438 **Project Template functionality can be used to copy private project data, such as repository, confidential issues, snippets, and merg equests**

Share:

TIMELINE

bert submitted a report to GitLab. Sep 6th (3 ye

I've found a three minor vulnerabilities which, when combined, allow an attacker to copy private repositories, confidential issues, private snippets, and then some. through the code path to explain the vulnerabilities and how they are combined. See the **Proof of Concept** section if you want to reproduce it immediately.

Let's start at the `ProjectsController` of EE, which is prepended to `app/controllers/projects_controller.rb` in an EE instance.

ee/app/controllers/ee/projects_controller.rb

Code 252 Bytes Wrap lines Copy Dow

```
override :project_params_attributes
    def project_params_attributes
      super + project_params_ee
    end

def project_params_ee
  attrs = %i[
    # ...
    use_custom_template
    # ...
    group_with_project_templates_id
  ]

  # ...

  attrs
end
```

This method defines what parameters can be passed by the user. The two notable parameters here are `use_custom_template` and `group_with_project_templates_` This method appends the result value of `project_params_attributes` method in `app/controllers/projects_controller.rb` on line 351, which specifies all the CE attributes a user can provide when creating a project. The CE controller allows the `template_name` parameter to be passed, too. This means that these three parameters can be passed to the `Projects::CreateService` in the `create` method:

app/controllers/projects_controller.rb

Code 236 Bytes Wrap lines Copy Dow

```
def create
  @project = ::Projects::CreateService.new(current_user, project_params(attributes: project_params_create_attributes)).execute

  # ...
end

# ...

def project_params_attributes
  [
    # ...
    :template_name,
    # ...
  ]
```

In EE, the `EE:Projects::CreateService` is prepended to the `Projects::CreateService`. The prepended EE code contains logic to validate the `use_custom_template` `group_with_project_templates_id` parameters.

ee/app/services/ee/projects/create_service.rb

Code 765 Bytes Wrap lines Copy Dow

```
def execute
  # ...

  group_with_project_templates_id = params.delete(:group_with_project_templates_id) if params[:template_name].blank?

  # ...

    validate_namespace_used_with_template(project, group_with_project_templates_id)
end

# ...

def validate_namespace_used_with_template(project, group_with_project_templates_id)
  return unless project.group
```

```
  templates_owner = ::Group.find(subgroup_with_templates_id).parent

  unless templates_owner.self_and_descendants.exists?(id: project.namespace_id)
    project.errors.add(:namespace, _("is not a descendant of the Group owning the template"))
  end
end
```

The code above is where the first vulnerability can be found. In a normal situation, a Project Template can only be copied to a namespace (group) that is a descenda the project template. However, the `validate_namespace_used_with_template` method returns a `nil` value when the project is **not** being created for a group (`retu unless project.group`). This means that if a `group_with_project_templates_id` is given for a project that is created in a `User` namespace, the authorization / valid logic is never executed. This means that the `use_custom_template` and `group_with_project_templates_id` parameters remain to be set on the instance variable `params` .

Because the EE code is prepended, the `execute` method is executed before the `Projects::CreateService` is called. Because the EE class its validation logic is bypassed, the `execute` method of the `Projects::CreateService` class is called:

app/services/projects/create_service.rb

Code 151 Bytes Wrap lines Copy Dow

```
def execute
  if @params[:template_name].present?
    return ::Projects::CreateFromTemplateService.new(current_user, params).execute
  end

  # ...
end
```

When a `template_name` is given, instead of executing the normal execution flow, the result of `Projects::CreateFromTemplateService` is returned. The CE code for class isn't very important. The EE class contains the logic that is worth checking out:

ee/app/services/ee/projects/create_from_template_service.rb

Code 704 Bytes Wrap lines Copy Dow

```
def execute
  return super unless use_custom_template?

  override_params = params.dup
  params[:custom_template] = template_project if template_project

  ::Projects::GitlabProjectsImportService.new(current_user, params, override_params).execute
end

private

def use_custom_template?
  # ...
    template_name &&
      ::Gitlab::Utils.to_boolean(params.delete(:use_custom_template)) &&
      ::Gitlab::CurrentSettings.custom_project_templates_enabled?
  # ...
end

def template_project
  # ...
    current_user.available_custom_project_templates(search: template_name, subgroup_id: subgroup_id)
                .first
  # ...
end

def subgroup_id
  params[:group_with_project_templates_id].presence
end
```

