

35 A deactivated user can access data through GraphQL

Share:     

TIMELINE



maxcar submitted a report to [GitLab](#).

May 11th (2 ye

Summary

A deactivated user should not be able to access information through the API. This rule is not enforced when making requests through the GraphQL endpoint.

When reading through the changelog for 13.11.2 I noticed that the rule for a deactivated user allows for :log_in (as it should) but it is restricted from :access_api(as should) [link](#). The GraphQL endpoint does not seem to use this rules when authorizing a user. I guess GraphQL only checks for api scope on the user.

This opens for three potential problems:

- A user using its account through the GraphQL API (through some script or similar) would not get a warning that the account is deactivated. This could lead to the account being removed if the entities controlling the GitLab instance has any automatic procedures deleting accounts. When reading about the deactivation feature I got the impression that most admins requesting the feature would use it in automated "cleanings" of their user base. I could see how an admin could implement a "deactivate after 90 days inactivity" and "delete after 180 days inactivity" rule or similar. This could lead to an account being "in use" through GraphQL could get deleted without proper warnings.
- An admin could use deactivated accounts as "bots" or "service accounts" bypassing the billing of these accounts. (an admin can create users and deactivate them directly, before ever using the account)
- The fact that the account should not be able to do this. An admin reading the docs are under the assumption that a deactivated account is blocked from using the GraphQL API. An inactive user could have left some form of scripts running that would keep on using resources on the GitLab instance, which I guess the admin would like to remediate by deactivating the account.

as of 13.10.4: A deactivated user can (without activating its account) use read queries on the GraphQL endpoint. The latest security patch removes the ability to use mutations due to the fact that

Code 148 Bytes

[Wrap lines](#) [Copy](#) [Download](#)

```
1 rule { deactivated }.policy do
2   prevent :access_git
3   prevent :access_api
4   prevent :receive_notifications
5   prevent :use_slash_commands
6 end
```

prevents :access_api, and

Code 61 Bytes

[Wrap lines](#) [Copy](#) [Download](#)

```
1 rule { ~can?(:access_api) }.prevent :execute_graphql_mutation
```

prevents from using mutations if I understand the code correctly.

tested on 13.11.1: (Prior to latest security patch 13.11.2) A deactivated user can (without activating its account) use queries and mutations on the GraphQL endpoint

Steps to reproduce

Unlimited service accounts

1. Login as admin
2. Create a user
3. Deactivate the user
4. Create an api token for the deactivated user
5. Use the token in GraphQL requests such as (replacing url and token)

Code 186 Bytes

[Wrap lines](#) [Copy](#) [Download](#)

```
1 curl 'https://gitlab.com/api/graphql' -H 'Accept: application/json' -H 'Content-Type: application/json' -H 'Authorization: Bearer <<TOKEN>>' --data '{
```

User with deactivated account

1. Use any old token from your deactivated account in requests such as

Code 186 Bytes

[Wrap lines](#) [Copy](#) [Download](#)

```
1 curl 'https://gitlab.com/api/graphql' -H 'Accept: application/json' -H 'Content-Type: application/json' -H 'Authorization: Bearer <<TOKEN>>' --data '{
```

or on servers prior to 13.11.2 (tested on 13.11.1)

1. Login as admin
2. Create a user
3. Deactivate the user
4. Create an api token for the deactivated user
5. Add the user to a project with (use admin token, and a real project id)

Code 141 Bytes

6. Then perform a mutation with the disabled account:

Code 315 Bytes [Wrap lines](#) [Copy](#) [Down](#)

```
1 curl 'https://gitlab.domain.com/api/graphql' -H 'Content-Type: application/json' -H 'Accept: application/json' -H 'Authorization: Bearer <<DEACTIVATED>>'
```

to create a label in the project.

Impact

For GitLab it could lead to loss of revenue due to the ability to create accounts that are not billable but "usable". At the moment the GraphQL API is a bit limited but probably grow in scope.

For users. Running the risk of missing warnings about disabled accounts. Could lead to deletion of account if admins does not notice that the account is being use

Examples

I would guess that this is affecting GitLab.com but can not create a disabled account there.

What is the current *bug* behavior?

