Ruby - Feature #16131

Remove $SAFE, taint and trust

08/29/2019 07:14 AM - naruse (Yui NARUSE)

Status:	Closed
Priority:	Normal
Assignee:	
Target version:	

Description

Ruby had Taint checking which is originally introduced in Perl.
https://en.wikipedia.org/wiki/Taint_checking

It was intended to provide a useful tool for handle objects which are come from outside.
Input data is set as tainted by default and call untaint if you checked or filtered the value.
Some people used this feature in the age of CGI.

But these days, no one use the mechanism and input libraries usually doesn't support it.
For example rack, as following shows its input is not tainted and the mechanism is unusable.

```
% cat foo.ru
run ->(env) do
  ['200', {'Content-Type' => 'text/plain'}, ["Is QUERY_STRING tainted?: #{env["QUERY_STRING"].tain
ted?}"]]
end
% rackup foo.ru
[51724] Puma starting in cluster mode...
[51724] * Version 3.12.1 (ruby 2.6.3-p62), codename: Llamas in Pajamas
[51724] * Min threads: 3, max threads: 3
[51724] * Environment: development
[51724] * Process workers: 1
[51724] * Preloading application
[51724] * Listening on tcp://localhost:9292
[51724] Use Ctrl-C to stop
[51737] + Gemfile in context: /Users/naruse/work/td-cdp-api/Gemfile
[51724] - Worker 0 (pid: 51737) booted, phase: 0

% curl http://localhost:9292/\?foo=1
Is QUERY_STRING tainted?: false
```

Therefore I think Taint checking mechanism is unusable on the current Ruby ecosystem.

On the other hand we experienced multiple vulnerability around $SAFE and taint mechanism.
https://cse.google.com/cse?q=taint&cx=008288045305770251182%3Afvruzsaknew&ie=UTF-8
The cost of maintaining it is expensive.

In conclusion, I think the taint mechanism is too expensive to maintain for the merit of it.
I suggest to remove it.

Related issues:

Related to Ruby - Feature #15998: Allow String#-@ to deduplicate tainted stri...	**Closed**	
Related to Ruby - Feature #8468: Remove $SAFE	**Closed**	**06/01/2013**
Related to Ruby - Bug #9588: program name variables tainted	**Closed**	

Associated revisions

Revision 9594f57f3df6c2538f96f018fa5f9a775ac7dde1 - 11/11/2019 11:31 PM - mame (Yusuke Endoh)

test/ruby/test_require.rb: Remove the tests of require with $SAFE

The taint mechanism is decided to be removed at 2.7. [Feature #16131]
So, this change removes the tests that expects a SecurityError when
requiring a file under $SAFE >= 1.

The reason why they should be removed in advance is because the upstream

of rubygems has already removed a call to "untaint" method, which makes
the tests fail.

Revision 9594f57f3df6c2538f96f018fa5f9a775ac7dde1 - 11/11/2019 11:31 PM - mame (Yusuke Endoh)

test/ruby/test_require.rb: Remove the tests of require with $SAFE

The taint mechanism is decided to be removed at 2.7. [Feature #16131]
So, this change removes the tests that expects a SecurityError when
requiring a file under $SAFE >= 1.

The reason why they should be removed in advance is because the upstream
of rubygems has already removed a call to "untaint" method, which makes
the tests fail.

Revision 9594f57f - 11/11/2019 11:31 PM - mame (Yusuke Endoh)

test/ruby/test_require.rb: Remove the tests of require with $SAFE

The taint mechanism is decided to be removed at 2.7. [Feature #16131]
So, this change removes the tests that expects a SecurityError when
requiring a file under $SAFE >= 1.

The reason why they should be removed in advance is because the upstream
of rubygems has already removed a call to "untaint" method, which makes
the tests fail.

History

#1 - 08/29/2019 07:15 AM - naruse (Yui NARUSE)

- Related to Feature #15998: Allow String#-@ to deduplicate tainted string, but return an untainted one added

#2 - 08/29/2019 07:16 AM - naruse (Yui NARUSE)

- Related to Feature #8468: Remove $SAFE added

#3 - 08/29/2019 03:46 PM - jeremyevans0 (Jeremy Evans)

I agree with the removal of $SAFE and the taint tracking. Proposed timeline:

2.7:

- Remove taint tracking/mechanism.
- Non-verbose warning on setting/access of $SAFE
- taint/trust/untaint/untrust become no-ops, verbose warning when called

3.0:

- No warning on setting/access of $SAFE, it switches to normal global variable.