This class does a couple of things: it makes sure a custom template name is given, that it should use the given template name, and that the GitLab instance has cu project templates enabled. For what it's worth: gitlab.com has this setting enabled. When it passes those checks, the `template_project` method is invoked. Here definition of the `available_custom_project_templates` method:

ee/app/models/ee/user.rb

Code 357 Bytes Wrap lines Copy Dow

```
def available_custom_project_templates(search: nil, subgroup_id: nil)
  templates = ::Gitlab::CurrentSettings.available_custom_project_templates(subgroup_id)

  ::ProjectsFinder.new(current_user: self,
                       project_ids_relation: templates,
                       params: { search: search, sort: 'name_asc' })
                .execute
end
```

ee/app/models/ee/application_setting.rb

Code 208 Bytes Wrap lines Copy Dow

```
def available_custom_project_templates(subgroup_id = nil)
  group_id = subgroup_id || custom_project_templates_group_id

  return ::Project.none unless group_id

  ::Project.where(namespace_id: group_id)
end
```

This method will return all `Project` models based on the `namespace_id` that is provided in the `subgroup_id` parameter. This is then passed to the `ProjectsFinde` the `available_custom_project_templates` method on the `User` model. This is where the second vulnerability can be found. The `ProjectsFinder` uses an initial collection, which consists of the projects the authenticated user can access. However, it does **not** check the access level of the user. This means that any project is public, but has Repository, Issue, Snippets (etc.) access disabled for Guests, will be returned by the `available_custom_project_templates` method on the `User` model. In a perfect world, it seems that this method would limit the projects that can be returned based on the user's permissions for said projects.

If we go back to the `EE:Projects::CreateFromTemplateService` file, you can see that the `template_project` will return the first project that is returned by the `available_custom_project_templates` method. This means that `params[:custom_template]` may contain a `Project` model that the user is not authorized to see everything for. The `EE::Projects::CreateFromTemplateService` class then calls the `Projects::GitlabProjectsImportService` class with the updated parameters.

ee/app/services/ee/projects/gitlab_projects_import_service.rb

Code 675 Bytes Wrap lines Copy Dow

```
def execute
  super.tap do |project|
    if project.saved? && custom_template
      custom_template.add_export_job(current_user: current_user,
                                     after_export_strategy: export_strategy(project))
    end
  end
end

private

override :prepare_import_params
def prepare_import_params
  super

  if custom_template
    params[:import_type] = 'gitlab_custom_project_template'
  end
end

def custom_template
  strong_memoize(:custom_template) do
    params.delete(:custom_template)
  end
end

def export_strategy(project)
 Gitlab::ImportExport::AfterExportStrategies::CustomTemplateExportImportStrategy.new(export_into_project_id: project.id)
end
```

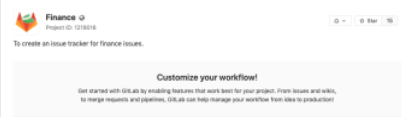
This EE class is prepended, but uses `super.tap` to call the CE code (`super`) and then taps into the result of the CE code. If `params[:custom_template]` has been s and the project was successfully saved by the `super` call, an export job is scheduled for the `custom_template` that was returned by the `ProjectsFinder` . It's wort nothing that at this point the user may not be authorized to see the code, issues, etc., of the project. Additionally, an export strategy is passed that imports the ex file in the newly created project.

This is where the third vulnerability can be found. When an export job is scheduled, it assumes the user is authorized to make the export. Ideally, the Sidekiq job (`ProjectExportWorker`) that is scheduled would do an authorization check to make sure that the user is authorized to export the project. This would also avoid a TOCTOU issue where the user schedules a job when the queue is clogged / Sidekiq workers are paused and would leave the project before the job is executed. Wh the export is created, it'll automatically be imported in the project that the user has full access to.

Combined, these vulnerabilities results in an attacker being able to obtain any confidential information that is included in a project export. This vulnerability **only** w for public projects with limited access levels for repositories, issues, pipelines, merge requests (and more) that belong to a group. A good example of this would be `gitlab-org` , `gitlab-data` , `gitlab-com` , on gitlab.com. There are plenty of repositories, such as https://gitlab.com/gitlab-com/finance (see below), that are pub but don't expose the repository, issues, and merge requests.