With a token from a disabled account the REST API gives:

Code 280 Bytes [Wrap lines](#) [Copy](#) [Down](#)

```
1 curl --header "Authorization: Bearer jKSvxhuDN-Noag6N-w7R" "http://gitlab.joaxcar.com/api/v4/user"
2
3 {"message":"403 Forbidden - Your account has been deactivated by your administrator. Please log back in from a web browser to reactivate your account"}
```

with GraphQL

Code 259 Bytes [Wrap lines](#) [Copy](#) [Down](#)

```
1 curl 'http://gitlab.joaxcar.com/api/graphql' -H 'Accept: application/json' -H 'Content-Type: application/json' -H 'Authorization: Bearer jKSvxhuDN-Noag6N-w7R'
2
3 {"data":{"currentUser":{"id":"gid://gitlab/User/15"}}}
```

What is the expected *correct* behavior?

GraphQL should give a warning as the REST API and block disabled users from accessing data.

Results of GitLab environment info

Code 1.84 KiB [Wrap lines](#) [Copy](#) [Down](#)

```
1 System information
2 System:
3 Current User: gitlab
4 Using RVM: no
5 Ruby Version: 3.0.1p64
6 Gem Version: /usr/lib/ruby/2.7.0/bundler/spec_set.rb:86:in `block in materialize': Could not find rake-13.0.3 in any of the sources (Bundler::GemNotFo
7 from /usr/lib/ruby/2.7.0/bundler/spec_set.rb:80:in `map!'
8 from /usr/lib/ruby/2.7.0/bundler/spec_set.rb:80:in `materialize'
9 from /usr/lib/ruby/2.7.0/bundler/definition.rb:170:in `specs'
10 from /usr/lib/ruby/2.7.0/bundler/definition.rb:237:in `specs_for'
11 from /usr/lib/ruby/2.7.0/bundler/definition.rb:226:in `requested_specs'
12 from /usr/lib/ruby/2.7.0/bundler/runtime.rb:101:in `block in definition_method'
13 from /usr/lib/ruby/2.7.0/bundler/runtime.rb:20:in `setup'
14 from /usr/lib/ruby/2.7.0/bundler.rb:149:in `setup'
15 from /usr/lib/ruby/2.7.0/bundler/setup.rb:20:in `block in <top (required)>'
16 from /usr/lib/ruby/2.7.0/bundler/ui/shell.rb:136:in `with_level'
17 from /usr/lib/ruby/2.7.0/bundler/ui/shell.rb:88:in `silence'
18 from /usr/lib/ruby/2.7.0/bundler/setup.rb:20:in `<top (required)>'
19 from <internal:/usr/lib/ruby/3.0.0/rubygems/core_ext/kernel_require.rb>:85:in `require'
20 from <internal:/usr/lib/ruby/3.0.0/rubygems/core_ext/kernel_require.rb>:85:in `require'
21 Bundler Version: unknown
22 Rake Version: 13.0.3
23 Redis Version: 6.2.3
24 Git Version: 2.31.1
25 Sidekiq Version: 5.2.9
26 Go Version: go1.16.4 linux/amd64
27
28 GitLab information
29 Version: 13.10.4
30 Revision: e11cc45d59e
31 Directory: /usr/share/webapps/gitlab
32 DB Adapter: PostgreSQL
```


```
36 SSH Clone URL: gitlab@gitlab.joaxcar.com:some-group/some-project.git
37 Using LDAP: no
38 Using Omniauth: yes
39 Omniauth Providers:
40
41 GitLab Shell
42 Version: 13.17.0
43 Repository storage paths:
44 - default: /var/lib/gitlab/repositories
45 GitLab Shell path: /usr/share/webapps/gitlab-shell
46 Git: /usr/bin/git
```

Impact

A user with a disabled account can access the GraphQL API without activating the account. Running the risk of missing warnings about disabled accounts. Could lead to deletion of account if admins does not notice that the account is being used. Or accessing data without admins knowing the account is in use.

An admin could create accounts that are not billable but "usable". By creating disabled accounts and use them through GraphQL.

