Ruby - Feature #9439

Remove OpenSSL from stdlib

01/22/2014 02:59 AM - zzak (zzak _)

Status:	Rejected	
Priority:	Normal	
Assignee:		
Target version:	2.2.0	

Description

Regarding [ruby-core:59943], I agree with nobu that we should remove OpenSSL from ruby.

It's become too hard to maintain, and would better serve our users to encourage the use of a different implementation.

Another benefit of removing OpenSSL is the impact backport fixes have on the release management team.

Although I haven't yet determined the extent of work required to remove it (ie: tooling, tests, etc). We can discuss them here.

History

#1 - 01/22/2014 02:59 AM - zzak (zzak _)

See also #9424

#2 - 01/22/2014 03:22 AM - shyouhei (Shyouhei Urabe)

:+1:

Nursing OpenSSL's being bad is beyond our power.

#3 - 01/22/2014 03:59 AM - Anonymous

How will RubyGems use HTTPS without OpenSSL?

#4 - 01/22/2014 06:15 AM - naruse (Yui NARUSE)

Charlie Somerville wrote:

> How will RubyGems use HTTPS without OpenSSL?

Good point.
This proposal should include the answer of this issue.

There are some ways for example use package manager of the OS or a browser or curl or something and download openssl.gem.
Use rvm and rvm will include openssl.gem as the part of it can be also a way.

#5 - 01/22/2014 11:40 AM - usa (Usaku NAKAMURA)

Yui NARUSE wrote:

> There are some ways for example use package manager of the OS or a browser or curl or something and download openssl.gem.
> Use rvm and rvm will include openssl.gem as the part of it can be also a way.

If it is right, don't we have the necessity of bundling rubygems itself any longer?

#6 - 01/22/2014 06:08 PM - shyouhei (Shyouhei Urabe)

Why must RubyGems use https? Shouldn't it suffice to check the sanity of downloaded packages?

I mean, can't we live without the secure socket? All what gem need is the digital sigunature.

#7 - 01/22/2014 06:11 PM - shyouhei (Shyouhei Urabe)

> All what gem need is the digital sigunature.

To be precise it only needs to verify signatures. Signing itself can be done using other tools, like gpg(1).

#8 - 01/22/2014 06:13 PM - luislavena (Luis Lavena)

Shyouhei Urabe wrote:

> > All what gem need is the digital sigunature.
>
> To be precise it only needs to verify signatures. Signing itself can be done using other tools, like gpg(1).

That means gpg becomes an external dependency of the build/integration process.

gpg is not available in all the platforms.

There has been a bunch of research and investigation in relation to trusted RubyGems, I strongly recommend that is been analyzed prior the decision to remove such critical package like OpenSSL is made.

#9 - 01/22/2014 06:26 PM - shyouhei (Shyouhei Urabe)

Luis Lavena wrote:

> Shyouhei Urabe wrote:
>
> > > All what gem need is the digital sigunature.
> >
> > To be precise it only needs to verify signatures. Signing itself can be done using other tools, like gpg(1).
>
> That means gpg becomes an external dependency of the build/integration process.
>
> gpg is not available in all the platforms.
>
> There has been a bunch of research and investigation in relation to trusted RubyGems, I strongly recommend that is been analyzed prior the decision to remove such critical package like OpenSSL is made.

Very true. I have no idea on why RubyGems use https and not other tools. Any pointers?

#10 - 01/22/2014 06:39 PM - luislavena (Luis Lavena)

Shyouhei Urabe wrote:

> Very true. I have no idea on why RubyGems use https and not other tools. Any pointers?

AFAIK is to avoid MITM attacks and such, since if signatures are also stored along packages, how can you verify that the packages are not altered?

You will say why RubyGems don't use something else, well, why will you impose gpg to all the users using RubyGems, even when the tools is not available?

Back in november there was an initiative to implement TUF on top of RubyGems, please see the following link:

http://rubyforge.org/pipermail/rubygems-developers/2013-November/thread.html

More and better responses on this can be provided by Eric Hodel

But again, removal of OpenSSL is not the answer to what some core committers see as personal attacks around OpenSSL defaults.

#11 - 01/22/2014 09:28 PM - drbrain (Eric Hodel)

I'm not informed of the details of how TUF works, but the implementation in progress uses OpenSSL to verify metadata and packages, so Ruby will still require OpenSSL.