Image F576178: Screen_Shot_2019-09-05_at_10.08.18_PM.png 41.82 KiB

Zoom in Zoom out Copy Download



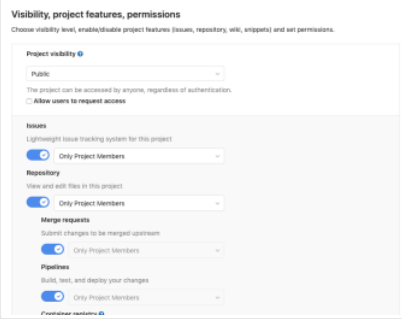
Proof of Concept

To reproduce this vulnerability:

Members:

Image F576180: Screen_Shot_2019-09-05_at_10.13.21_PM.png 75.53 KiB

Zoom in Zoom out Copy Download



- sign into another account and go to http://instance/projects/new
- create a new project and intercept the request, it'll look something like this (I've left out unimportant parameters):

Code 602 Bytes Wrap lines Copy Dow

```
POST /projects HTTP/1.1
Host: instance
...

----------506740453
Content-Disposition: form-data; name="project[use_custom_template]"

false
----------506740453
Content-Disposition: form-data; name="project[template_name]"

----------506740453
Content-Disposition: form-data; name="project[group_with_project_templates_id]"

----------506740453
Content-Disposition: form-data; name="project[name]"

project_name
----------506740453
Content-Disposition: form-data; name="project[namespace_id]"

1
----------506740453
Content-Disposition: form-data; name="project[path]"

project_name
----------506740453--
```

- in this request, change `use_custom_template` to `true`, the `template_name` to the name the victim gave to the project (`test_project`), and `group_with_project_templates_id` to the group ID of the public group the victim created (`1`). When forwarded, the server will respond with a redirect and, wh followed, show a page indicating that the project is being imported:

Image F576184: Screen_Shot_2019-09-05_at_10.17.09_PM.png 13.88 KiB

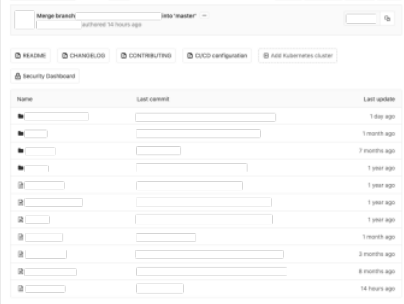
Zoom in Zoom out Copy Download

Depending on the size of the project and how busy the queues are, it can take a couple of minutes to generate the export of the project and then import it to the n project. Come back in a couple minutes and find the repository, confidential issues, private snippets, merge requests, CI pipelines, and more being copied to the attacker's project.

Redacted copy of `gitlab-com/finance`

Image F576189: Screen_Shot_2019-09-05_at_10.19.15_PM.png 79.69 KiB

Zoom in Zoom out Copy Download



Impact

Any access level that has been put in place for projects the user can access can be bypassed using this vulnerability. According to the documentation, this means t
the following information can be obtained:

- Project and wiki repositories
- Project uploads
- Project configuration, including services
- Issues with comments, merge requests with diffs and comments, labels, milestones, snippets, and other project entities
- LFS objects
- Issue Boards

Image F576190: cat.gif 726.45 KiB

Zoom in Zoom out Copy Download

5 attachments:
F576178: Screen_Shot_2019-09-05_at_10.08.18_PM.png
F576180: Screen_Shot_2019-09-05_at_10.13.21_PM.png
F576184: Screen_Shot_2019-09-05_at_10.17.09_PM.png
F576189: Screen_Shot_2019-09-05_at_10.19.15_PM.png
F576190: cat.gif

OT: gitlab-securitybot posted a comment. Sep 6th (3 ye

Hi @jobert,

Thank you for submitting this report. We will investigate the issue as soon as possible.
Due to our current workload, we will get back within 10 business days with an update.

Please refrain from submitting your report or inquiring about its status through
additional channels, as this unnecessarily binds resources in the security team.

Best regards,
GitLab Security Team

jmatos_bgtvf changed the status to **Triaged**. Sep 6th (3 ye

Hello @jobert,

Thank you for this amazing report.

We have verified this finding and our engineering team is working on a fix ASAP. We will be tracking progress internally at https://gitlab.com/gitlab-org/gitlab-ce/issues/67109.

Best regards,
GitLab Security Team

jritchey GitLab staff posted a comment. Sep 6th (3 ye

Hi @jobert ,

We've deployed a hotpatch on production, can you verify if this mitigates the immediate problem?

Thanks again!
James

OT: gitlab-securitybot posted a comment. Sep 7th (3 ye

The GitLab issue created from your report is currently scheduled for 2019-09-22.

Thank you again for contacting us!

