Ruby master - Misc #15202

Adding Coverity Scan to CI to see the result casually

10/04/2018 02:06 PM - jaruga (Jun Aruga)

Status:	Open	
Priority:	Normal	
Assignee:		

Description

Recently I reported issues detected by code analysis tool mainly using Coverity Scan.

The 9 issues categorized as "important" was fixed by #15116. (Thank you!)

> https://bugs.ruby-lang.org/issues/15116
>
> However as a "not important" issues, around 1000 issues were detected by the tool for the ruby 2.5.1.
> I am considering how to deal with this or report those.
> I might open an another ticket for that.

However there are around 1000 "not important" issues.

Right now I do not share the report file (840KByte) for that, because it makes people tired.
If someone want to see it, I am happy to share it here as an attachment.

Instead of that, It looks good to me that someone could see the result of coverity scan casually anytime to fix those in long term.

What I want to propose is to add coverity scan test on rubyci or Travis CI.

I do not know how coverity scan is used on current Ruby project as a regular workflow.
But I could see it is actually used from the setting [2] and some tickets. [3]

I found how to use Coverity Scan on Travis CI [4], and the used cases [5][6].

How do you think?

- [1] rubyci: https://www.rubyci.org/
- [2] coverity scan ruby project: https://scan.coverity.com/projects/ruby
- [3] coverity scan used tickets:
  - https://bugs.ruby-lang.org/projects/ruby-trunk/repository/revisions/61862
  - https://bugs.ruby-lang.org/projects/ruby-trunk/repository/revisions/55763
  - https://bugs.ruby-lang.org/projects/ruby-trunk/repository/revisions/50734
- [4] How to use Coverity Scan on Travis CI: https://scan.coverity.com/travis_ci
- [5] The cases for coverity scan on Travis CI:
  - https://github.com/nanoporetech/scrappie/blob/master/.travis.yml
  - https://github.com/JanusGraph/janusgraph/blob/master/.travis.yml

History

#1 - 10/04/2018 02:13 PM - jaruga (Jun Aruga)

- Description updated

#2 - 10/04/2018 02:17 PM - jaruga (Jun Aruga)

- Description updated

#3 - 10/04/2018 03:06 PM - mame (Yusuke Endoh)

I had run the Coverity Scan analysis on CI (twice a week), and I had checked the result only when I felt like. But recently I forgot it completely. By this ticket, I have just noticed that the analysis has not worked since Feb. 2018 :-)

Personally, I no longer like to spend effort for Coverity Scan. The analysis result includes too many false warnings. I have no intention to blame Coverity Scan; many of the false alarms are really subtle and even hard for human to understand that they are actually false. Anyway, it really makes me tired. Indeed, it sometimes tells us actual bugs, but I don't see that the advantage is worth the cost.

If anyone wants to use Coverity Scan again, I'd like to grant her/him to restore the build analysis. It would be preferable that s/he is a Ruby committer

according to Coverity Scan FAQ "Who may be granted access to a Registered Project?".

#4 - 10/04/2018 09:34 PM - shevegen (Robert A. Heiler)

> However there are around 1000 "not important" issues.

I think the ruby core team always likes to remove bugs from ruby, if these reports constitute real bugs that is. At the github page of mruby one can see that some sort of systematically try to find ways to break ruby, if you look at some of the issues there. :P

Many such reported examples are really quite contrived. I think that makes a huge difference towards bugs that are caused by a "regular" use of ruby, by a real person - that is, you write ruby code in some application and discover that things are failing or breaking, as opposed to automatic/systematic testing of syntax that can crash ruby, like fuzzing in C/C++ and such.

I remember being able to do so (that is, causing segfaults) with the ruby-gtk bindings - not on purpose, but simply because some of the code I used is quite long, many thousand lines of code, and very old too (much of which was written by me too, many years ago), and it is hard to make changes (I got a bit lazy from gtk2 to gtk3 so I don't write anywhere near as much gtk-code in ruby these days); and a bit cumbersome to find and report bugs too. (Even if Kouhei Sutou does a great job improving, fixing bugs and maintaining the ruby-gnome bindings.) But I think developer manpower is still limited, in numbers alone. Perhaps not only in numbers but in knowledge too. I am sure there are many people who know ruby quite well, but significantly fewer who know both ruby and C very well.

