :

Hello [@]legit-security,

Thank you for your submission!

We were able to validate your report, and have submitted it to the appropriate remediation team for review. They will let us know the final ruling on this report, and when/if a fix will be implemented. Please note that the status and severity are subject to change.

Thanks, [@]princeofpersia



GitLab SecurityBot @gitlab-securitybot · 6 months ago

Author

Reporter

[HackerOne comment](#) by kmorrison :

Hi [@]legit-security I just tried this in a local installation of 14.10.2-ee and while the preview does take a few seconds to render, the request does not time out. Can you try this in the latest version of GitLab?

Thanks, [@]kmorrison GitLab Security Team



GitLab SecurityBot @gitlab-securitybot · 6 months ago

Author

Reporter

[HackerOne comment](#) by legit-security :

Hi [@]kmorrison, Thank you for your response! We attached a recording of a simple demo - showing the effect with a single issue preview. Please let us know if you would like us to record a demonstration of a complete DOS (502), or send any extra information. Thanks, [@]legit-security

Attachments

Warning: Attachments received through HackerOne, please exercise caution!

- [recording-1652173255753.webm](#)



GitLab SecurityBot @gitlab-securitybot · 6 months ago

Author

Reporter

[HackerOne comment](#) by legit-security :

p.s. In addition to this issue, #1543606 and #1543584, we also opened #1544465 which was not reopened for review. #1544465 shows a similar DOS effect but is exploitable as an unauthenticated user via API calls. We'd appreciate your response on that issue as well.

Thanks, [@]legit-security



Kevin Morrison @kmorrison1 · 6 months ago

Developer

I reproduced this. I was able to temporarily take down my local GL instance by duplicating enough pages and hitting the 'preview' link on all of them. Once the processes timed out my instance became responsive again. An attacker could just continue spawning processes to keep the processor maxed out. I believe the issue is REDOS rather than just the length of the input because when I changed the characters in the string to just 'abc...' the DOS did not happen.



Kevin Morrison removed [priority: 2](#) label 6 months ago

- Kevin Morrison removed severity: 2 label 6 months ago
- Kevin Morrison added severity: 3 scoped label 6 months ago
- Kevin Morrison added priority: 3 scoped label 6 months ago
- Kevin Morrison changed due date to July 18, 2022 6 months ago
- Kevin Morrison added group workspace scoped label 6 months ago
- Kevin Morrison removed group workspace label 6 months ago
- Kevin Morrison added group project management scoped label 6 months ago



GitLab SecurityBot @gitlab-securitybot · 6 months ago

Author Reporter

@gweaver @donaldcook @johnhope @cmxim This issue is ready for triage as per [HackerOne process](#).

About this automation: [AppSec Escalation Engine](#)



Gabe Weaver @gweaver · 6 months ago

Developer

@donaldcook feel free to adjust the milestone as you deem necessary!



John Hope @johnhope · 6 months ago

Developer

@gweaver @donaldcook This looks like it has the same root cause as <https://gitlab.com/gitlab-org/gitlab/-/issues/362558+> and [DOS via move issue \(#362379 - closed\)](#). So it can likely be closed in favour of the latter as well.



Donald Cook @donaldcook · 6 months ago

Developer

Thanks @johnhope. I'll close this one then.

/cc @kmmorrison1



John Hope @johnhope · 6 months ago

Developer

Reopening as I was wrong about their sharing a root cause: [#362379 \(comment 955779025\)](#).



Nick Malcolm @nmalcolm · 6 months ago

Developer

@johnhope thanks for checking! In that case, the [Move Issue issue](#) can be downgraded back to a severity: 3 as originally indicated, whereas this issue becomes the priority: 2 due to it being exploitable by unauthenticated users.

I'll update the priority here, and let you downgrade [DOS via move issue \(#362379 - closed\)](#) if you want to.

They have identified an unauthenticated path to exploit this, via the markdown preview API [/api/v4/markdown](#).

Given that, the CVSS for this is upgraded to [CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H](#) (7.5 High / \$7,640.00 / new bounty range). Privilege Required: None.



John Hope @johnhope · 6 months ago

Developer

/cc @donaldcook as EM for group project management

















Please [register](#) or [sign in](#) to reply




GitLab Bot @gitlab-bot · 6 months ago

Maintainer

Setting label(s) [devops](#) [plan](#) [section](#) [dev](#) based on group [project management](#).