I put it at medium due to the risk of data loss if not getting proper warnings and the fact that it has access to the API even if explicitly told that it should not in the documentation. But feel free to lower the severity if you disagree.


 **h1_analyst_robert** HackerOne triage posted a comment. May 12th (2 years ago)

Hi @joaxcar,

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,
[@croissants](#)

 **h1_analyst_robert** HackerOne triage changed the status to **Needs more info**. May 12th (2 years ago)

Hi @joaxcar

Please provide more information about the impact of the reported behavior. How can an attacker benefit from this? How does this affect end-users or the application infrastructure? The GraphQL query you showed only shows your own ID, nothing sensitive gets leaked.

Thank you in advance for your help!

Regards,
[@croissants](#)

 **joaxcar** changed the status to **New**. May 12th (2 years ago)

Hi @croissants

Thanks for taking the time to review my report! I'm new to bug hunting, so maybe I misunderstand what could count as a vulnerability. I will try to explain my motivation.

There are at least two (might be minor but still) potential scenarios where I can think of this bug affecting users or service providers.

1

First there is the possibility of using the API and thus accessing all information in scope (internal projects, issues, merge requests, code and so on) while still being as an inactive user. There might be scenarios where an admin keeps track of active users by calls to

Code 103 Bytes Wrap lines Copy Download

```
1 curl --header "Authorization: Bearer <ADMIN_TOKEN>" "http://gitlab.domain.com/api/v4/users?active=true"
```

and will not see the inactive user having access to the data. This I would guess is to be considered LOW impact due to the fact that the user could activate its account at any moment and regain the previous privileges. The example when listing my ID was just to show that the call gets through.

2

I don't know exactly how GitLabs priced tiers work. But as I understand it there are no way to create accounts to use as service accounts other than the new "project token" bots. All other accounts are counted as "Billable", see [list billable users](#) and [what users are counted](#). This bug makes it possible for an admin account to create "free" service accounts to be used through GraphQL (deactivated accounts does not count as billable).

But I see the point of this being quite minor. It kind of depends on how GitLab views the ability to create these "free" accounts I guess. If this is not enough for at least LOW severity then feel free to drop the report and I will keep on hunting :)

 **h1_analyst_robert** HackerOne triage closed the report and changed the status to **Duplicate** (#1186729). May 18th (2 years ago)

Hi @joaxcar,

Thank you for your report!

Unfortunately, this was submitted previously by another researcher, but we appreciate your work and look forward to additional reports from you.

For transparency, we have invited you to the original report. Please do not comment on the original submission. If you have any further questions or concerns, please post it on this report instead.

Have a great day ahead!

 maxcar posted a comment.

updated May 18th (2 ye

Hello again @croissants thanks for the reply!


I think that you might have misunderstood the report. The report you linked as a duplicate is mine as well. The other report (#1186729) is about any user listing information about a user with "private user" activated on its account. This report is about a "deactivated user" see https://docs.gitlab.com/ee/user/admin_area/moderate_users.html#deactivating-a-user

As stated in the docs a deactivated user: "Cannot access Git repositories or the API."

Which I have shown they can through GraphQL.

(a deactivated user is a light version of a blocked user, but should still be blocked from API access)

I would like to open up the report again to have a closer look at the difference between "private user" and "deactivated user". As far as I can tell the reports are not related

 maxcar posted a comment.

May 19th (2 ye

There is an additional bug caused by the same underlying bug to the one presented in the initial report. As their mitigation are the same I think that it's better to tr them as one report. I would suggest a change of title to reflect the broader scope but can't change it. Se suggestion for mitigation below.

Summary

If an administrator on a GitLab instance activates a requirement of all users to accept a "terms of service" (see documentation in [link](#)) all current users will get a po stating that the user need to accept the terms to continue using the service. A user who have not accepted the terms or who declines the terms should be loggec from GitLab and denied from usage of the APIs. This blockage works for the REST API but is not enforced on the GraphQL endpoint. If the user have an active acce token the user can use it to access the service without accepting the terms of service.

Impact

A terms of service can be used by organizations to enforce rules and agreements on their users. The terms of service could also be used as a legal contract toward users(some info [link](#) and [link](#)). A user that can access the service despite declining the terms could use this to dispute disciplinary or legal actions taken toward the If the terms are broken. And the bug can render these terms "non enforceable" (see the same link).

