Closed · Issue created 1 year ago by GitLab SecurityBot

A deactivated user can access data through GraphQL

HackerOne report #1192460 by joaxcar on 2021-05-11, assigned to @dcouture:

Report | How To Reproduce

Report

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 it should) link. The GraphQL endpoint does not seam 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 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

```
  rule { deactivated }.policy do
   prevent :access_git
   prevent :access_api
   prevent :receive_notifications
   prevent :use_slash_commands
  end
```

prevents :access_api, and

```
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)

```
curl 'https://gitlab.com/api/graphql' -H 'Accept: application/json' -H 'Content-Type: application/json' -H 'Authorization:
```

User with deactivated account

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

```
curl 'https://gitlab.com/api/graphql' -H 'Accept: application/json' -H 'Content-Type: application/json' -H 'Authorization:
```

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)

```
curl --header "Authorization: Bearer <<ADMIN TOKEN>>" "https://gitlab.domain.com/api/v4/projects//members" --data "user_id=
```

6. Then perform a mutation with the disabled account:

```
curl 'https://gitlab.domain.com/api/graphql' -H 'Content-Type: application/json' -H 'Accept: application/json' -H 'Authori
```

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 billabel but "usable". At the moment the GraphQL API is a bit limited but will 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 used.

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:

```
curl --header "Authorization: Bearer jKSvxhuDN-Noag6N-w7R" "http://gitlab.joaxcar.com/api/v4/user"

{"message":"403 Forbidden - Your account has been deactivated by your administrator. Please log back in from a web browser
```

with GraphQL

```
curl 'http://gitlab.joaxcar.com/api/graphql' -H 'Accept: application/json' -H 'Content-Type: application/json' -H 'Authori

{"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

```
System information
System:
Current User:   gitlab
Using RVM:      no
Ruby Version:   3.0.1p64
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
        from /usr/lib/ruby/2.7.0/bundler/spec_set.rb:80:in `map!'
        from /usr/lib/ruby/2.7.0/bundler/spec_set.rb:80:in `materialize'
        from /usr/lib/ruby/2.7.0/bundler/definition.rb:170:in `specs'
        from /usr/lib/ruby/2.7.0/bundler/definition.rb:237:in `specs_for'
        from /usr/lib/ruby/2.7.0/bundler/definition.rb:226:in `requested_specs'
        from /usr/lib/ruby/2.7.0/bundler/runtime.rb:101:in `block in definition_method'
        from /usr/lib/ruby/2.7.0/bundler/runtime.rb:20:in `setup'
        from /usr/lib/ruby/2.7.0/bundler.rb:149:in `setup'
        from /usr/lib/ruby/2.7.0/bundler/setup.rb:20:in `block in <top (required)>'
        from /usr/lib/ruby/2.7.0/bundler/ui/shell.rb:136:in `with_level'
        from /usr/lib/ruby/2.7.0/bundler/ui/shell.rb:88:in `silence'
        from /usr/lib/ruby/2.7.0/bundler/setup.rb:20:in `<top (required)>'
        from <internal:/usr/lib/ruby/3.0.0/rubygems/core_ext/kernel_require.rb>:85:in `require'
        from <internal:/usr/lib/ruby/3.0.0/rubygems/core_ext/kernel_require.rb>:85:in `require'
Bundler Version:unknown
Rake Version:   13.0.3
Redis Version:  6.2.3
Git Version:    2.31.1
Sidekiq Version:5.2.9
Go Version:     go1.16.4 linux/amd64

GitLab information
Version:        13.10.4
Revision:       e11cc45d59e
Directory:      /usr/share/webapps/gitlab
DB Adapter:     PostgreSQL
DB Version:     13.2
URL:            http://gitlab.joaxcar.com
HTTP Clone URL: http://gitlab.joaxcar.com/some-group/some-project.git
SSH Clone URL:  gitlab@gitlab.joaxcar.com:some-group/some-project.git
Using LDAP:     no
Using Omniauth: yes
Omniauth Providers:

GitLab Shell
Version:        13.17.0
Repository storage paths:
- default:      /var/lib/gitlab/repositories
GitLab Shell path:             /usr/share/webapps/gitlab-shell
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 disables 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.

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

Add API access check to Graphql

gitlab-foss!26570 · 11.10

Activity

GitLab SecurityBot changed due date to September 01, 2021 1 year ago

GitLab SecurityBot added HackerOne security labels 1 year ago

GitLab SecurityBot added Weakness CWE-284 priority 3 severity 3 scoped labels 1 year ago

GitLab SecurityBot @gitlab-securitybot · 1 year ago Author Reporter
HackerOne comment by croissants :

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

GitLab SecurityBot @gitlab-securitybot · 1 year ago Author Reporter
HackerOne comment by croissants :

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

GitLab SecurityBot @gitlab-securitybot · 1 year ago Author Reporter
HackerOne comment by joaxcar :

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 listed as an inactive user. There might be scenarios where an admin keeps track of active users by calls to

```
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 a LOW severity then feel free to drop the report and I will keep on hunting :)

GitLab SecurityBot @gitlab-securitybot · 1 year ago

Author Reporter

HackerOne comment by `croissants`:

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!