RubyGems already supports signing gems using OpenSSL, but there are numerous usability issues which have prevented this from becoming widely used. GPG signing suffers from the same issues. TUF is designed to avoid these issues which means RubyGems can provide usable security for users.

Even after TUF is deployed RubyGems will still need HTTPS for secure communication with legacy private repositories that haven't switched to using TUF.

I don't know how RubyGems can work without HTTPS connections for backwards compatibility.

http://theupdateframework.github.io has information on the TUF specification and reference implementation. The rubygems-tuf mailing list is the place to ask questions about how TUF will work with RubyGems: https://groups.google.com/forum/#!forum/rubygems-tuf

#12 - 01/22/2014 10:10 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

Em 22-01-2014 16:39, luislavena@gmail.com escreveu:

> Issue #9439 has been updated by Luis Lavena.
>
> Shyouhei Urabe wrote:
>
> > Very true. I have no idea on why RubyGems use https and not other tools. Any pointers?
> > AFAIK is to avoid MITM attacks and such, since if signatures are also stored along packages, how can you verify that the packages are not altered?

I've always been curious about that, specially because the introduction of https://rubygems.org slowed down our deploys considerably and it's way slower to run bundler over the https version when compared to the regular http version.

If the only concern is about MITM attacks and if the reason for the much slower gems downloading is because they are being served through an HTTPS connection, then it would probably be much faster if we only got the list of gems signature over an HTTPS connection, That way we'd be able to download the gems over regular http and then calculate the checksum and verify against the list of checksums downloaded from the secure connection. Or we could download the public key that signed all gems (in the case rubygems.org itself signed all gems) from a secure location (https) and perform all checks locally. Wouldn't that work and be much faster than the current alternative?

#13 - 01/22/2014 11:38 PM - samkottler (Sam Kottler)

Rodrigo Rosenfeld Rosas wrote:

> Em 22-01-2014 16:39, luislavena@gmail.com escreveu:
>
> > Issue #9439 has been updated by Luis Lavena.
> >
> > Shyouhei Urabe wrote:
> >
> > > Very true. I have no idea on why RubyGems use https and not other tools. Any pointers?
> > > AFAIK is to avoid MITM attacks and such, since if signatures are also stored along packages, how can you verify that the packages are not altered?
>
> I've always been curious about that, specially because the introduction of https://rubygems.org slowed down our deploys considerably and it's way slower to run bundler over the https version when compared to the regular http version.
>
> If the only concern is about MITM attacks and if the reason for the much slower gems downloading is because they are being served through an HTTPS connection, then it would probably be much faster if we only got the list of gems signature over an HTTPS connection, That way we'd be able to download the gems over regular http and then calculate the checksum and verify against the list of checksums downloaded from the secure connection. Or we could download the public key that signed all gems (in the case rubygems.org itself signed all gems) from a secure location (https) and perform all checks locally. Wouldn't that work and be much faster than the current alternative?

I'm one of the maintainers of rubygems.org and bundler and haven't ever heard this complaint before. Can you email me with more info about your environment? https should not be notably slower than http. And yes, your solution works, but the problem is that we currently don't have a mechanism for matching checksums to binaries AFAIK. So while possible in the theoretical sense, it's not as straight forward as you make it seem.

#14 - 01/22/2014 11:41 PM - samkottler (Sam Kottler)

Shyouhei Urabe wrote:

> All what gem need is the digital sigunature.

> To be precise it only needs to verify signatures. Signing itself can be done using other tools, like gpg(1).

Not really. GPG implementations are platform specific and require a higher level of end-user involvement than just plain SSL. Additionally, we need a way to handle secure public-key delivery and SSL is simply the most simple, and bulletproof way to do that.

#15 - 01/23/2014 02:02 AM - shyouhei (Shyouhei Urabe)

Sam Kottler wrote:

> Shyouhei Urabe wrote:
>
> > All what gem need is the digital sigunature.
>
> To be precise it only needs to verify signatures. Signing itself can be done using other tools, like gpg(1).
>
> Not really. GPG implementations are platform specific and require a higher level of end-user involvement than just plain SSL. Additionally, we need a way to handle secure public-key delivery and SSL is simply the most simple, and bulletproof way to do that.