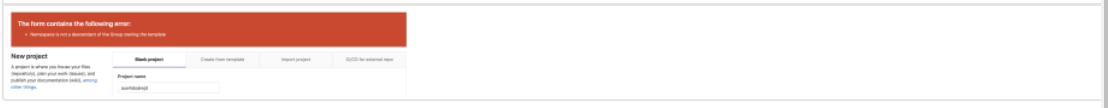
Best regards,
GitLab Security Team

jobert posted a comment. Sep 7th (3 ye

Hi @jritchey and @jmatos_bgtvf - I'm not able to reproduce the security vulnerability anymore. Great turnaround time! When I tried to reproduce it, I'm seeing an when creating the project.

Image F577306: Screen_Shot_2019-09-07_at_11.57.32_AM.png 124.38 KiB

Zoom in Zoom out Copy Download



I've also tried working around the patch by adding myself as a Guest of a subgroup and a Developer of the parent and then trying to export a confidential issue of a Project in the subgroup (not available by default for Guest users). However, because permissions trickle down to its descendants, I'm automatically granted Devel rights on the subgroup as well.

If you have code available somewhere, I'm happy to check that out, too.

1 attachment:
F577306: Screen_Shot_2019-09-07_at_11.57.32_AM.png

dappelt GitLab staff posted a comment. Sep 9th (3 ye

Hi @jobert - thanks for validating. Attached is the code change to fix the vuln. Do you see any problem?

Thanks,
Dennis

1 attachment:
F578278: Screenshot_2019-09-09_at_11.00.42.png

jmatos_bgtvf posted a comment. Sep 9th (3 ye

Hello @jobert,

This is the fix for the 1st vulnerability of your exploit chain. We have created 2 other issues with lower priority for the 2 other vulnerabilities you used (https://gitlab.com/gitlab-org/gitlab-ee/issues/14861 and https://gitlab.com/gitlab-org/gitlab-ee/issues/14860).

Do not hesitate to let us know if you believe fixing first only this vulnerability is a bad idea.

Thank you again,
Jeremy

jobert posted a comment. Sep 9th (3 ye

Hi @dappelt and @jmatos_bgtvf - thanks for sending over the code. It didn't help me identify a bypass for the original vulnerabilities that I reported, but it did poin a potential new one. A new `Project` model gets initialized with `params`, which may lead to a mass-assignment vulnerability. E.g. the `namespace_id` validation ha run yet when that code gets executed, which means that if someone would provide a `namespace_id` that doesn't belong to them, the view will render with a `@pro` instance variable that, if it'd call `namespace`, would return a namespace the user may not be authorized to see. I personally wouldn't rely on that logic, so you might want to stay away from initializing the model with the raw parameters *or* run the additional authorization checks before initializing the model. Hope that helps!

GitLab rewarded jobert with a **$12,000** bounty. Sep 11th (3 ye

estrike GitLab staff closed the report and changed the status to **0 Resolved**. Sep 11th (3 ye

Hi @jobert,

Thank you again for the excellent report! Your finding has been mitigated in GitLab versions 12.2.5, 12.1.9, and 12.0.9, and we are awarding a $12,000 bounty. Congratulations!

The mitigation put is a revised version of the fix put in place on Friday. As Jeremy noted, we still have two more outstanding issues related to the report. We are planning to make all of the GitLab issues public once they are all fixed.

For this report, we are going to mark as Resolved since the primary impact is mitigated. We do request that we hold off on making it public in 30 days since it includ some outstanding vulns. What do you think?

Please let us know if you find that the revised patch does not mitigate your finding.

We look forward to your next report!

Best regards,
Ethan
Security Team | GitLab Inc.

jobert posted a comment. Sep 11th (3 ye

jobert requested to disclose this report. Oct 11th (3 ye

Hi @estrike - hope all is well! It's been 30 days since the release of the security advisory and the new version. Requesting disclosure as per your last comment. Tha

estrike GitLab staff posted a comment. Oct 11th (3 ye

Hi @jobert ,

I just checked the remaining two issues, and unfortunately they are still open. We will review those with the development team, to get those fixes scheduled. I am going to disclose or cancel the disclosure request at the moment, in hopes that we can get them addressed within the 30 day window. We will keep you updated h we progress towards getting them fixed. Thank you for your patience as we work towards mitigating each of the findings in this report.

Best Regards,
Ethan

jobert posted a comment. Nov 26th (3 ye

Hi @estrike - is there an update you can provide on the remaining tickets? Let me know if there's anything I can help with here. Thanks!

jmatos_bgtvf posted a comment. Nov 27th (3 ye