Best regards, [@]croissants

GitLab SecurityBot @gitlab-securitybot · 1 year ago

Author Reporter

HackerOne comment by `joaxcar`:

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

---

GitLab SecurityBot @gitlab-securitybot · 1 year ago

Author Reporter

HackerOne comment by `joaxcar`:

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 treat 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 popup 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 logged out 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 access 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 towards the 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 user 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

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

and be presented with the answer:

```
"message": "403 Forbidden - You ([@]unwilling) must accept the Terms of Service in order to perform this action.
```

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

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

And be presented with real data and no warning:

```
{
  "data": {
    "currentUser": {
      "id": "gid://gitlab/User/61",
      "username": "unwilling",
      "name": "Unwilling User"
    }
  }
}
```

11. Make a more fun request listing all projects

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

What is happening

As reported in this issue on GitLab #255354 (comment 424042116) there is a discrepancy in how REST and GraphQL validates users. When sending requests through the GraphQL endpoint the user has to have the permission to log in, see /app/policies/global_policy.rb where the default policy is

```
  rule { default }.policy do
    enable :log_in
    enable :access_api
    enable :access_git
    enable :receive_notifications
    enable :use_quick_actions
    enable :use_slash_commands
    enable :execute_graphql_mutation
  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

```
rule { deactivated }.policy do
  prevent :access_git
  prevent :access_api
  prevent :receive_notifications
  prevent :use_slash_commands
end

rule { required_terms_not_accepted }.policy do
  prevent :access_api
  prevent :access_git
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

```
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

```
rule { project_bot }.policy do
  prevent :log_in
  prevent :receive_notifications
end
```

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

GitLab SecurityBot @gitlab-securitybot · 1 year ago Author Reporter

HackerOne comment by `joaxcar` :

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.

GitLab SecurityBot @gitlab-securitybot · 1 year ago Author Reporter

HackerOne comment by `joaxcar` :

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.

GitLab SecurityBot @gitlab-securitybot · 1 year ago Author Reporter

HackerOne comment by `joaxcar` :

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

GitLab SecurityBot @gitlab-securitybot · 1 year ago Author Reporter

HackerOne comment by `dcouture` :

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

GitLab SecurityBot @gitlab-securitybot · 1 year ago Author Reporter

HackerOne comment by `joaxcar` :

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 have 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?

(there is also the problem that my "signal" is not high enough as my account is new, so at the moment I can't create new reports for GitLab. When my "trail reports" gets generated I will submit some more findings and could also post the ToC one if necessary)