3.2:

- taint/trust/untaint/untrust non-verbose warning when called

3.3:

- taint/trust/untaint/untrust removed

The reasoning behind the delayed removal of the taint/trust/untaint/untrust methods is that most gems want to support all currently supported Ruby versions, and removing these methods soon may make that more difficult.

#4 - 08/29/2019 06:49 PM - byroot (Jean Boussier)

> 3.2 taint/trust/untaint/untrust non-verbose warning when called

Maybe you meant verbose here?

Other than that I agree with the proposed timeline, and as soon as these methods are noop, their cost become mostly null.

Making them noop also allow for easy feature testing: Object.new.taint.tainted? # => wether or not tainting is supported.

#5 - 08/29/2019 07:43 PM - jeremyevans0 (Jeremy Evans)

byroot (Jean Boussier) wrote:

> 3.2 taint/trust/untaint/untrust non-verbose warning when called
>
> Maybe you meant verbose here?

No. Verbose warning means a warning only printed in verbose mode (ruby -w, or $VERBOSE = true). Non-verbose warning means a warning printed even in regular mode.

#6 - 08/30/2019 04:41 AM - mame (Yusuke Endoh)

+1 for the removal, and I agree with Jeremy's plan for 2.7 and 3.0.
For 3.2 and 3.3, I think we may keep all the methods as no-op because old not-maintained-well scripts may break, though I'm not so strongly against the removal.
(Anyway, tainted? and trusted? should be also cared.)

#7 - 08/30/2019 08:49 AM - hsbt (Hiroshi SHIBATA)

I'm also +1 for jeremy's proposal.

I often got the test fails related $SAFE on rubygems. I'm happy to leave them with this proposal.

#8 - 08/30/2019 04:19 PM - Dan0042 (Daniel DeLorme)

I must admit to using taint sometimes in my code, as a way to keep track of dirty/modified status on an object (mea culpa)

```
hash.taint[key] = newvalue
...
save(hash.untaint) if hash.tainted?
```

It's probably not common at all. Still, I think since tainted state has been there for such a long time we should not introduce backwards incompatibility (making it a no-op) right away in 2.7. Adding a deprecation warning in 2.7 and then making it a no-op in 3 should be the usual way of handling deprecation no? Although removing the interaction with $SAFE seems ok to me even for 2.7.

#9 - 08/30/2019 04:57 PM - Dan0042 (Daniel DeLorme)

@jeremyevans0 (Jeremy Evans), by "no-op" did you mean only in the context of $SAFE mode, or did you mean that tainted? and trusted? would always return false? In the second case I think it's better to just remove the method, at least that's an obvious and easy bug to fix.

#10 - 08/30/2019 05:29 PM - jeremyevans0 (Jeremy Evans)

By no-op, I meant they would make no changes and return self. I didn't mention tainted? or trusted? earlier, but I think it may make sense to remove them earlier than taint/trust/untaint/untrust. Maybe a non-verbose warning stating they always return false in 2.7, and then remove them in 3.0. The reason for the different behavior is that taint/trust/untaint/untrust are often called by code without caring what they actually do (other than to make the objects work with certain core methods). tainted?/trusted? are only called when the code wants to have different behavior based on the taint flag.

For tainted?/trusted? to work correctly, we would need to continue to support taint tracking at least in some state. We could reduce the scope of the taint flag, though. For example, we could make it so the taint flag is never checked by any core/stdlib code, and never transfered to another object. However calling taint/trust/untaint/untrust on an object and then calling tainted?/trusted? on the same object will still behave as it does in 2.6. That would allow your abuse of taint for dirty tracking to continue to work in 2.7. If we do that, I think we should still add a non-verbose warning in 2.7 when tainted?/trusted? are called, and remove tainted?/trusted? in 3.0.

#11 - 08/30/2019 05:45 PM - Dan0042 (Daniel DeLorme)

jeremyevans0 (Jeremy Evans) wrote:

> For tainted?/trusted? to work correctly, we would need to continue to support taint tracking at least in some state. We could reduce the scope of the taint flag, though. For example, we could make it so the taint flag is never checked by any core/stdlib code, and never transfered to another object. However calling taint/trust/untaint/untrust on an object and then calling tainted?/trusted? on the same object will still behave as it does in 2.6. That would allow your abuse of taint for dirty tracking to continue to work in 2.7. If we do that, I think we should still add a non-verbose warning in 2.7 when tainted?/trusted? are called, and remove tainted?/trusted? in 3.0.