-  **GitLab Bot** added `devops` `plan` `section` `dev` scoped labels 6 months ago
-  **Gabe Weaver** changed milestone to `%15.2` 6 months ago
-  **Donald Cook** added `backend` label 6 months ago
-  **GitLab Bot** added `Accepting merge requests` label 6 months ago
-  **Kevin Morrison** mentioned in issue `#362379 (closed)` 6 months ago
-  **Donald Cook** marked this issue as related to `#362379 (closed)` 6 months ago
-  **Donald Cook** marked this issue as a duplicate of `#362379 (closed)` 6 months ago
-  **Donald Cook** closed 6 months ago
-  **John Hope** reopened 6 months ago
-  **Nick Malcolm** changed due date to June 26, 2022 6 months ago
-  **Nick Malcolm** added `priority: 2` `severity: 2` scoped labels and automatically removed `priority: 3` `severity: 3` labels 6 months ago
-  **Nick Malcolm** changed title from **DOS via issue preview** to **DOS via issue preview and markdown preview** 6 months ago
-  **Nick Malcolm** changed the description 6 months ago
-  **Donald Cook** changed milestone to `%15.1` 5 months ago
-  **Brett Walker** assigned to `@digitalmoksha` 5 months ago
-  **GitLab Bot** `@gitlab-bot` 5 months ago Maintainer

Thanks for working on this @!(confidential)! We've removed the `Seeking community contributions` label to avoid having multiple people working on the same issue.
-  **GitLab Bot** removed `Accepting merge requests` label 5 months ago

 **Brett Walker** `@digitalmoksha` · 5 months ago Maintainer

After some poking around, I'm confident that the initial problem stems from the CommonMark parser itself. Loading up our code with `rails c` and running just our `Banzai::Pipeline::PlainMarkdownPipeline`

```
[1] pry(main)> def measure(&block)
  start = Time.now
  block.call
  Time.now - start
=> :measure
[2] pry(main)> t1 = measure { Banzai::Pipeline::PlainMarkdownPipeline.call('!' * 1000,
=> 0.20019
[3] pry(main)> t1 = measure { Banzai::Pipeline::PlainMarkdownPipeline.call('!' * 100000
=> 15.944566
[4] pry(main)> t1 = measure { Banzai::Pipeline::PlainMarkdownPipeline.call('!' * 300000
=> 121.138789
```

I ran across this issue for the parser, [Pathological input: Deeply nested lists](#), which reminded me that there is a set of test for pathological input, [pathological_tests.py](#).

While our input is an open ended image, `![1]`, the [pathological_tests.py](#) tests for open ended links, `[a`. So trying that in our pipeline works well:

```
[5] pry(main)> t1 = measure { Banzai::Pipeline::PlainMarkdownPipeline.call('!' * 300000, => 0.144444
```

So I think we're going to need to be looking at a fix for [cmark](#). That will, I think, fix our main problem.

Whether or not it completely fixes the problem is unclear. However if I run our full pipeline on open ended links, it's relatively quick:

```
[7] pry(main)> t1 = measure { Banzai::Pipeline::FullPipeline.call('!' * 300000, {project => 3.265583
```

So I think that's a good sign.



Brett Walker @digitalmoksha · 5 months ago

Maintainer

A little more looking at this. It looks like it's not `cmark`, but the `autolink` extension of [cmark-gfm](#).

Edited with better explanation...

When using the `autolink` extension, certain input will cause `cmark-gfm` to consume excessive memory and CPU.

The input used is multiples of `![a` - an unclosed image link. For example `![a![a![a`

Here is the output from several runs:

```
> time ruby -e 'puts "![a" * 300' | cmark-gfm -e autolink
> time ruby markdown_data.rb 300 | cmark-gfm -e autolink
ruby -e 'puts "![a" * 300' 0.11s user 0.13s system 111% cpu 0.208 total
cmark-gfm -e autolink 0.00s user 0.00s system 1% cpu 0.208 total

> time ruby -e 'puts "![a" * 3000' | cmark-gfm -e autolink
ruby -e 'puts "![a" * 3000' 0.11s user 0.13s system 110% cpu 0.216 total
cmark-gfm -e autolink 0.03s user 0.00s system 12% cpu 0.243 total

> time ruby -e 'puts "![a" * 30000' | cmark-gfm -e autolink
ruby -e 'puts "![a" * 30000' 0.11s user 0.13s system 109% cpu 0.215 total
cmark-gfm -e autolink 3.43s user 0.01s system 93% cpu 3.662 total

> time ruby -e 'puts "![a" * 300000' | cmark-gfm -e autolink
ruby -e 'puts "![a" * 300000' 0.11s user 0.13s system 108% cpu 0.222 total
cmark-gfm -e autolink 1426.25s user 4.04s system 99% cpu 23:54.39 total
```

