Redmine - Defect #35789

Redmine is leaking usernames on activities index view

2021-08-20 19:17 - Mischa The Evil

Status:	Closed	**Start date:**	
Priority:	High	**Due date:**	
Assignee:	Marius BĂLTEANU	**% Done:**	0%
Category:	Security	**Estimated time:**	0.00 hour
Target version:	4.1.5		
Resolution:	Fixed	**Affected version:**	

Description

Redmine currently leaks usernames when the activities index view is requested with a user_id param that has a non-visible user_id argument.
The cause of this is that the @author instance variable in the activities controller is populated with the user having the user_id argument without doing a visible check (see source:/trunk/app/controllers/activities_controller.rb@21197#L36).

This issue has been present since Redmine 0.8 (where the user activities list feature was introduced, feature #1002) and exists up until now (trunk @ r21197). Though from 0.8 up to and including 2.6.x there wasn't an explicit setting to control user visibility. With 3.0.0 we got the user visibility feature from #11724, but this case wasn't modified to obey that particular setting.

I'll leave two (cumulative) patches with test coverage:

- The first one is pretty simple. It just adds the visibility check and as a result, when the page is requested with an non-visible user, renders a 404 instantaneously. This relies on the fact that the controller already rescues ActiveRecord::RecordNotFound exceptions via source:/trunk/app/controllers/activities_controller.rb@21197#L83.
- The second one changes the above given behavior a bit to work in a slightly more sophisticated manner. It wraps the @author population in a block that rescues the ActiveRecord::RecordNotFound exception itself and populates @author with either the visible (and active) user or nil according to the result of the call to User.visible.active.find(params[:user_id]). This way Redmine doesn't throw a 404 error immediately. Instead, it will respond with a sanitized activities index view when it is requested with a user_id param with a user_id argument that is not visible.

FWIW: I have no particular preference for how this leakage gets resolved. I'd be ok with both the solutions I propose.

Please let me now if more information is needed.

Associated revisions

Revision 21209 - 2021-09-06 20:40 - Marius BĂLTEANU

Return 404 when filtering by a non-visible user in activity view (#35789).

Patch by Mischa The Evil.

Revision 21215 - 2021-09-06 23:02 - Marius BĂLTEANU

Merge r21209 from trunk to 4.2-stable (#35789).

Revision 21216 - 2021-09-06 23:03 - Marius BĂLTEANU

Merged r21209 to 4.1-stable (#35789).

History

#1 - 2021-08-22 04:11 - Go MAEDA

- File 35789.png added

- Status changed from New to Confirmed

- Target version set to 4.1.5

35789.png

#2 - 2021-08-24 20:30 - Holger Just

Michael Vorobiev: Thank you for finding this and providing the patches!

I think it is clearer to explicitly deny access rather than silently ignoring an explicit parameter. As such, I prefer the first patch. I'm aware that Redmine in fact sometimes ignores invalid parameters rather than rejecting them, but I believe this should be restricted.

In case we still want to use the second patch, I think the code could be simplified to this with the exact same semantics:

```
@author = User.visible.active.find_by_id(params[:user_id].to_i)
```

#3 - 2021-08-25 02:46 - Mischa The Evil

Holger Just wrote:

> Michael Vorobiev: Thank you for finding this and providing the patches!

No problem. Off-topic: Fun fact is that I actually stumbled upon this (blatantly obvious, once identified) issue while reviewing the implementation of #33602 for potential leaks exactly like this one...

Holger Just wrote:

> I think it is clearer to explicitly deny access rather than silently ignoring an explicit parameter. As such, I prefer the first patch. I'm aware that Redmine in fact sometimes ignores invalid parameters rather than rejecting them, but I believe this should be restricted.

I tend to lean towards that opinion too.

> In case we still want to use the second patch, I think the code could be simplified to this with the exact same semantics [...]

FWIW: I just tried to provide code that was as explicit as possible. Nevertheless: that line looks nice and tidy. Gotta like Ruby for such...

#4 - 2021-09-04 11:10 - Marius BĂLTEANU

- Assignee set to Marius BĂLTEANU

#5 - 2021-09-04 11:28 - Marius BĂLTEANU

Thanks Mischa for catching this!

Holger Just wrote:

> I think it is clearer to explicitly deny access rather than silently ignoring an explicit parameter. As such, I prefer the first patch. I'm aware that Redmine in fact sometimes ignores invalid parameters rather than rejecting them, but I believe this should be restricted.