That sounds good to me. At that point you could even replace the taint/trust bit flags by instance variables.

#12 - 09/02/2019 05:36 AM - ko1 (Koichi Sasada)

- Related to Bug #9588: program name variables tainted added

#13 - 09/02/2019 05:57 AM - mame (Yusuke Endoh)

@headius (Charles Nutter) @Eregon (Benoit Daloze) @brixen

Do you have any opinion about this as developers of other Ruby implementations?

#14 - 09/07/2019 11:40 AM - Eregon (Benoit Daloze)

I agree it would be best to remove the implicit taint state, and particularly the interaction with $SAFE.

FWIW, TruffleRuby already prevents setting $SAFE to anything else than 0:
https://github.com/oracle/truffleruby/blob/master/doc/user/security.md#unimplemented-security-features

Without $SAFE (which I think most people agree to remove), I think tainting has very few use-cases, which I think doesn't warrant staying a core feature.

Tracking tainting has a performance cost, e.g., String#+ must check if either LHS or RHS is tainted and taint the result in that case.
This can introduce extra polymorphism or branches in code which needs to check for the taint state.

#15 - 09/19/2019 08:00 AM - matz (Yukihiro Matsumoto)

Basically agreed.
My proposal for the schedule:

2.7:

- Remove taint tracking/mechanism.
- Non-verbose warning on setting/access of $SAFE
- taint/trust/untaint/untrust become no-ops, verbose warning when called

3.0:

- No warning on setting/access of $SAFE, it switches to normal global variable.
- taint/trust/untaint/untrust non-verbose warning when called

3.2:

- taint/trust/untaint/untrust removed

But it's not a big issue.

Matz.

#16 - 09/19/2019 01:26 PM - headius (Charles Nutter)

I look forward to removing all tainting logic!

#17 - 09/21/2019 07:17 AM - jeremyevans0 (Jeremy Evans)

I've added a pull request that adds warnings to setting/access of $SAFE, as well as public C function that deal with $SAFE:
https://github.com/ruby/ruby/pull/2476

As the taint tracking/mechanism is being removed, I was not sure if we want to keep any other features of $SAFE. The pull request does not keep any features, after it is applied, nothing in the core or stdlib uses $SAFE. I think that is what was desired, but I'm not sure, as the log for the last developer meeting hasn't been released yet.

#18 - 09/25/2019 04:08 AM - jeremyevans0 (Jeremy Evans)

I've expanded my pull request to deprecate taint/trust and related methods with verbose warnings, and make the methods no-ops. I believe this implements matz's plan for Ruby 2.7.

The changes involved removing tainting from all included libraries, which includes libraries such as rubygems, bundler, and json, that may want to support older versions of ruby upstream (and may need to keep taint code to work correctly in older ruby versions). I'm not sure how we want to handle this, and I'm open to ideas.

#19 - 10/04/2019 04:17 PM - jeremyevans0 (Jeremy Evans)

I've rebased my pull request against master and fixed the conflicts (https://github.com/ruby/ruby/pull/2476). I've also removed mentions of $SAFE and taint from the documentation.

Due to the extent of the changes, I don't want to wait too long before merging this. Otherwise, there will probably be more conflicts to resolve, and increased chance of a untaint/taint call being introduced. Also due to the extent of the changes, another committer should review.

We still need to decide how we want to handle upstreams that want to support older ruby versions. Do we want to just notify upstreams and request that they fix it? Do we want to recommend a specific approach, such as (for rubygems):

```
if RUBY_VERSION >= '2.7'
  def Gem.untaint_obj(obj)
  end
```

```
else
  def Gem.untaint_obj(obj)
    obj.untaint
  end
end
```

And changing all the calls? Or wrapping all calls in if RUBY_VERSION < '2.7'

test-bundled-gems is failing with this patch (a single rake test). I submitted a patch upstream to skip that test on Ruby 2.7+: https://github.com/ruby/rake/pull/329

#20 - 10/17/2019 06:58 AM - mame (Yusuke Endoh)

Hi @jeremyevans0 (Jeremy Evans),

> I've rebased my pull request against master and fixed the conflicts

Thank you for the great work! I've discussed this issue on the developer meeting, and all agreed with the change.