RubyGems can include its public key in its canonical distribution.
Regardless of their doing so or not, never use a ruby source code
that you cannot trust. Including public key(s) in distribution is
not a practically complicated thing to do.
Of course, RubyGems itself might have to be downloaded using https.
but that can be done without ruby's supporting OpenSSL.

#16 - 01/23/2014 04:22 AM - shyouhei (Shyouhei Urabe)

Forget GPG if you guys don't like it; it's just an alternative.

I had a really rough glance at the TUF pull request and it seems
OpenSSL is used only for SHA and RSA algorithms and not for the
secure socket layer. We can find adequate other implementations
for both SHA and RSA so TUF can be OpenSSL-free I believe.

#17 - 01/23/2014 10:18 AM - databus23 (Fabian Ruff)

I must be missing something here but I feel like I'm going crazy over this.
Are we really talking about removing the capability for https communication from ruby core while the world is shifting towards https-only?

To me this is bigger than how to secure the installation of gems without openssl.
With HTTP being the universal internet protocol I think a programming language has to support the secured version from the get go.

Apologies if I misinterpreted the ramifications of removing OpenSSL from ruby core.

#18 - 01/23/2014 11:18 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

Em 22-01-2014 21:38, sam@kottlerdevelopment.com escreveu:

> Issue #9439 has been updated by Sam Kottler.
>
> Rodrigo Rosenfeld Rosas wrote:
>
> > Em 22-01-2014 16:39, luislavena@gmail.com escreveu:
> >
> > > Issue #9439 has been updated by Luis Lavena.
> > >
> > > Shyouhei Urabe wrote:
> > >
> > > > Very true. I have no idea on why RubyGems use https and not other tools. Any pointers?
> > > > AFAIK is to avoid MITM attacks and such, since if signatures are also stored along packages, how can you verify that the packages are not altered?
>
> I've always been curious about that, specially because the introduction of https://rubygems.org slowed down our deploys considerably and it's way slower to run bundler over the https version when compared to the regular http version.
>
> If the only concern is about MITM attacks and if the reason for the much

> slower gems downloading is because they are being served through an HTTPS connection, then it would probably be much faster if we only got the list of gems signature over an HTTPS connection, That way we'd be able to download the gems over regular http and then calculate the checksum and verify against the list of checksums downloaded from the secure connection. Or we could download the public key that signed all gems (in the case rubygems.org itself signed all gems) from a secure location (https) and perform all checks locally. Wouldn't that work and be much faster than the current alternative?
> I'm one of the maintainers of rubygems.org and bundler and haven't ever heard this complaint before. Can you email me with more info about your environment? https should not be notably slower than http. And yes, your solution works, but the problem is that we currently don't have a mechanism for matching checksums to binaries AFAIK. So while possible in the theoretical sense, it's not as straight forward as you make it seem.

I've sent the difference in total time (7m for https vs 5m for http) off the list, but there's another thing to consider in my opinion: the use of proxy servers, like Squid.

As far as I understand, proxy servers can't cache https data. So, if you're behind a proxy, either because you work inside a corporation (or with other coworkers) or if you want to test your Vagrant recipes and avoid re-downloading the same gems over and over you could take advantage of the proxy cache if the gems themselves are served through regular http.

This is something to consider, don't you think?

#19 - 01/23/2014 05:49 PM - bascule (Tony Arcieri)

Eric Hodel wrote:

> I'm not informed of the details of how TUF works, but the implementation in progress uses OpenSSL to verify metadata and packages, so Ruby will still require OpenSSL.

Yes, TUF uses the RSASSA PKCS#1v1.5 digital signature algorithm to verify package integrity, and uses the implementation in OpenSSL

#20 - 01/24/2014 05:17 AM - shyouhei (Shyouhei Urabe)

Fabian Ruff wrote:

> I must be missing something here but I feel like I'm going crazy over this.
> Are we really talking about removing the capability for https communication from ruby core while the world is shifting towards https-only?

Yes. I have to admit that ruby devs, especially myself, are not ready. Immature to support OpenSSL.

OpenSSL seemd easy to support at first. We were only needed to wrap the C library with Ruby and that's it. Now, things gets more complicated. People requests us to keep being MORE SECURE THAN the OpenSSL itself. That costs us very much. I studied this topic these days very much and still have no idea how to actually absuse CRIME to get any uncrypted data. It's as clear as sky that I cannot be more secure than the default without actually understand its backgrounds; I lack knowledge, or experience, or maybe both.