Steps to recreate

1. Login as admin
2. Go to https://gitlab.domain.com/admin/application_settings/general
3. Make sure "Terms of Service and Privacy Policy" is disabled
4. Log in as any normal user
5. Go to https://gitlab.domain.com/~profile/personal_access_tokens and create an api token (take a note of the token)
6. Login as admin
7. Go to https://gitlab.domain.com/admin/application_settings/general again
8. Now activate "Terms of Service and Privacy Policy" (if the normal user is still logged in in another browser window, click "decline and log out" in the popup)
9. Make a request to the REST API with the user token from step 5

Code 85 Bytes

[Wrap lines](#) [Copy](#) [Dow](#)

```
1 curl --header "Authorization: Bearer <TOKEN>" "https://gitlab.domain.com/api/v4/user"
```

and be presented with the answer:

Code 174 Bytes

[Wrap lines](#) [Copy](#) [Dow](#)

```
1 {"message": "403 Forbidden - You (@unwilling) must accept the Terms of Service in order to perform this action. Please access GitLab from a web browser"}
```

10. Make a request to the GraphQL API with the same token:

Code 209 Bytes

[Wrap lines](#) [Copy](#) [Dow](#)

```
1 curl 'https://gitlab.domain.com/api/graphql' \
2 -H 'Content-Type: application/json' \
3 -H 'Authorization: Bearer <TOKEN>' \
4 --data '{"query": "query {\\n  currentUser {\\n    id\\n    username\\n    name\\n  }\\n}\\n"}'
```

And be presented with real data and no warning:

Code 144 Bytes

[Wrap lines](#) [Copy](#) [Dow](#)

```
1 {
2   "data": {
3     "currentUser": {
4       "id": "gid://gitlab/User/61",
5       "username": "unwilling",
6       "name": "Unwilling User"
7     }
8   }
9 }
```

11. Make a more fun request listing all projects

Code 202 Bytes

[Wrap lines](#) [Copy](#) [Dow](#)

```
1 curl 'https://gitlab.domain.com/api/graphql' \
2 -H 'Content-Type: application/json' \
3 -H 'Authorization: Bearer <TOKEN>' \
4 --data '{"query": "query {\\n  projects {\\n    nodes {\\n      id\\n      name\\n    }\\n}\\n}\\n"}'
```

What is happening

Code 231 Bytes [Wrap lines](#) [Copy](#) [Download](#)

```

1 rule { default }.policy do
2   enable :log_in
3   enable :access_api
4   enable :access_git
5   enable :receive_notifications
6   enable :use_quick_actions
7   enable :use_slash_commands
8   enable :execute_graphql_mutation
9 end

```

For a user to be able to use GraphQL it seems that only :log_in is needed and :access_api have no effect. So when restricted by either

Code 257 Bytes [Wrap lines](#) [Copy](#) [Download](#)

```

1 rule { deactivated }.policy do
2   prevent :access_git
3   prevent :access_api
4   prevent :receive_notifications
5   prevent :use_slash_commands
6 end
7
8 rule { required_terms_not_accepted }.policy do
9   prevent :access_api
10  prevent :access_git
11 end

```

The user still have log in capabilities (as it should) but the "prevent :access_api" does not restrict the access to GraphQL. Prior to the security patch in [13.11.2](#) to "Require 'api' scope to execute mutations". These users could use both mutations and queries. After the patch the line

Code 61 Bytes [Wrap lines](#) [Copy](#) [Download](#)

```

1 rule { ~can?(:access_api) }.prevent :execute_graphql_mutation

```

effectively blocks mutations for these account while read queries are still possible post 13.11.2.

A side effect of how authentication is handled at the GraphQL endpoint is that a "project access token" cant be used. This is related to the implementation of these access tokens being connected to bot users. And as stated in the same file


Code 93 Bytes [Wrap lines](#) [Copy](#) [Download](#)

```

1 rule { project_bot }.policy do
2   prevent :log_in
3   prevent :receive_notifications
4 end

```


A bot user have no login permission and can thus not access GraphQL despite having :access_api enabled



joaxcar posted a comment.