I also don't think there is any ... realistic way that the ruby bug tracker could handle reports like +1000 bugs started. :)

The total amount of bugs reported on the issue tracker so far here is 8307. Even though nobu is the patch monster from Mount Fuji with 20 arms, even he may have a hard time tackling 1000 different bugs (even if they are well documented and reproducible),

What may be useful, perhaps, is if there could be some way to not only find "real" bugs, but those that seem to be more "promising" and worthwhile to fix. Like to somehow semi-automatically "promote" the more outstanding bugs from Coverity Scan or any other method that may have a real, larger impact. And then say that you may distribute a few of these onto the bug tracker here somehow, distributed somewhat slowly so that they would not take too much time away (I think people also prefer to add features since it is something new, whereas fixing bugs is not quite as fun as playing around with new features :D).

Developer time is limited and I think different members of the ruby core team said it before, that often priority will be given to "real" problems, as opposed to theoretical problems or ... not sure about the word ... very unlikely bugs?

Anyway, perhaps some people like to look at CI to discover "worthwhile" bugs to fix. A few bug reports in the last ~3 months really amazed me, how they were found. Some other bugs (or "surprising behaviour") are interesting too, like the one where << can change the encoding of Strings (https://bugs.ruby-lang.org/issues/14975), even though that one was found by regular use of ruby rather than any automated or semi-automated CI (may not be a bug and perhaps just a specific behaviour but it still surprised me when I read it).

#5 - 10/05/2018 09:09 AM - jaruga (Jun Aruga)

- Description updated

#6 - 10/05/2018 02:59 PM - jaruga (Jun Aruga)

Yusuke and Robert, thank you for sharing current status and thoughts.

> I had run the Coverity Scan analysis on CI (twice a week), and I had checked the result only when I felt like. But recently I forgot it completely. By this ticket, I have just noticed that the analysis has not worked since Feb. 2018 :-)

Alright :-) The CI is with Travis CI or a CI in rubyci.org?

> I think the ruby core team always likes to remove bugs from ruby, if these reports constitute real bugs that is. At the github page of mruby one can see that some sort of systematically try to find ways to break ruby, if you look at some of the issues there. :P

I found the issues on mruby project. They are doing something :)

https://github.com/mruby/mruby/commit/8da787b
https://github.com/mruby/mruby/commit/a4f63ca
https://github.com/mruby/mruby/commit/b071dcd

> But I think developer manpower is still limited, in numbers alone.
> Perhaps not only in numbers but in knowledge too. I am sure there are many people who
> know ruby quite well, but significantly fewer who know both ruby and C very well.

One of the benefits to access the result of Coverity Scan casually is that it can be good opportunity for people like me who do not know C very well, to make friends with C in Ruby project.
Because fixing the some issues is easier than implementing new features. Though it might not be a real important issue.

That might increase future potential developer with C in Ruby project.
Recently I fixed a GCC warning issue for Ruby, it was the good opportunity for me.

> What may be useful, perhaps, is if there could be some way to not only find "real"
> bugs, but those that seem to be more "promising" and worthwhile to fix. Like to
> somehow semi-automatically "promote" the more outstanding bugs from Coverity Scan
> or any other method that may have a real, larger impact.

I think that it might be possible to pick up only "real" bugs from the result of Coverity Scan semi-automatically.

> Developer time is limited and I think different members of the ruby core team said
> it before, that often priority will be given to "real" problems, as opposed to
> theoretical problems or ... not sure about the word ... very unlikely bugs?

I can understand the meaning.

#7 - 10/13/2018 10:33 AM - jaruga (Jun Aruga)

I like to share a good example that they only run Coverity Scan for Travis's cron job.
https://github.com/systemd/systemd/blob/master/.travis.yml

#8 - 10/25/2018 10:50 AM - jaruga (Jun Aruga)

> Instead of that, It looks good to me that someone could see the result of coverity scan casually anytime to fix those in long term.

Current Travis CI "pedanticism" test case to output compiler warnings with -Wall and -Wextra is very helpful to see issues included in a result of Coverity Scan possibly.
Thank you for adding the test case!

See also https://github.com/ruby/ruby/blob/trunk/.travis.yml

```
  - name: pedanticism
...
      warnflags_array=(
          -Wall
          -Wextra
...
```