People say just providing OpenSSL functionality is not sufficient so we have to work hard to provide something perfect. That might be true. But I'm afraid we can't. If "being secure out of the box" is mandatory, we'd better run away form OpenSSL.

It's just beyond our power.

> To me this is bigger than how to secure the installation of gems without openssl.
> With HTTP being the universal internet protocol I think a programming language has to support the secured version from the get go.
>
> Apologies if I misinterpreted the ramifications of removing OpenSSL from ruby core.

#21 - 01/24/2014 05:33 AM - postmodern (Hal Brodigan)

Just like to point out that unless the indexes and metadata exposed by api.rubygems.org is somehow cryptographically signed, an attacker can MITM rubygems.org and tell clients that rails version 9000 is available for updating.

#22 - 01/24/2014 07:13 PM - mame (Yusuke Endoh)

Shyouhei Urabe wrote:

> OpenSSL seemd easy to support at first. We were only needed to
> wrap the C library with Ruby and that's it. Now, things gets more
> complicated. People requests us to keep being MORE SECURE THAN
> the OpenSSL itself. That costs us very much. I studied this topic
> these days very much and still have no idea how to actually absuse
> CRIME to get any uncrypted data. It's as clear as sky that I cannot
> be more secure than the default without actually understand
> its backgrounds; I lack knowledge, or experience, or maybe both.

Fully agreed. That said, I'm not positive for the proposal as long as Martin Bosslet is willing to maintain the library.

Off-topic: A bigger problem is "kind" people. In an extreme case, we can ignore those who just require something. But it is far more difficult to ignore those who kindly say "I can help you." In fact, they are sometimes much more harmful; they let us take a hard road, saying "don't worry, I'll help you!", and then they disappear.

This time, there are some persons who said so. But I don't see their action after the actual patch appears. Did they read it? Did they verify it? Do they have no opinion? I don't know. They must become busy suddenly. If they would not return, I'd have to say they tried to bullshit us in an innocent way.

Thus, I do not trust such "kind" words. True contributors do never say "I can contribute!", but do contribute first.

--
Yusuke Endoh mame@tsg.ne.jp

#23 - 01/24/2014 08:52 PM - spatulasnout (B Kelly)

mame@tsg.ne.jp wrote:

> Off-topic: A bigger problem is "kind" people. In an extreme case, we can
> ignore those who just require something. But it is far more difficult to
> ignore those who kindly say "I can help you." In fact, they are sometimes
> much more harmful; they let us take a hard road, saying "don't worry, I'll
> help you!", and then they disappear.
>
> This time, there are some persons who said so. But I don't see their
> action after the actual patch appears. Did they read it? Did they verify
> it? Do they have no opinion? I don't know. They must become busy
> suddenly. If they would not return, I'd have to say they tried to
> bullshit us in an innocent way.
>
> Thus, I do not trust such "kind" words. True contributors do never say "I
> can contribute!", but do contribute first.

Ah, but what about people who parachute into the thread, making dispassionate arguments from the sidelines, while offering no help at all?

(Sorry :)

#24 - 01/24/2014 09:05 PM - usa (Usaku NAKAMURA)

I would like to clarify the problem.

As already stated, RubyGems uses OpenSSL.
To say strictly, RubyGems uses OpenSSL for https, signing, and its verification.
Therefore, the option which we can take is as follows:
(1) Maintain the present condition.
(2) Remove OpenSSL and RubyGems together.
(3) Prepare the alternate features of https, signing, and its verification after removing OpenSSL.
(4) Remove the dependence to these features from RubyGems after removing OpenSSL.
(5) Mixture of (3) and (4). That is, remove the dependence to some features from RubyGems, and prepares substitutes about another features.

To my understanding, Shyouhei is taking a position on (5).
That is, changing RubyGems to use plain http in default, and write substitutes for about signing and its verification (with GPG?).

There may be also a position in which (a part of) the features which OpenSSL offers is still required as a part of Ruby, even if RubyGems sets aside.
I understand that Fabian said that the https support itself is required.

How do you think, everyone?
I am in the same position about https support as Fabian, and I think the above (2) is too much nonsense.