I'm in favour of keeping the existing behaviour from filters where the filter value is ignored. Lets take a case where a user don't have access to an issue:

- opening the issue url (/issues/:issue_id) returns 403 which is correct.
- filtering the issues list after issue id (issues?v[issue_id][]=issue_id) returns "No data to display" which is more friendly (from my perspective).

Also, I played with some filters from Github/Gitlab and they have the same behaviour, showing messages like "No results found" if you enter invalid params.

> In case we still want to use the second patch, I think the code could be simplified to this with the exact same semantics:
>
> [...]

Good ideea, thanks!

#6 - 2021-09-06 18:55 - Holger Just

Marius BALTEANU wrote:

> I'm in favour of keeping the existing behaviour from filters where the filter value is ignored. Lets take a case where a user don't have access to an issue:
>
> - opening the issue url (/issues/:issue_id) returns 403 which is correct.
> - filtering the issues list after issue id (issues?v[issue_id][]=issue_id) returns "No data to display" which is more friendly (from my perspective).

This is not entirely the same situation. In the issue filter example you describe, Redmine "pretends" the issue does not exist but still applies the filter.

In the case at hand however, if we ignore the filter (i.e. option 2), the behavior would be the same as if no filter would have been supplied in the request at all, thus showing the normal activity view of all users, not just for the given user id. In my eyes, this would be rather surprising and

confusing.

Following your logic, the behavior should be correct if we either:

- show a 404, i.e. show that there is no data for the given filter, or
- to return an empty activity list with a 200, i.e., pretend the user has no activities (whether that is true or not)

The option to return an empty results set however makes it impossible to distinguish between an invalid filter and a valid one which just returns an empty set. We would also have to make sure that we are still not leaking any data of the hidden author.

As such, I'm still in favor of option 1 as it makes it explicit that the whole filter is invalid. This allows to distinguish it from a valid query which returns either an empty results set (if we return an empty activity list for an invalid user) or the result set for an entirely different request (with option 2). In my eyes, option 1 is the easiest option to implement with the fewest corner cases.

#7 - 2021-09-06 20:45 - Marius BĂLTEANU

Thanks Holger for your clarifications, I've committed the first patch.

Should we release new versions?

#8 - 2021-09-06 23:04 - Marius BĂLTEANU

- Status changed from Confirmed to Resolved

- Resolution set to Fixed

#9 - 2021-09-07 14:10 - Holger Just

Marius BALTEANU wrote:

> Should we release new versions?

Sure. You are planning to release 4.1.5 and 4.2.3, right? I would then request a CVE entry for this.

In the security scanner, I would mark this issue as moderate severity, given that this circumvents the user visibility restrictions in many cases. Do you agree?

#10 - 2021-10-02 20:38 - Marius BĂLTEANU

Holger Just wrote:

> Marius BALTEANU wrote:
>
> > Should we release new versions?
>
> Sure. You are planning to release 4.1.5 and 4.2.3, right? I would then request a CVE entry for this.
>
> In the security scanner, I would mark this issue as moderate severity, given that this circumvents the user visibility restrictions in many cases. Do you agree?

Sorry for my late reply. Yes, I'm thinking to release 4.1.5 and 4.2.3 next weekend (October 10th). Regarding the severity, yes, it sounds good to me.

#11 - 2021-10-10 11:59 - Marius BĂLTEANU

- Status changed from Resolved to Closed

#12 - 2021-10-10 12:08 - Marius BĂLTEANU

I've updated the Security_Advisories. Holger, feel free to update based on the security scanner.

#13 - 2021-10-10 15:54 - Holger Just

Thanks for the release. I have updated the security scanner with the new releases.

On Monday, I'll request a CVE entry for this. Once it's assigned, I'll update the Security_Advisories wiki page and the Security scanner accordingly.

#14 - 2021-10-12 21:19 - Holger Just

CVE-2021-42326 was assigned for this issue.

#15 - 2022-06-21 08:10 - Marius BĂLTEANU

- Private changed from Yes to No

Files

0001-Fix-username-leakage-on-activities-index-with-user_i.patch	1.96 KB	2021-08-20	Mischa The Evil
0002-Don-t-throw-a-404-error-on-activities-index-with-a-n.patch	2.54 KB	2021-08-20	Mischa The Evil
35789.png	243 KB	2021-08-22	Go MAEDA