Ruby - Feature #6503

Support for the NPN extension to TLS/SSL

05/27/2012 02:57 PM - igrigorik (Ilya Grigorik)

Status:	Closed	
Priority:	Normal	
Assignee:	MartinBosslet (Martin Bosslet)	
Target version:	2.0.0	

Description

OpenSSL 1.0.1+ added support for Next Protocol Negotiation (NPN) extensions. A couple of relevant links:

- Google technical note: https://technotes.googlecode.com/git/nextprotoneg.html
- IETF draft: http://tools.ietf.org/html/draft-agl-tls-nextprotoneg-02

NPN allows the client to negotiate the session protocol as part of the TLS handshake (ex, "http 1.1", or "spdy/v{1,2,3}"). To support SPDY we need NPN support within OpenSSL core in Ruby. The API is already implemented in OpenSSL 1.0.1+, so it's a matter of adding support in Ruby core.

Sister bug for Python 3.3: http://bugs.python.org/issue14204

Associated revisions

Revision 25e6db3e3cb52c1a81e1e4a958a8d520a996812e - 08/31/2012 09:47 AM - MartinBosslet (Martin Bosslet)

- ext/openssl/extconf.rb: Check existence of OPENSSL_NPN_NEGOTIATED.
  ext/ossl_ssl.c: Support Next Protocol Negotiation. Protocols to be
  advertised by the server can be set in the SSLContext by using
  SSLContext#npn_protocols=, protocol selection on the client is
  supported by providing a selection callback with
  SSLContext#npn_select_cb. The protocol that was finally negotiated
  is available through SSL#npn_protocol.
  test/openssl/test_ssl.rb: Add tests for Next Protocol Negotiation.
  NEWS: add news about NPN support.
  [Feature #6503] [ruby-core:45272]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@36871 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 25e6db3e - 08/31/2012 09:47 AM - MartinBosslet (Martin Bosslet)

- ext/openssl/extconf.rb: Check existence of OPENSSL_NPN_NEGOTIATED.
  ext/ossl_ssl.c: Support Next Protocol Negotiation. Protocols to be
  advertised by the server can be set in the SSLContext by using
  SSLContext#npn_protocols=, protocol selection on the client is
  supported by providing a selection callback with
  SSLContext#npn_select_cb. The protocol that was finally negotiated
  is available through SSL#npn_protocol.
  test/openssl/test_ssl.rb: Add tests for Next Protocol Negotiation.
  NEWS: add news about NPN support.
  [Feature #6503] [ruby-core:45272]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@36871 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 05/27/2012 07:25 PM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

- Assignee set to MartinBosslet (Martin Bosslet)

Thank you Ilya!

Martin, could you tell me how hard is it to implement this?

--
Yusuke Endoh mame@tsg.ne.jp

#2 - 06/06/2012 07:32 AM - davidbalbert (David Albert)

If nobody has claimed this yet, I'm happy to take a crack at it over the next couple of days. I know the guy who wrote the Python patch and have a decent understanding of what went into it. It should not be a tremendous amount of work (famous last words). If there aren't any objections, I'll send a first pass at a patch soon.

#3 - 06/06/2012 08:31 AM - MartinBosslet (Martin Bosslet)

Thanks, Ilya, for the links!

@mame (Yusuke Endoh): I just checked the Python patch and what OpenSSL already provides and what would be needed on our side. It's really not too much, basically feeding OpenSSL API with parameters that we could make a part of SSL context objects. One thing that worries me though is that we have nothing to really test this.

@Ilya: Would you have any ideas what we could do? The RFC is still in draft status, and I've followed the conversation in [1]. Can 13172 and 67 be taken for granted? :)

@David: It's OK, I'll take this, but thanks for your support!

[1] http://www.ietf.org/mail-archive/web/tls/current/msg08605.html

#4 - 07/06/2012 03:23 AM - igrigorik (Ilya Grigorik)

Hey guys, apologies about the wait.

@Martin: I don't follow the IANA politics, but for what its worth, I would consider it stable at this point. The support is there in OpenSSL, we have 50% of the browser market share using it to negotiate SPDY (Chrome + FF), and we have commercial vendors like F5, Akamai, and others supporting it.. :-)

Also, just realized that I linked to wrong version earlier: http://tools.ietf.org/html/draft-agl-tls-nextprotoneg-04

As far as testing, this is a bit of a chicken and egg problem. There are no pure Ruby libraries that you can run this against.. For an integration test, you could try performing a hadshake against a https://google.com server and test the TLS upgrade. I do have a pure Ruby spdy gem, but it needs a few updates (NPN support is the missing link, really): http://github.com/igrigorik/spdy

Let me know how/if I can help.