#25 - 01/24/2014 11:51 PM - tenderlovemaking (Aaron Patterson)

On Fri, Jan 24, 2014 at 09:05:18PM +0000, usa@garbagecollect.jp wrote:

> Issue #9439 has been updated by Usaku NAKAMURA.
>
> I would like to clarify the problem.
>
> As already stated, RubyGems uses OpenSSL.
> To say strictly, RubyGems uses OpenSSL for https, signing, and its verification.
> Therefore, the option which we can take is as follows:
> (1) Maintain the present condition.
> (2) Remove OpenSSL and RubyGems together.
> (3) Prepare the alternate features of https, signing, and its verification after removing OpenSSL.
> (4) Remove the dependence to these features from RubyGems after removing OpenSSL.
> (5) Mixture of (3) and (4). That is, remove the dependence to some features from RubyGems, and prepares substitutes about another features.
>
> To my understanding, Shyouhei is taking a position on (4).
> That is, changing RubyGems to use plain http in default, and write substitutes for about signing and its verification (with GPG?).
>
> There may be also a position in which (a part of) the features which OpenSSL offers is still required as a part of Ruby, even if RubyGems sets aside.
> I understand that Fabian said that the https support itself is required.
>
> How do you think, everyone?

Can we take a less extreme approach? We should convert openssl to a gem
that ships with Ruby (like json, minitest, psych, etc). Then in case of
security issues in OpenSSL, we can just release the gem independently of
Ruby itself. Such a case has already happened with the json gem.

I've done the initial work to make openssl a gem that ships with Ruby.
The patch is here:

https://github.com/tenderlove/ruby/commit/fd96a5b1123ba1e56081ef2741a456096b4c4d12

It installs to my machine as a gem:

https://dl.dropbox.com/s/km9msdsb0uuq3mj/ruby__bash__16136_20140124_105412.png

The downside is that the openssl extension uses Ruby internals
([ruby-core:60063]), so we can't actually ship a gem until it is
decoupled from Ruby internals.

Personally, I prefer that we continue to ship with openssl. However,
even if I am in the minority, openssl must become a gem in order to
satisfy backwards compatibility requirements. I would like to continue
to download my gems over SSL, use net/http in SSL mode, use securerandom
OpenSSL, etc. :-)

--
Aaron Patterson
http://tenderlovemaking.com/

#26 - 01/25/2014 12:22 AM - mame (Yusuke Endoh)

B Kelly wrote:

> > Thus, I do not trust such "kind" words. True contributors do never say "I can contribute!", but do contribute first.
>
> Ah, but what about people who parachute into the thread, making dispassionate arguments from the sidelines, while offering no help at all?

(Sorry I'm unsure which blacklisting and whitelisting is better, but in general) I think reviewing a patch is a valuable contribution. Please continue.
Thank you!

--
Yusuke Endoh mame@tsg.ne.jp

#27 - 01/25/2014 12:32 AM - mame (Yusuke Endoh)

Aaron Patterson wrote:

> Can we take a less extreme approach? We should convert openssl to a gem
> that ships with Ruby (like json, minitest, psych, etc).

Then, who will maintain the OpenSSL gem?
Shyouhei's point is that we can no longer develop the OpenSSL extension.
Just converting it to a gem does not solve the problem at all.

--
Yusuke Endoh mame@tsg.ne.jp

#28 - 01/25/2014 02:12 AM - tenderlovemaking (Aaron Patterson)

On Sat, Jan 25, 2014 at 12:32:12AM +0000, mame@tsg.ne.jp wrote:

> Issue #9439 has been updated by Yusuke Endoh.
>
> Aaron Patterson wrote:
>
> > Can we take a less extreme approach? We should convert openssl to a gem
> > that ships with Ruby (like json, minitest, psych, etc).
>
> Then, who will maintain the OpenSSL gem?

Presumably Martin. Did he abandon it?

> Shyouhei's point is that we can no longer develop the OpenSSL extension.
> Just converting it to a gem does not solve the problem at all.

Which problem are you referring to?

--
Aaron Patterson
http://tenderlovemaking.com/

#29 - 01/25/2014 07:37 AM - shyouhei (Shyouhei Urabe)

My point is OpenSSL extension shall not be maintained by ruby-core any longer. OpenSSL itself is useful. The thing is the fact that we are maintaining that extension, is greatly prejudicing its value. It does not benefit anyone.