May 24th (2 years ago)

Hi again @croissants any updates on this issue? I still feel that this is not a duplicate, especially now with the added example. I could write a more detailed comparison between the two reports if needed.



joaxcar requested to disclose this report.

May 26th (2 years ago)

If you are not going to consider this as a valid report I would like to have it disclosed to be able to use the information in a student report at my university.

As the report that is referenced as the "original" of this one was considered "not a bug"(closed as informative) and thus is considered open as of GitLabs policy

Informative or self-closed reports that are determined to be bugs or new feature requests with no current security impact may be imported as public issues in our issue tracker at <https://gitlab.com/gitlab-org/gitlab/issues>.

I would assume that this report could also be disclosed.



joaxcar posted a comment.

Jun 2nd (2 years ago)

Just checking in after another week of silence. No updates on this?



dcouture GitLab staff reopened this report.

Jun 2nd (2 years ago)

Hi @joaxcar,

I reopened the report and will look into it. For the future in situations like this where you add so much details that it's basically describing a new bug I'd suggest simply opening a new report. Messages on closed reports can sometimes get lost through the noise as you have unfortunately noticed here. :)

Thanks for your patience,
Dominic
GitLab Security Team



joaxcar posted a comment.

Jun 2nd (2 years ago)

Hi @dcouture, thank you for looking into the report!

I was hesitant to create a separate report as the underlying bug generating both vulnerabilities probably stem from the same root cause and this one got closed. I since understood that it is usually preferred to split reports containing multiple vulnerabilities anyways and will do that in the future

Do you want me to create a separate report for the "Terms of Service"-bug or can I leave it as is?

dcouture

GitLab staff

posted a comment.

It's OK we'll handle this one here, thanks!

joaxcar

posted a comment.

Updated Jun 3rd (2 years ago)

Updated Jun 4th (2 years ago)

User with expired password

Even after security release 13.12.2 a user with expired password can access the GraphQL API.

As this is related and on your radar now, I will continue to update here. As this also stems from the same underlying bug while broadening the impact.

In the new security release 13.12.2 there is a fix "Insufficient Expired Password Validation" see [commit](#).

This fix is supposed to block users with `expired_passwords` from accessing the API. The fix is subject to the same flaw as for `deactivated user` and `Terms not accepted`. The global policy added for users with expired passwords looks like

Code 123 Bytes

```
1 rule { password_expired }.policy do
2   prevent :access_api
3   prevent :access_git
4   prevent :use_slash_commands
5 end
```

[Wrap lines](#) [Copy](#) [Download](#)

[global_policy.rb](#)

This effectively blocks calls to the REST API but not GraphQL

As you can see, it is the same pattern of prevention of `:access_api` that seem to not affect the GraphQL endpoint. As long as the user have the `:log_in` ability and `api` scope(not the same as `:access_api` ability it seems) a token from the user can be used. I guess all unit tests related to this issue misses this bug as they only test for REST calls.

Impact

The impact is the same as for the other user states. A token from the user put in the state of `expired password` can still be used for (read) queries through the GraphQL endpoint. (mutations are blocked as for the other states as of patch 13.11.2)

Steps to reproduce

(tested on 13.12.2 self-hosted)

1. Create a user `user01`, and log in
2. Go to https://gitlab.domain.com/-/profile/personal_access_tokens and create a `personal access token`
3. Log in as an administrator
4. Go to the admin page for editing the user <https://gitlab.domain.com/admin/users/user01/edit> and change the users password. This triggers `password expired` to be set to the current time. Effectively putting the user in the state of expired password
5. Now try to use the `user01` token in a REST request

Code 84 Bytes

```
1 curl --header "PRIVATE-TOKEN: <TOKEN>" \
2 "https://gitlab.domain.com/api/v4/projects"
```

[Wrap lines](#) [Copy](#) [Download](#)

giving

Code 121 Bytes

```
1 {
2   "message": "403 Forbidden - Your password expired. Please access GitLab from a web browser to update your password."
3 }
```

[Wrap lines](#) [Copy](#) [Download](#)

5. Now try the same with a GraphQL request

Code 253 Bytes

```
1 curl --request POST \
2   --url https://gitlab.domain.com/api/graphql \
3   --header 'Authorization: Bearer <TOKEN>' \
4   --header 'Content-Type: application/json' \
5   --data '{"query":"query {\\n  projects{\\n    nodes{\\n      id\\n      name\\n    }\\n }\\n}"}'
```

[Wrap lines](#) [Copy](#) [Download](#)

and all projects are listed. No warnings or errors

dcouture

GitLab staff

posted a comment.