#5 - 07/06/2012 03:53 PM - duerst (Martin Dürst)

On 2012/07/06 3:23, igrigorik (Ilya Grigorik) wrote:

> Issue #6503 has been updated by igrigorik (Ilya Grigorik).
>
> Hey guys, apologies about the wait.
>
> @Martin: I don't follow the IANA politics,

Just a small detail: That should be IETF politics, I guess. But I'm also
not familiar with that corner of the IETF, sorry.

Regards, Martin.

> but for what its worth, I would consider it stable at this point. The support is there in OpenSSL, we have 50% of the browser market share using it to negotiate SPDY (Chrome + FF), and we have commercial vendors like F5, Akamai, and others supporting it.. :-)
>
> Also, just realized that I linked to wrong version earlier: http://tools.ietf.org/html/draft-agl-tls-nextprotoneg-04
>
> As far as testing, this is a bit of a chicken and egg problem. There are no pure Ruby libraries that you can run this against.. For an integration test, you could try performing a hadshake against a https://google.com server and test the TLS upgrade. I do have a pure Ruby spdy gem, but it needs a few updates (NPN support is the missing link, really): http://github.com/igrigorik/spdy
>
> ## Let me know how/if I can help.
>
> Feature #6503: Support for the NPN extension to TLS/SSL
> https://bugs.ruby-lang.org/issues/6503#change-27833
>
> Author: igrigorik (Ilya Grigorik)
> Status: Assigned
> Priority: Normal
> Assignee: MartinBosslet (Martin Bosslet)
> Category:
> Target version:
>
> OpenSSL 1.0.1+ added support for Next Protocol Negotiation (NPN) extensions. A couple of relevant links:

- Google technical note: https://technotes.googlecode.com/git/nextprotoneg.html
- IETF draft: http://tools.ietf.org/html/draft-agl-tls-nextprotoneg-02

NPN allows the client to negotiate the session protocol as part of the TLS handshake (ex, "http 1.1", or "spdy/v{1,2,3}"). To support SPDY we need NPN support within OpenSSL core in Ruby. The API is already implemented in OpenSSL 1.0.1+, so it's a matter of adding support in Ruby core.

Sister bug for Python 3.3: http://bugs.python.org/issue14204

#6 - 07/06/2012 04:03 PM - MartinBosslet (Martin Bosslet)

- Category set to ext

- Target version set to 2.0.0

> On 2012/07/06 3:23, igrigorik (Ilya Grigorik) wrote:
>
> Issue #6503 has been updated by igrigorik (Ilya Grigorik).
>
> Hey guys, apologies about the wait.

No problem :)

> @Martin: I don't follow the IANA politics,
>
> but for what its worth, I would consider it stable at this point. The support is there in OpenSSL, we have 50% of the browser market share using it to negotiate SPDY (Chrome + FF), and we have commercial vendors like F5, Akamai, and others supporting it.. :-)

Yes, and to be honest, I'm also in favor of the technology, just wanted to make sure that it's stable enough. But from what I saw, we could handle most of it transparently, OpenSSL does the heavy lifting - so even if there were major changes, they should only affect OpenSSL itself, but hopefully not the API exposing the feature.

> Also, just realized that I linked to wrong version earlier: http://tools.ietf.org/html/draft-agl-tls-nextprotoneg-04

OK, thanks for the hint!

> As far as testing, this is a bit of a chicken and egg problem. There are no pure Ruby libraries that you can run this against.. For an integration test, you could try performing a hadshake against a https://google.com server and test the TLS upgrade. I do have a pure Ruby spdy gem, but it needs a few updates (NPN support is the missing link, really): http://github.com/igrigorik/spdy

True. I also thought of directly testing against https://google.com, it's a fairly stable server ;) But I was wondering how internal policies are, is it sound to test against external URLs? Could some of the other devs please comment on this?

> Let me know how/if I can help.

Will do, thanks for your help! If nobody has major reservations, I would add support soon.

-Martin

#7 - 07/06/2012 04:08 PM - MartinBosslet (Martin Bosslet)

duerst (Martin Dürst) wrote:

> Just a small detail: That should be IETF politics, I guess. But I'm also
> not familiar with that corner of the IETF, sorry.