So gemmifying is a good step towards removal. With that, and if RubyGems can run without https, removing OpenSSL from core doesn't affect your daily life because you can easily resurrect it using gem install openssl.

#30 - 01/26/2014 10:19 PM - bascule (Tony Arcieri)

This seems ok so long as Digest is still available and RubyGems pins to particular builds of OpenSSL via a hash. That way RubyGems could download a known good OpenSSL over plaintext HTTP and check its hash to know it got the copy it's expecting.

I am sure the interim solution, were this to go forward, is for RubyGems to "trust on first use" and hope it got a good copy of OpenSSL. That is of course vulnerable to MitM attackers which would be a shame.

#31 - 01/27/2014 02:15 AM - MartinBosslet (Martin Bosslet)

Aaron Patterson wrote:

> On Sat, Jan 25, 2014 at 12:32:12AM +0000, mame@tsg.ne.jp wrote:
>
> > Issue #9439 has been updated by Yusuke Endoh.
> >
> > Aaron Patterson wrote:
> >
> > > Can we take a less extreme approach? We should convert openssl to a gem
> > > that ships with Ruby (like json, minitest, psych, etc).
> >
> > Then, who will maintain the OpenSSL gem?
>
> Presumably Martin. Did he abandon it?

No, not at all :)

> Shyouhei's point is that we can no longer develop the OpenSSL extension.
> Just converting it to a gem does not solve the problem at all.

Which problem are you referring to?

Those who know me also know that I've had my trouble with OpenSSL and I've been working towards an alternative. But right now, I think people are overreacting. Let's not jump ship just yet, especially not with any viable alternative in sight. I guess we can all agree that RubyGems with SSL is better than nothing at all, and for that alone I believe we cannot discard Ruby OpenSSL, at least today. And while #9424 is getting fairly emotional and frustrating for both sides, I say it's nothing that can't be fixed at this point.

I do like the idea of gemifying OpenSSL though (and of course, I'd also continue to maintain it), mostly because it would be easier to ship updates without requiring a full release of Ruby itself all the time. But there's one big problem that is hard to solve. I had already been discussing this with JRuby devs. If we make OpenSSL a gem outside stdlib, we immediately run into a chicken-egg problem: To be secure, we would want https gem downloads - but to get https, we need... the OpenSSL gem. Something like TUF will hopefully lead us to an alternative in the future.

Pinning RubyGems or any relying software to a fixed hash of a trusted version only goes so far, because you would have to retrieve that hash with out-of-bands means first, probably from an https site. Automating this process, e.g. with an http download of the expected hash, doesn't work because this hash could easily be MITMed without an https connection.

My feeling is that removal might be a tad too drastic, at least right now, but then, my answer is probably biased :)

#32 - 02/04/2014 09:10 PM - DanKegel (Dan Kegel)

+1 on moving OpenSSL out into a gem.

Note: Apple has marked OpenSSL obsolete, and will probably remove it in a future release of Mac OS X.

I work at a shop that ships a set of libraries to customers that make use of OpenSSL, and
although we would like to ship our own, newer, OpenSSL, we probably can't because Ruby apps that
use both our libraries and system OpenSSL would violate C's One Definition Rule
by linking in two different OpenSSLs.

In the long run, the way to minimize pain for all concerned is for all players
to try to migrate to Apple's preferred TLS API.

Therefore, I would like to see a gem added for Apple's "Secure Transport API",
and in my wildest dreams, I'd like both the OpenSSL gem and this gem to be
more or less interchangable, to ease migration between the two APIs.

And I'd like a pony.

#33 - 02/25/2014 11:38 PM - docwhat (Christian Höltje)

I like Aaron Patterson (TenderLove's) suggestion in #25: Split it into a gem and include the gem in ruby-core by default (like JSON).

It would solve the chicken-and-the-egg problem, keep rubygems happy, and allow upgrading.

At some point in the future, we could switch to TLS or the new security hotness to include in ruby without breaking anything and people who wanted last weeks security api can just gem install it.

#34 - 03/08/2014 02:26 AM - zzak (zzak _)

- Status changed from Open to Rejected

Closing in favor of #9612