In contrast, enabling the other extensions, except for `autolink`, doesn't exhibit this problem.

```
> time ruby -e 'puts "![a" * 300000' | cmark-gfm -e footnotes -e table -e strikethrough
ruby -e 'puts "![a" * 300000' 0.12s user 0.14s system 109% cpu 0.238 total
cmark-gfm -e footnotes -e table -e strikethrough -e tagfilter -e tasklist 0.09s user
```

Edited by [Brett Walker](#) 5 months ago



Brett Walker @digitalmoksha · 5 months ago


Maintainer

Since we're treating this as a security issue, instead of opening an issue directly on <https://github.com/github/cmark-gfm>, I've used their stated [security policy](#) and submitted the issue to opensource-security@github.com

I do have a fix that is at https://github.com/github/cmark-gfm/compare/master...digitalmoksha:pathological_autolink which I hope I can submit as a PR soon.

Please [register](#) or [sign in](#) to reply

 **Brett Walker** added `workflow` `in dev` scoped label 5 months ago

 **Brett Walker** added `workflow` `blocked` scoped label and automatically removed `workflow` `in dev` label 5 months ago



Brett Walker @digitalmoksha · 5 months ago

Maintainer

This issue is currently blocked waiting on a fix from upstream



Donald Cook @donaldcook · 5 months ago

Developer

[@digitalmoksha](#) Have you received a response from them yet (per [#361982 \(comment 977411334\)](#))?



Brett Walker @digitalmoksha · 5 months ago

Maintainer

[@donaldcook](#) Unfortunately I haven't heard anything from them.

I'm tempted to submit it as a real PR. I personally don't consider it a huge security issue.



Donald Cook @donaldcook · 5 months ago

Developer

[@digitalmoksha](#) Yeah I agree, I think it's okay as a regular PR. [@nmalcolm](#) WDYT?



Nick Malcolm @nmalcolm · 5 months ago

Developer

[@donaldcook](#) [@digitalmoksha](#) are you talking about submitting "a regular PR" to the upstream project? I'd caution against that. This security issue allows an unauthenticated caller to exhaust the resources of a GitLab server. A public upstream PR could give attackers a heads up and allow them to exploit it before we can ship patched versions.

Our first priority is protecting GitLab customers and users, so we need to mitigate the impact here first. Perhaps via a fork or a monkey patch to fix at the root, or some kind of validation higher in the call stack in our own code if easier. See <https://about.gitlab.com/security/disclosure/#disclosure-guidelines-for-vulnerabilities-in-3rd-party-software>

Looks like [@digitalmoksha](#) you've already pushed code in public though? So maybe the cat is out of the bag already.

[@joernchen](#) as our resident expert on working with third parties on disclosures, what are some approaches you've seen in situations like this?



Brett Walker @digitalmoksha · 5 months ago

Maintainer

[@nmalcolm](#) True, I have a fix on my personal fork, though I highly doubt it's being monitored in any way. As there is no way to make a public fork into a private one, I can delete that branch if necessary. All discussions about this problem have been in this confidential issue.

Fixing this ourselves would require not only forking [cmark-gfm](#), the C library, but also forking the [commonmarker](#) gem, which builds on top of [cmark-gfm](#)

Obviously doable, but a huge PITA.

So I can delete the branch and email them a patch file instead. Based on the above link, we are allowing 90 days for them to generate a fix. I think we're about 30 down at the moment.

I'm open to other suggestions 🤔



Nick Malcolm @nmalcolm · 5 months ago

Developer

[@digitalmoksha](#) yeah that is a huge pain! What's the viability of preventing malicious input it higher in the callstack, in our own code?

[@cmaxim](#) your input would be welcome



Brett Walker @digitalmoksha · 5 months ago

Maintainer

It obviously has to be detected before running through the markdown processor. I think we could either inject a new filter into our `PreProcessPipeline` or just update the `TruncateSourceFilter` to regex detect whether there was minimal number of nested unclosed image references (`![a]`). One or two I would consider user error. 5 or more is sloppy and potentially an attack.

If such a pattern was found, I would be inclined to simply return a message like "Unable to render markdown", similar to how the `TruncateSourceFilter` does it.

Linking to [Denial of Service \(ReDoS\) / Catastrophic Backtracking](#) to remind the implementor of these issues (yes, I'm literally completing my OWASP training right now 😊)

The best solution of course is to have it fixed upstream.