```
/Johan
```

GitLab SecurityBot @gitlab-securitybot · 1 year ago Author Reporter

HackerOne comment by `dcouture` :

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

Dominic Couture @dcouture · 1 year ago Developer

I imported this one a little too quickly, I need to test a few things still

GitLab SecurityBot added security-triage-accsec label 1 year ago

Dominic Couture @dcouture · 1 year ago Developer

@gweaver @jlear Hello folks! It seems like the GraphQL API doesn't do as many checks on the user as the REST API does.

Some examples of account "states" that are able to use the API and probably shouldn't (they can't use the REST API at least):

- A disabled user (with a PAT and possible an OAuth token as well)
- A user with an expired password (with PAT or OAuth token)
- A user that hasn't accepted the Terms of Services (if the admin flips the switch to require it after the user has already obtained a PAT... which is very unlikely of course)

I'm not sure if all those things are the same fix or not. Please let me know if you'd like me to split this in several issues.

Jake Lear @jlear · 1 year ago Contributor

@cablett @digitalmoksha - I could use some help understanding if these is something that we could fix in one go or if we'd want to fix these individually?

Brett Walker @digitalmoksha · 1 year ago Maintainer

I haven't looked at the REST code yet. But it might be as simple as checking if the user has `api_access` privs for all graphql requests.

charlie ablett @cablett · 1 year ago Maintainer

Yea we have https://gitlab.com/gitlab-org/gitlab/-/blob/e5c04dd74419c9d0872849e208cc570419e7ca58/lib/api/api_guard.rb#L103 in the REST API. I wonder if we can easily apply it in GraphQL as well 🤔

charlie ablett @cablett · 1 year ago Maintainer

Yea, we have policy `:access_api` which is set to `prevent` for `blocked | internal` as well as `inactive` . But our GraphQL controller checks for `can?(current_user, :access_api` ...

Edited by charlie ablett 1 year ago

charlie ablett @cablett · 1 year ago Maintainer

What's the difference between a blocked user using our API and an anonymous user doing the same? I suppose we can block the user entirely and they'd have to logout or use no PAT (etc) in order to access a GraphQL query.

Edited by charlie ablett 1 year ago

Jake Lear @jlear · 1 year ago — Contributor

> Yea, we have policy `:access_api` which is set to `prevent` for `blocked | internal` as well as `inactive`. But our GraphQL controller checks for `can?(current_user, :access_api`

@cablett I'm looking at https://gitlab.com/gitlab-org/gitlab/-/blob/master/app/controllers/graphql_controller.rb#L132-134 and https://gitlab.com/gitlab-org/gitlab/-/blob/e5c04dd74419c9d0872849e208cc570419e7ca58/lib/api/api_guard.rb#L103-105 and I'm wondering why the REST API properly rejects deactivated users and the GraphQL doesn't, it seems like the checks are the same? I take the point of your other question with regard to anonymous users, but I'd like to get to the bottom of why these exhibit different behaviors.

charlie ablett @cablett · 1 year ago — Maintainer

> What's the difference between a blocked user using our API and an anonymous user doing the same? I suppose we can block the user entirely and they'd have to logout or use no PAT (etc) in order to access a GraphQL query.

To answer my own question, if the deactivated user could access any private projects, that would be denied and being an anonymous user wouldn't help.

charlie ablett @cablett · 1 year ago — Maintainer

> I'm wondering why the REST API properly rejects deactivated users and the GraphQL doesn't, it seems like the checks are the same?

I did some digging using the DeclarativePolicy debugger and here's what I came up with. Reproducing as above with a deactivated user and an API token and the following bit of code:

```
diff --git a/app/controllers/graphql_controller.rb b/app/controllers/graphql_controller.rb
index 725d8b62c77..952b423007c 100644
--- a/app/controllers/graphql_controller.rb
+++ b/app/controllers/graphql_controller.rb
@@ -130,6 +130,7 @@ def multiplex?
   end

   def authorize_access_api!
+    pp "GraphQL: CAN ACCESS API?? #{can?(current_user, :access_api)}"
     access_denied!("API not accessible for user.") unless can?(current_user, :access_api)
   end

diff --git a/app/models/ability.rb b/app/models/ability.rb
index a185448d5ea..d5583196522 100644
--- a/app/models/ability.rb
+++ b/app/models/ability.rb
@@ -68,6 +68,10 @@ def allowed?(user, ability, subject = :global, opts = {})
         subject = :global
       end

+      pp "user"
+      pp user
+      pp "subject"
+      pp subject
       policy = policy_for(user, subject)

       case opts[:scope]
@@ -76,6 +80,7 @@ def allowed?(user, ability, subject = :global, opts = {})
       when :subject
         DeclarativePolicy.subject_scope { policy.allowed?(ability) }
       else
+        pp policy.debug(:access_api) if ability == :access_api
         policy.allowed?(ability)
       end
     ensure
```

yields:

```
"user"
nil
"subject"
:global
+ [0] enable when default ((<anonymous> : :global))
- [0] prevent when inactive ((<anonymous> : :global))
- [0] prevent when blocked ((<anonymous> : :global))
- [0] prevent when internal ((<anonymous> : :global))
- [0] prevent when deactivated ((<anonymous> : :global))
- [0] prevent when required_terms_not_accepted ((<anonymous> : :global))
- [0] prevent when password_expired ((<anonymous> : :global))
#<DeclarativePolicy::Runner::State:0x000055b701234930
 @enabled=true,
 @prevented=false>
"GraphQL: CAN ACCESS API?? true"
<snip>
```

Looks like the user becomes `anonymous` sometime before this policy check is made, but only in the case of the GraphQL API.

charlie ablett @cablett · 1 year ago — Maintainer

In the GraphQL controller itself, `current_user` returns `nil`. I'm just figuring out why that is. But there's the difference for sure.

charlie ablett @cablett · 1 year ago — Maintainer

I was able to get an `Unauthorized` response by doing the following:

```
diff --git a/app/controllers/application_controller.rb b/app/controllers/application_controller.rb
index 34bad74a9fc..20e0e28d853 100644
--- a/app/controllers/application_controller.rb
+++ b/app/controllers/application_controller.rb
@@ -163,6 +163,8 @@ def auth_user
     strong_memoize(:auth_user) do
       if user_signed_in?
         current_user
+      elsif current_user && current_user.deactivated?
+        current_user
       else
         try(:authenticated_user)
       end
```

It's a bit hamfisted but it illustrates that we're using different methods to get the authorised user depending on the API controller (REST vs GraphQL). Should they be consistent? I feel like -"group::access" is probably the best placed to answer this. @manojmj @mksionek @dblessing @alexpooley @ifarkas WDYT?

Imre Farkas @ifarkas · 1 year ago — Maintainer

@cablett, `user_signed_in?` is a devise helper. I would be worried about the side-effects of setting a user in `#auth_user` despite we are not able to successfully log the user in. 🙁 Can we override it in `GraphqlController` to limit the scope of the change?

Another idea: is the endpoint publicly available? So when the token (and thus the user) fails to authenticate, maybe the request continues as an anonymous user?

Gosia Ksionek @mksionek · 1 year ago — Developer

@jlear @cablett the difference between GraphQl controller and rest api is visible here: https://gitlab.com/gitlab-org/gitlab/-/blob/master/lib/api/api_guard.rb#L54

In the REST api we first check if user exists, then we check if they are allowed to access api :).

I hope it solves part of the mystery!

charlie ablett @cablett · 1 year ago — Maintainer

Thanks everyone! I've asked in the `#f_graphql` but I see that @felipe_artur @smcgivern @reprazent @nick.thomas may also have insight here.

Original issue: gitlab-foss#58547 (closed)

MR: gitlab-foss!26570 (merged)

Bob Van Landuyt @reprazent · 1 year ago — Maintainer

@cablett So the deactivated user can access resources that would be publicly available otherwise, is my understanding correct?

I think @mksionek mentioned the cause: the authentication for Grape finds the user itself, which would include the deactivated user, I think. I think we didn't use the `APIGuard` at the time because it seems very tied to the Grape API. But I do like the idea of finally unifying the approach.

Perhaps an override for the user used in the `GraphqlController` to be released in a security release, keeping the changes contained. Following up with a way to reuse the `APIGuard` in the controller?

Alex Kalderimis @alexkalderimis · 1 year ago — Maintainer

> But I do like the idea of finally unifying the approach.

This seems like the key point to me

Alex Kalderimis @alexkalderimis · 1 year ago — Maintainer

> Perhaps an override for the user used in the `GraphqlController` to be released in a security release,

👍 to that

Alex Kalderimis @alexkalderimis · 1 year ago — Maintainer

Actually it turns out to be much simpler than this - the root cause is that the before actions were in the wrong order.

See: https://gitlab.com/gitlab-org/security/gitlab/-/merge_requests/1517

Please register or sign in to reply

Dominic Couture added GraphQL label 1 year ago

Dominic Couture added group::project management devops::plan scoped labels 1 year ago

GitLab SecurityBot removed security-group-missing security-triage-apisec labels 1 year ago

GitLab SecurityBot @gitlab-securitybot · 1 year ago — Author, Reporter

@qweaver @donaldcook @jlear @cmaxim This issue is ready for triage as per HackerOne process.

About this automation: AppSec Escalation Engine

Donald Cook added backend label 1 year ago

🚘 GitLab Bot 🚘 added section::dev scoped label 1 year ago

Alex Kalderimis mentioned in issue #325945 1 year ago

Costel Maxim @cmaxim · 1 year ago — Developer

CVE request: https://gitlab.com/gitlab-org/cves/-/issues/196

Costel Maxim @cmaxim · 1 year ago — Developer

Fixed with 14.0.2

Costel Maxim closed 1 year ago

GitLab SecurityBot @gitlab-securitybot · 1 year ago — Author, Reporter

This HackerOne security issue was closed 30 days ago and may become public.

Please ensure the following items are true and add a ✅ reaction:

- Issue description and comments do not contain sensitive data belonging to GitLab.
- Issue does not reveal private information of the reporter (i.e. session IDs, passwords).

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.

Dominic Couture made the issue visible to everyone 1 year ago

GitLab SecurityBot @gitlab-securitybot · 1 year ago — Author, Reporter

HackerOne report #1192460 was disclosed on 2021-08-30 @ 13:25.

- Bounty awarded: $1370

Please register or sign in to reply