> We still need to decide how we want to handle upstreams that want to support older ruby versions.

This should be discussed and agreed with the maintainers for each code (rubygems, bundler, etc). In regard to rubygems and bundler, I hear from @hsbt (Hiroshi SHIBATA) that the incompatibility would not matter even if we just remove the code related to $SAFE. (@hsbt (Hiroshi SHIBATA), am I correct?)

#21 - 10/17/2019 03:52 PM - jeremyevans0 (Jeremy Evans)

The blocker on merging the pull request is that test-bundled-gems is failing due to the rake test failure. https://github.com/ruby/rake/pull/329 needs to be merged (and I don't have permissions to merge it), and a new rake released and bundled with Ruby.

I checked and Bundler and Rubygems are the only libraries affected that use external upstreams. All other affected libraries (default gems) are under the ruby organization on GitHub. We need to decide how we want to handle these:

Default gems without extensions

```
fileutils
irb
reline
rexml
rss
webrick
```

Default gems with extensions:

```
bigdecimal
date
dbm
etc
fiddle
gdbm
io-console
openssl
psych
stringio
strscan
zlib
```

Are we OK with just removing the calls to taint/untaint? I'm not sure, but I believe that may cause issues when using previous versions of Ruby. The simplest fix here is to set the required ruby version in the related gemspecs to 2.6.99 to allow 2.7.0 preview/beta versions and above to work. That will mean older versions of Ruby cannot install newer versions of the gems. Is that acceptable?

#22 - 10/18/2019 03:28 AM - mame (Yusuke Endoh)

> Are we OK with just removing the calls to taint/untaint?

Each maintainer should determine that.

This is my personal opinion: In principle, we should be conservative against incompatibility. But in regard to $SAFE, we can be flexible because it seems really rare to be used.

Anyway, I'd like to keep no warnings in CI even in verbose mode.

#23 - 10/18/2019 05:44 AM - jeremyevans0 (Jeremy Evans)

mame (Yusuke Endoh) wrote:

> > Are we OK with just removing the calls to taint/untaint?
>
> Each maintainer should determine that.
>
> This is my personal opinion: In principle, we should be conservative against incompatibility. But in regard to $SAFE, we can be flexible because it seems really rare to be used.
>
> Anyway, I'd like to keep no warnings in CI even in verbose mode.

I agree with your points. Here is my implementation plan:

- I will submit pull requests upstream to all projects that remove the calls and bump the required ruby version to 2.6.99.
- For upstreams without a maintainer, I will wait one week to allow input from the community, and assuming no input, I will merge the changes.
- If the upstream has a maintainer, and the maintainer requests different behavior, I will work with them to implement their desired behavior.
- If the upstream has a maintainer, and the maintainer doesn't respond in one month, I will merge the changes (assuming I have access to do so).

This plan should ensure that all upstreams are consulted and all maintainers can choose the path they feel is best. It should also ensure the changes can be merged in time for Ruby 2.7. Is this plan acceptable?

#24 - 10/18/2019 10:26 PM - jeremyevans0 (Jeremy Evans)

I have added pull requests for all upstream projects. After some thought, I think many maintainers may consider dropping Ruby <2.7 support not acceptable. So the pull requests I submitted will continue to work on older Ruby versions. In cases where untaint is used, that means using a conditional, because the calling code may want an untainted string. In cases where taint or tainted? is used, those were generally just removed. While that does change behavior slightly, it is unlikely anyone is relying on things being tainted (they may relying on things not being tainted).

Here are links to all pull requests:

Bundled gems with external upstreams:

- rake: https://github.com/ruby/rake/pull/329

Default gems with external upstreams:

- bundler: https://github.com/bundler/bundler/pull/7385
- rubygems: https://github.com/rubygems/rubygems/pull/2951

Default gems without C extensions:

- fileutils: https://github.com/ruby/fileutils/pull/45
- irb: https://github.com/ruby/irb/pull/30
- reline: https://github.com/ruby/reline/pull/61
- rexml: https://github.com/ruby/rexml/pull/21
- rss: https://github.com/ruby/rss/pull/7
- webrick: https://github.com/ruby/webrick/pull/34

Default gems with C extensions:

- bigdecimal: https://github.com/ruby/bigdecimal/pull/157
- date: https://github.com/ruby/date/pull/14
- dbm: https://github.com/ruby/dbm/pull/4
- etc: https://github.com/ruby/etc/pull/5
- fiddle: https://github.com/ruby/fiddle/pull/21
- gdbm: https://github.com/ruby/gdbm/pull/3
- io-console: https://github.com/ruby/io-console/pull/6
- openssl: https://github.com/ruby/openssl/pull/273
- psych: https://github.com/ruby/psych/pull/419
- stringio: https://github.com/ruby/stringio/pull/6
- strscan: https://github.com/ruby/strscan/pull/11
- zlib: https://github.com/ruby/zlib/pull/9

#25 - 10/30/2019 08:54 PM - jeremyevans0 (Jeremy Evans)

Most of the pull requests to fix taint/$SAFE issues have been merged. These are the remaining ones that haven't been merged yet:

Bundled gems with external upstreams:

- rake: https://github.com/ruby/rake/pull/329 (Can one of the rack maintainers merge and bump version?)

Default gems without C extensions:

- irb: https://github.com/ruby/irb/pull/30
- reline: https://github.com/ruby/reline/pull/61

Default gems with C extensions:

- bigdecimal: https://github.com/ruby/bigdecimal/pull/157
- psych: https://github.com/ruby/psych/pull/419

#26 - 11/11/2019 05:14 PM - mame (Yusuke Endoh)

Hi @jeremyevans0 (Jeremy Evans) , thank you for your great work.

I might be one lap behind, but as far as I undestand, the taint tracking will be removed in 2.7. However, it looks still enabled:

```
$ ./miniruby -e '$SAFE=1; File.symlink?("/etc/passwd".taint)'
Traceback (most recent call last):
    1: from -e:1:in `<main>'
-e:1:in `symlink?': Insecure operation - symlink? (SecurityError)
```

Rubygems removed untaint operations, which leads to Insecure operation - symlink? error in rubygems test suite:

```
  1) Failure:
TestRequire#test_require_insecure_path [/home/hsbt/chkbuild/tmp/build/20191111T153007Z/ruby/test/ruby/test_req
uire.rb:66]:
Expected "Insecure operation - symlink?" to include "loading from unsafe path".

  2) Failure:
TestRequire#test_require_insecure_path_shift_jis [/home/hsbt/chkbuild/tmp/build/20191111T153007Z/ruby/test/rub
y/test_require.rb:94]:
Expected "Insecure operation - symlink?" to include "loading from unsafe path".
```

https://rubyci.org/logs/rubyci.s3.amazonaws.com/debian8/ruby-master/log/20191111T153007Z.fail.html.gz

Thanks,

#27 - 11/11/2019 05:55 PM - jeremyevans0 (Jeremy Evans)

mame (Yusuke Endoh) wrote:

> Hi @jeremyevans0 (Jeremy Evans) , thank you for your great work.
>
> I might be one lap behind, but as far as I undestand, the taint tracking will be removed in 2.7. However, it looks still enabled:
>
> ```
> $ ./miniruby -e '$SAFE=1; File.symlink?("/etc/passwd".taint)'
> Traceback (most recent call last):
>     1: from -e:1:in `<main>'
> -e:1:in `symlink?': Insecure operation - symlink? (SecurityError)
> ```
>
> Rubygems removed untaint operations, which leads to Insecure operation - symlink? error in rubygems test suite:
>
> ```
>   1) Failure:
> TestRequire#test_require_insecure_path [/home/hsbt/chkbuild/tmp/build/20191111T153007Z/ruby/test/ruby/test
> _require.rb:66]:
> Expected "Insecure operation - symlink?" to include "loading from unsafe path".
>
>   2) Failure:
> TestRequire#test_require_insecure_path_shift_jis [/home/hsbt/chkbuild/tmp/build/20191111T153007Z/ruby/test
> /ruby/test_require.rb:94]:
> Expected "Insecure operation - symlink?" to include "loading from unsafe path".
> ```
>
> https://rubyci.org/logs/rubyci.s3.amazonaws.com/debian8/ruby-master/log/20191111T153007Z.fail.html.gz
>
> Thanks,

I haven't committed the changes to Ruby core yet. Committing the Ruby core changes first would have broken it as well. I will try to commit the changes later this week. If it cannot wait that long, please let me know, but I'll be traveling and not able to do much for the next ~36 hours.