Hi @joaxcar,

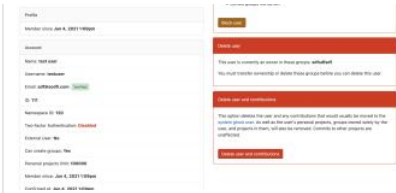
For that last point I'm not sure the bug is that GraphQL is still accessible as much as the REST API should have remained accessible. I'm not quite sure Personal Access Token should really be tied to the user's password expiry and will discuss that with the team.

To be honest there is a lot going on in this report and it's getting very complicated to handle. It's going to take a bit more time, sorry for the delay.

Going back to the first thing you reported, are there any settings to enable to be able to deactivate a user? I don't have the option in my admin dashboard and I'm not sure where to go. You've probably used this feature more than I did so maybe you can help :)

Image F1326225: deactivate.jpg 447.32 KiB

[Zoom in](#) [Zoom out](#) [Copy](#) [Download](#)



1 attachment:
F1326225: deactivate.jpg

dcouture GitLab staff posted a comment. Jun 4th (2 years ago)
Also you're using the projects API but this is one that's actually available unauthorized and isn't leaking anything. Can you still fetch private project data?

joaxcar posted a comment. Jun 4th (2 years ago)
Hi yes sorry about all the extra fluff. I will try to clarify:

Deactivation
A user can only be deactivated by an admin if it has not been active during the last 90 days. This is as I understand because a deactivated user does not add cost as billable seat. But if you as an administrator creates a new user in the admin panel, this user can be deactivated directly (as it has not been used). So to test it out go to the admin panel, create a new user and the option to deactivate will be visible

Image F1326261: deactivate.png 189.98 KiB
[Zoom in](#) [Zoom out](#) [Copy](#) [Download](#)

You then have to generate an access token (impersonation token) as the administrator https://gitlab.domain.com/admin/users/new_user/impersonation_tokens because if you log in you will have to wait 90 days to deactivate. The outcome is the same as for a real deactivated user even though this is a workaround for the PC

Fetching data
I forgot to mention that I am using a server where `public` visibility is turned off. So on my server everything is either internal or private. You can use the API as if you have no restrictions as far as reading goes (as mentioned mutations are restricted). To add information, you can request the field `visibility` besides `id` in the GraphQL request to see that it is there. More than that, in the REST API you are turned away with a warning, seeing none of the public data as far as I could understand if using the token.

So to emphasize, I can see `internal` and `private` (where the user is a member when active) data. This should not be possible.

Password expiration
Yes this is probably a matter of business logic. From what I could get out of the latest patch, the accessibility of the API was considered a medium (6.5) severity vulnerability. And the fix was to close down API access for these users. Including by use of tokens. If it is an issue or not is of course up to you the developers of the application :)

The long and messy information
Sorry again. It grew out of control a bit as I discovered more related issues. The main thing is that the GraphQL API uses a different approach to authorize users. If user have the right to `log in` and have a `valid scope` they are not stopped by `prevent :access_api`. This affects all states of users where login is allowed but the API is restricted. As for now this seems to be the states `deactivated`, `terms not accepted` and `password expired`. The reason for these to have login permission is obvious in context. But all also have reasons why they should not be able to use the API.

1 attachment:
F1326261: deactivate.png

joaxcar posted a comment. Jun 4th (2 years ago)
And as for any delay in handling the report, I am just glad that it got reopened and that you are looking into it! I will try not to add any more unasked for information

And I am more than happy to try to unwind the report, so please don't hesitate to ask if I can help with that!

joaxcar posted a comment. Updated Jun 6th (2 years ago)
I know I promised to not add information. But as you mentioned that it might be that a `personal access token` is not the target of the fix for expired password. I just wanted to add that this same problem also effect OAuth2 tokens (and thereby application connections the user have authenticated towards). I have tested this for case `terms not accepted` and for `password expired`, I am certain that the same issue affects the deactivated user, but I am unsure if a OAuth token can survive the days needed to get the user deactivated.

