ChiliProject - Bug # 345: Entering large numbers for 'Estimated Time' fails with 'Invalid big Decimal Value'

Status:	Closed	**Priority:**	Normal
Author:	Gregor Schmidt	**Category:**	Issue tracking
Created:	2011-04-20	**Assignee:**	Eric Davis
Updated:	2011-05-27	**Due date:**	

Remote issue URL:

Affected version: master

Description:

While trying to set estimated time of an issue to 5000000 hours, I am getting a Server Error. The back trace looks like the following:

```
<pre>
RuntimeError (Invalid big Decimal Value):
  config/initializers/bigdecimal-segfault-fix.rb:31:in `BigDecimal'
  /Users/schmidt/.rvm/rubies/ree-1.8.7-2011.03/lib/ruby/1.8/bigdecimal/util.rb:20:in `to_d'
  app/models/issue.rb:741:in `recalculate_attributes_for_without_remaining_hours'
  app/models/issue.rb:720:in `update_parent_attributes'
  app/models/issue.rb:556:in `save_issue_with_child_records'
  app/models/issue.rb:538:in `save_issue_with_child_records'
  app/controllers/issues_controller.rb:170:in `update'
</pre>
```

I am using Ruby 1.8.7-p334, ChiliProject master, mysql2 with MySQL 5.1. It also fails in ree-1.8.7-2011.03.

Analysis:

Back in Summer 09, Eric Davis added a workaround for a segfault bug in Ruby to the ChiliProject sources (source:config/initializers/bigdecimal-segfault-fix.rb)

That bug was present in

* 1.8.6-p368 and all prior versions
* 1.8.7-p160 and all prior versions

and is fixed in all ruby version released after June 2009. More information may be found at https://github.com/NZKoz/bigdecimal-segfault-fix.

Anyway, this work around has some undesired side effects, i.e. you cannot enter really large numbers in some form fields.

I am proposing to remove that workaround, since we should be able to assume, that nowadays, nearly 2 years later, everybody was able to update their ruby interpreter. I think, there have been multiple other bugs and security issues, that where reported since then and ChiliProject does not provide work-arounds for them either. Ergo, this work around does not increase ChiliProject's security in a significant way.

If removing the workaround seems to be unfeasible, we could at least guard the patch with checks, so that only those versions are patched, that are affected by the bug.

Associated revisions

2011-05-27 11:37 pm - Eric Davis

History

2011-04-20 04:34 pm - Gregor Schmidt
- Status changed from Open to Ready for review

The outlined changes may be found at

https://github.com/schmidt/chiliproject/tree/b/345-remove-big-decimal-patch

and

https://github.com/schmidt/chiliproject/tree/b/345-limit-scope-of-big-decimal-patch

2011-04-20 06:51 pm - Eric Davis
Since most people won't be logging 5,000,000 hours in one time entry (570 years of work), I think we should just remove the patch from 2.0.0 since we would be dropping/phasing out 1.8.6 then.

2011-04-20 08:03 pm - Gregor Schmidt
Thanks for having a look at this issue.

Although I agree with your opinion, I do not follow your conclusion:

The security issue, that is fixed by this work around, is present in older versions of 1.8.6 and 1.8.7. Furthermore it is fixed in current versions of 1.8.6 and 1.8.7. Therefore, phasing out 1.8.6 is not helping in this context. This would also mean, that removing the patch does not need a major release but could be done in 1.3.0.

2011-04-20 11:59 pm - Eric Davis
Some clarification as to why I said to just remove it in 2.0.0

* From what I remember, this bug was worked around in a newer rails version which is in unstable already, so 2.0 could have it removed. (I'll need to check the security report).
* Removing the patch from 1.x could re-expose the security hole for users on older versions of Ruby. (You can't assume everyone is on the latest Ruby. I had a client on 1.8.5 until only a few months ago.)
* If the scoping patch works for all users on older Ruby versions, then we might be able to add it for 1.x *but* this feels like such an edge case, I'm not sure there is enough time to include it (i.e. more important bugs are still pending). If someone has the time to review it with older versions of Ruby before mid-next week, we might be able to include it in 1.3.0. Otherwise it will need to wait until 1.4.0, which might not be released if 2.0 is ready before then :)

2011-04-21 05:20 am - Gregor Schmidt
- Target version changed from 1.3.0 to 2.0.0

Thanks for your detailed explanations.

I don't think it is worth it to first add the guards, just to completely remove the patch in the next release.

Shall I open a pull request targeting the unstable branch which just removes the file?

2011-04-21 05:24 am - Gregor Schmidt

Eric Davis wrote:

> * From what I remember, this bug was worked around in a newer rails version which is in unstable already, so 2.0 could have it removed. (I'll need to check the security report).

According to "this post on the Rails Security mailing list":http://groups.google.com/group/rubyonrails-security/browse_thread/thread/dd820c64429b8bca?pli=1 these changes where introduced in Rails 2.3.3, i.e. they are already in master.

2011-05-27 09:40 pm - Eric Davis

- Assignee set to Eric Davis

- (deleted custom field) set to master

- Status changed from Ready for review to Closed

I've removed the patch. Looking at the history, it was added while on Rails 2.2.2. Since we are on 2.3.11 now Rails should handle it for us.

Thanks for reviewing and researching this. We could probably do a sweep through the code and remove old patches and compatibility hacks now (e.g. the cruft at the bottom of config/routes.rb...)