Unfortunately, there are about 25 separate repositories where changes need to be committed, and for most of those places the changes need to be backwards compatible with earlier versions, which wasn't part of the initial branch prepared. So for each of those repositories, the changes in the initial branch need to be backed out before merging. This is one of the negative aspects of gemifying the standard library and moving each library to its own repository. Additionally, more of the standard library got moved to gems since I prepared the per-gem commits, so I need to recheck all of those libraries and see if they are affected by the taint removal.

#28 - 11/11/2019 06:43 PM - Dan0042 (Daniel DeLorme)

Wait, I don't understand. You should be able to just leave str.untaint like it is since it's just a no-op in 2.7. Why the version check?

#29 - 11/11/2019 06:56 PM - jeremyevans0 (Jeremy Evans)

Dan0042 (Daniel DeLorme) wrote:

> Wait, I don't understand. You should be able to just leave str.untaint like it is since it's just a no-op in 2.7. Why the version check?

There is a verbose warning emitted if you call the method in 2.7, so we can't have anything in the core/stdlib calling it.

#30 - 11/11/2019 11:42 PM - mame (Yusuke Endoh)

- Status changed from Open to Closed

Applied in changeset git|9594f57f3df6c2538f96f018fa5f9a775ac7dde1.

---

test/ruby/test_require.rb: Remove the tests of require with $SAFE

The taint mechanism is decided to be removed at 2.7. [Feature #16131]
So, this change removes the tests that expects a SecurityError when
requiring a file under $SAFE >= 1.

The reason why they should be removed in advance is because the upstream
of rubygems has already removed a call to "untaint" method, which makes
the tests fail.

#31 - 11/11/2019 11:50 PM - mame (Yusuke Endoh)

- Status changed from Closed to Open

Oops, I closed it unintentionally. Reopening.

jeremyevans0 (Jeremy Evans) wrote:

> I haven't committed the changes to Ruby core yet. Committing the Ruby core changes first would have broken it as well. I will try to commit the changes later this week. If it cannot wait that long, please let me know, but I'll be traveling and not able to do much for the next ~36 hours.

Thanks, I understand! I have removed the failed tests of test/ruby/test_require.rb, which would be eventually removed because they check if "require" raises a SecurityError under $SAFE=1. So, currently there is no test failures.

I checked the status of your PRs:

- rake: already merged; a new version need to be released
- irb: already merged and backported to trunk
- reline: already merged and backported to trunk
- bigdecimal: already merged; but not backported yet to trunk
- psych: already merged; but not backported yet to trunk

@hsbt (Hiroshi SHIBATA) said that he will manage rake, bigdecimal, and psych. I hope you will be able to remove $SAFE mechanism when you return home :-) Have a nice travel!

#32 - 11/12/2019 03:50 AM - hsbt (Hiroshi SHIBATA)

I released Rake 13.0.1 and merged Jeremy's commits related untaint on bigdecimal and psych.

#33 - 11/15/2019 02:57 PM - jeremyevans0 (Jeremy Evans)

I updated https://github.com/ruby/ruby/pull/2476. There are a couple failing CI tests, both of which appear unrelated:

- https://ci.appveyor.com/project/ruby/ruby/builds/28875336/job/6udjor0n25yvgaan
- https://travis-ci.org/ruby/ruby/jobs/612185187?utm_medium=notification&utm_source=github_status

I had to merge some changes made in separate repositories that had not been merged into ruby yet: rexml, rss, etc, io-console, openssl, strscan

If another developer could review and let me know if it looks OK to merge, I would appreciate it.

#34 - 11/17/2019 11:14 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Open to Closed

I merged these changes at 4c7dc9fbe604cc0c8343b1225c96d4e5219b8147 . Still one failing CI test, but the same one that is failing in the master branch for a few days, related to makefile dependencies.

#35 - 11/30/2019 09:27 AM - hsbt (Hiroshi SHIBATA)

I released the new versions: fileutils, webrick, date, dbm, etc, gdbm, stringio, zlib.

@kou (Kouhei Sutou) rexml, rss, fiddle, strscan
@nobu (Nobuyoshi Nakada) io-console

Can you release the new versions contained to drop taint support? and Can you import upstream version to ruby-core repository before Ruby 2.7.0-rc1 release.