Just out of curiosity - because IETF is in charge of the TLS extension registry?
That's what I think I understood from [1] at least:

> TLS ExtensionType Registry: Future values are allocated via IETF Consensus

[1] http://tools.ietf.org/html/rfc5246#section-12

#8 - 07/10/2012 03:53 PM - duerst (Martin Dürst)

On 2012/07/06 16:10, MartinBosslet (Martin Bosslet) wrote:

> Issue #6503 has been updated by MartinBosslet (Martin Bosslet).
>
> duerst (Martin Dürst) wrote:
>
> > Just a small detail: That should be IETF politics, I guess. But I'm also
> > not familiar with that corner of the IETF, sorry.
>
> Just out of curiosity - because IETF is in charge of the TLS extension registry?
> That's what I think I understood from [1] at least:
>
> > TLS ExtensionType Registry: Future values are allocated via IETF Consensus

Yes. More generally, IANA is only a clerical office function.

Regards, Martin.

> **[1]** http://tools.ietf.org/html/rfc5246#section-12
>
> Feature #6503: Support for the NPN extension to TLS/SSL
> https://bugs.ruby-lang.org/issues/6503#change-27850
>
> Author: igrigorik (Ilya Grigorik)
> Status: Assigned
> Priority: Normal
> Assignee: MartinBosslet (Martin Bosslet)
> Category: ext
> Target version: 2.0.0
>
> OpenSSL 1.0.1+ added support for Next Protocol Negotiation (NPN) extensions. A couple of relevant links:
>
> - Google technical note: https://technotes.googlecode.com/git/nextprotoneg.html
> - IETF draft: http://tools.ietf.org/html/draft-agl-tls-nextprotoneg-02
>
> NPN allows the client to negotiate the session protocol as part of the TLS handshake (ex, "http 1.1", or "spdy/v{1,2,3}"). To support SPDY we need NPN support within OpenSSL core in Ruby. The API is already implemented in OpenSSL 1.0.1+, so it's a matter of adding support in Ruby core.
>
> Sister bug for Python 3.3: http://bugs.python.org/issue14204

#9 - 07/28/2012 01:37 PM - igrigorik (Ilya Grigorik)

Martin, let me know if you run into any questions or issues.. would love to see this working, sooner rather later. :-)

#10 - 08/01/2012 11:44 PM - MartinBosslet (Martin Bosslet)

igrigorik (Ilya Grigorik) wrote:

> Martin, let me know if you run into any questions or issues.. would love to see this working, sooner rather later. :-)

Thanks for the offer, I'll get back to you if I run into trouble :) I'll try to implement it for the next 1.9.3 patch release.

#11 - 08/31/2012 06:47 PM - Anonymous

- Status changed from Assigned to Closed

- % Done changed from 0 to 100

This issue was solved with changeset r36871.
Ilya, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- ext/openssl/extconf.rb: Check existence of OPENSSL_NPN_NEGOTIATED.
  ext/ossl_ssl.c: Support Next Protocol Negotiation. Protocols to be

advertised by the server can be set in the SSLContext by using
SSLContext#npn_protocols=, protocol selection on the client is
supported by providing a selection callback with
SSLContext#npn_select_cb. The protocol that was finally negotiated
is available through SSL#npn_protocol.
test/openssl/test_ssl.rb: Add tests for Next Protocol Negotiation.
NEWS: add news about NPN support.
[Feature #6503] [ruby-core:45272]

#12 - 08/31/2012 06:56 PM - MartinBosslet (Martin Bosslet)

Protocols to be advertised by the server can now be set like this:

ctx = ... # some OpenSSL::SSL::SSLContext
ctx.npn_protocols = ["spdy/3", "spdy/2", "http/1.1"]

Selection on the client is handled via callback:

ctx = ... # some OpenSSL::SSL::SSLContext
ctx.npn_select_cb = lambda do |protocols|

selection logic, return value must be the selected protocol

protocols.first
end

Raising or causing an error during the callback will effectively terminate the handshake.
The protocol that was finally chosen can be inspected on the resulting SSL instance with
SSL#npn_protocol. By default, not setting SSLContext#npn_protocols or SSLContext#npn_select_cb
will have the effect that NPN extension support is disabled.

@Ilya: Although I could write tests to assert the correctness of the basic behavior, I haven't
tried it in a real life scenario yet. Could you please confirm that this is working for you?