I tested this by first generating a OAuth token for a user, then using an administrator to either enable ToC or reset user password. The OAuth token is operational in GraphQL but not in REST after this just as the access token.

dcouture GitLab staff changed the status to **Triaged**. Jun 9th (2 years ago)
Hello @joaxcar,

[Original issue #22000](#). This issue will be made public 30 days following the release of a patch.

Given the severity of the report, we are paying an initial \$500 on triage. Congratulations!

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

Best regards,
Dominic
GitLab Security Team

GitLab rewarded [joaxcar](#) with a \$500 bounty. Jun 9th (2 ye

[joaxcar](#) posted a comment. Jun 16th (2 ye
Hi and thank you for the bounty! I am glad that the report got rescued from the "closed" state [@dcouture](#), much appreciated.

Did you get any insight into the matter of API usage for users with "expired password"? Was it the intended behavior, or was the API supposed to be closed?

Thanks again,
Johan

[dcouture](#) GitLab staff posted a comment. Jun 18th (about 1 y
Hi [@joaxcar](#),

I believe it's supposed to be equivalent to the REST API behavior, the team will look into it as the fix is scheduled and we'll have more details.

Best regards,
Dominic
GitLab Security Team

[dcouture](#) GitLab staff closed the report and changed the status to Resolved. Jul 2nd (about 1 y
Hi [@joaxcar](#),

Thank you again for the report! Your finding has been patched in GitLab version 14.0.2 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. If you'd like to publicly disclose this report or details of it in a blog post or elsewhere, please allow 30 days to pass before doing so to give time to our customers to upgrade to a patched version.

This fix addresses the deactivated user scenario and most likely the other ones as well, but in full transparency I haven't had the time to test those yet and didn't w to hold the fix until the next release for that so I'm going to mark this as resolved. I encourage you to open a new report if you find that one of the separate scenari you mentioned still works as it means that the root cause is different and should have been treated as such anyway.

We look forward to your next report!

Best regards,
Dominic
GitLab Security Team

GitLab rewarded [joaxcar](#) with a \$870 bounty. Jul 2nd (about 1 y

[joaxcar](#) posted a comment. Jul 3rd (about 1 y
Hi again [@dcouture](#),
thank you for the bounty, and good work on the quick fix!

I have tested the three scenarios "deactivated user", "not accepted TOC" and "password expired" and I can confirm that they are all blocked from accessing the GraphQL API now. Looked at the patch as well, and it seems that you found the root cause of the problem in the GraphQL controller. The fix looks like it should sto any similar situations in the future.

One thing that feels a bit rushed is that the error message is generic, stating "API not accessible for user" for all cases. Compared to the more verbose messages i REST API

Code 174 Bytes Wrap lines Copy Dow

```
1 {
2   "message": "403 Forbidden - You (@toc) must accept the Terms of Service in order to perform this action. Please access GitLab from a web browser to
3 }
```

Thank you again for handling the report and for the bounty!


Johan

[joaxcar](#) requested to disclose this report. Aug 16th (about 1 y
Would it be possible to disclose this report?

Regards
Johan


[vdesousa](#) GitLab staff posted a comment. Aug 30th (about 1 y
Hello [@joaxcar](#),

There's 2 tokens in the first post. Are they revoked and can we just disclose the report as is?

 [joaxcar](#) posted a comment.
Hi [@vdesousa](#), they are revoked and I cant see anything else that should not be disclosed. So lets disclose it as is


Aug 30th (about 1 y

Thanks
Johan

 [vdesousa](#) GitLab staff agreed to disclose this report.
Thanks for confirmation [@joaxcar](#) ! Disclosing report.

Aug 30th (about 1 y

Best regards,
Vitor
GitLab Security Team

 This report has been disclosed.

Aug 30th (about 1 y