Costel Maxim @cmaxim · 5 months ago

Developer

So I can delete the branch and email them a patch file instead. Based on the above link, we are allowing 90 days for them to generate a fix.

oh, I totally forgot about the times when this was the standard approach. 😊

@digitalmoksha Would it be possible to send the patch via email and if it does not get fixed in the next 60 days we apply If such a pattern was found, I would be inclined to simply return a message like "Unable to render markdown", similar to how the `TruncateSourceFilter` does it in the closes security release?

Edited by [Costel Maxim](#) 5 months ago



Brett Walker @digitalmoksha · 4 months ago

Maintainer

@cmaxim That sounds good. I've deleted my personal fork, and sent them the patch file. I'm attaching it here as well

I've reset the due date to Aug 26, two months out from the original due date, and moved the milestone to 15.4. Feel free to adjust if I got those wrong.

📎 [0001-Fix-pathological-case-in-autolink-extension.patch](#)

Edited by [Brett Walker](#) 3 months ago

Please [register](#) or [sign in](#) to reply



Brett Walker changed due date to August 26, 2022 4 months ago



Brett Walker changed milestone to %15.4 4 months ago



Ramya Authappan mentioned in issue [plan#642 \(closed\)](#) 4 months ago



Costel Maxim @cmaxim · 4 months ago

Developer

The reporter would like to do a blog post on this issue:

GitLab markdown DoS - How an external attacker can shut down your software factory

In this article, we'll explore a vulnerability we found in GitLab's markdown engine - commonmarker. The engine is exposed by a REST API that does not require authentication. As a result, an unauthenticated attacker could easily bring down any GitLab server, shutting down production pipelines. Private GitLab servers, which are accessible only within an internal network, can also be exploited by anyone with network access. Rate limiting does not prevent the attack from taking down the service on all GitLab tiers.

How does GitLab handle markdown?

GitLab uses commonmarker as one of its markdown engines. commonmarker is a Ruby gem wrapper for libmark-gfm - a library cloned in GitLab's repository under ext/commonmarker, and its logic is exposed within the gem as commonmarker.so. The library functions as a markdown strings renderer. GitLab exposes Markdown API to all GitLab tiers without permission restrictions. In addition, it is also exposed to authenticated users via issue creation, update, and preview. Deep dive into the markdown API - the code flow

The graph below represents the inner flow within GitLab of issuing a markdown API request, handling it using ruby, and propagating it to the commonmarker.so in C:

Deep dive into commonmarker

The graph below present the commonmarker knowledge needed for understanding the exploit. Every box represent a function and a flow and every arrow navigates to the inner flow of that function, relevant to the exploit.

Exploitation

This gif will present how the exploitation happens within the commonmarker engine, according to the understanding we have acquired in last section

Handling with 1 minute timeout

It's not over yet - this is a potential DoS, but Puma's architecture makes it a little bit tricky: first is the Puma workers. Puma has multiple workers (configurable by the admin), where worker is a thread. We can adjust to that by multiple requests, in accordance to the amount of CPUs/Puma workers on the machine. The other mechanism, is the PumaWorkerKiller, which after a timeout of 1 minute or too much memory used, kills the worker. It doesn't affect the vulnerability exploitation by much, since we can just loop our requests - as each request causes a DoS on the handler thread, when a thread is being killed and resurrected, it will handle our request, being unable to handle anything else, so we can block all the Puma workers - causing a complete DoS. Consequences

An attacker can take advantage of this vulnerability and shut down the software production pipeline of any accessible GitLab server.

As a result, an unauthenticated attacker could easily bring down any GitLab server, shutting down production pipelines.

How about we rephrase this to: As a result, **without proper configuration of Puma workers**, an unauthenticated attacker could easily bring down any GitLab server, shutting down production pipelines.

[@digitalmoksha](#) any updates from the package maintainers?



Donald Cook [@donaldcook](#) · 4 months ago

Developer

[@cmaxim](#) Brett is OOO for the next couple weeks, but as of the end of last week I don't believe he had heard back from the maintainers.



Costel Maxim [@cmaxim](#) · 4 months ago

Developer

Thanks for the update.



Brett Walker [@digitalmoksha](#) · 4 months ago

Maintainer

[@cmaxim](#) I just received a reply from them today. Boilerplate, thank you for reporting, we'll pass it to the right team for further investigation.



Brett Walker [@digitalmoksha](#) · 4 months ago

Maintainer

[@cmaxim](#) after reading the word "unauthenticated" over and over in the proposed post above, I wonder if we shouldn't put that API behind authentication, so that you at least need to be logged in to use it.

I'm not sure how many people may be relying on it being unauthenticated, but I do think it would reduce the attack surface for on-premise who don't allow random people to sign up. And it would allow us to at least tie a user to any abuse for .com.

wdyt?



Costel Maxim [@cmaxim](#) · 4 months ago

Developer

Adding authentication sounds like a good idea. Thanks for reading the blog post.



Brett Walker [@digitalmoksha](#) · 4 months ago

Maintainer

[@donaldcook](#) and I discussed and it seems like a good idea to add the authentication behind a feature flag, defaulted to on, in case we run into a customer situation and we need to turn it off temporarily.



Costel Maxim [@cmaxim](#) · 4 months ago

Developer

Sounds good to me!

Please [register](#) or [sign in](#) to reply



GitLab Bot @gitlab-bot · 4 months ago

Maintainer

👋 @donaldcook and @gweaver,

This `group project management` bug has at most 25% of the SLO duration remaining and is approaching-SLO breach. Please consider taking action before this becomes a `missed-SLO` in 14 days (2022-07-22).



GitLab Bot added `approaching-SLO` label 4 months ago



Gabe Weaver changed milestone to `%15.3` 4 months ago



GitLab Bot added `missed-SLO` label 4 months ago



Brett Walker marked this issue as related to [#369369 \(closed\)](#) 3 months ago



Brett Walker mentioned in issue [#370428 \(closed\)](#) 3 months ago



Brett Walker added `workflow in review` scoped label and automatically removed `workflow blocked` label 3 months ago



GitLab SecurityBot added `security-issue-escalated` label 2 months ago



GitLab SecurityBot @gitlab-securitybot · 2 months ago

Author

Reporter

@gweaver @donaldcook @cmaksim This `severity: 2` `bug vulnerability` issue's milestone has expired.

About this automation: [AppSec Escalation Engine](#)



Gabe Weaver @gweaver · 2 months ago

Developer

@donaldcook @digitalmoksha can we make any progress with this yet?



Costel Maxim @cmaksim · 2 months ago

Developer

This was patched with 15.3.2. Closing issue.



Kevin Morrison @kmorrison1 · 2 months ago

Developer

@cmaksim should we have reserved a CVE for this?



Costel Maxim @cmaksim · 2 months ago

Developer

I think we have. CVE requested at <https://gitlab.com/gitlab-org/cves/-/issues/478>



Kevin Morrison @kmorrison1 · 2 months ago

Developer

Ah ok. That has the wrong info for the researcher who submitted it. I can edit the CVE issue but how do we fix it in the NVD? @cmaksim

Please [register](#) or [sign in](#) to reply



Costel Maxim closed 2 months ago



GitLab SecurityBot removed `security-issue-escalated` label 2 months ago



Brett Walker mentioned in issue [#372891 \(closed\)](#) 1 month ago



Brett Walker @digitalmoksha · 1 month ago

Maintainer

It looks like both `cmark-gfm` and `commonmarker` have been updated with a patch, <https://github.com/github/cmark-gfm/security/advisories/GHSA-cgh3-p57x-9q7q> and <https://github.com/gjtorikian/commonmarker/security/advisories/GHSA-4qw4-jpp4-8gvp>.

I can now move forward on [Update `cmark-gfm` and `commonmarker` gems due ... \(#370428 - closed\)](#), and remove it's confidentiality flag since we applied our own patch for 15.1 and beyond.



GitLab SecurityBot @gitlab-securitybot · 1 month ago

Author

Reporter

@cmaxim - this [HackerOne](#) [bug vulnerability](#) issue was closed 30 days ago and should be made public. Please follow [the process for disclosing security issues](#).

If the issue needs to stay confidential, please add the [keep confidential](#) label.

If you removed confidential data from the issue description before making it public, make sure that the description history entry is deleted.



Nick Malcolm @nmalcolm · 1 month ago

Developer

@cmaxim @kmorrison1 the researchers have requested to [disclose this on H1](#). Please make this public here and on H1 if there isn't a reason to keep it confidential! 🙏

Please [register](#) or [sign in](#) to reply



Kevin Morrison made the issue visible to everyone 1 month ago



GitLab SecurityBot @gitlab-securitybot · 3 weeks ago

Author

Reporter

[HackerOne report #1543718](#) was disclosed on 2022-11-04 @ 03:47.

- Bounty awarded: \$7640

Please [register](#) or [sign in](#) to reply