Ruby - Bug #3150

net/https peer verification doesn't do anything

04/15/2010 06:39 PM - hongli (Hongli Lai)

Status:	Third Party's Issue		
Priority:	Normal		
Assignee:	nahi (Hiroshi Nakamura)		
Target version:	1.9.3		
ruby -v:	ruby 1.8.7 (2009-06-08 patchlevel 173) [universal-darwin10.0]	**Backport:**	

Description

=begin
Setting verify_mode to VERIFY_PEER should make net/https raise an exception if peer certificate verification fails. For example:

```
require 'net/http'
require 'net/https'
require 'openssl'

url = URI.parse("https://bugzilla.redhat.com/")
http = Net::HTTP.new(url.host, url.port)
http.use_ssl = true
http.verify_mode = OpenSSL::SSL::VERIFY_PEER

http.verify_callback = proc do |preverify_ok, ssl_context|
  puts "verification succeeded: #{preverify_ok}"
end

request = Net::HTTP::Get.new(url.path)
response = http.request(request)   # Should raise error
```

Expected output:

```
$ ruby ssltest.rb
verification succeeded: false
.../lib/ruby/1.8/net/http.rb:586:in `connect': certificate verify failed (OpenSSL::SSL::SSLError)
```

Actual output:

```
$ ruby ssltest.rb
verification succeeded: false
(no exception raised)
```

Either net/https is broken, or OpenSSL is broken, or OpenSSL changed some default behavior.
I can reproduce the problem OS X Snow Leopard with OpenSSL 0.9.8k and the following Ruby versions:

- ruby 1.8.6 (2010-02-05 patchlevel 399) [i686-darwin10.3.0]
- ruby 1.8.7 (2009-06-08 patchlevel 173) [universal-darwin10.0]
- ruby 1.9.1p376 (2009-12-07 revision 26041) [i386-darwin10.2.0]
- ruby 1.9.2dev (2010-04-09 trunk 27271) [x86_64-darwin10.3.0]

The problem does not occur on Debian Linux 5 with OpenSSL 0.9.8g and the following Ruby versions:

- ruby 1.8.6 (2008-08-11 patchlevel 287) [i686-linux]

I don't know whether 1.8.6-p287 exhibits the problem on Snow Leopard, it fails to compile with the following errors:

gcc -I. -I../.. -I../../. -I../../../ext/openssl -DRUBY_EXTCONF_H="extconf.h" -fno-common -g -O2 -pipe -fno-common -c openssl_missing.c
In file included from openssl_missing.c:22:
openssl_missing.h:123: error: conflicting types for 'BN_rand_range'
/usr/include/openssl/bn.h:411: error: previous declaration of 'BN_rand_range' was here
openssl_missing.h:124: error: conflicting types for 'BN_pseudo_rand_range'

/usr/include/openssl/bn.h:412: error: previous declaration of 'BN_pseudo_rand_range' was here
=end

History

#1 - 08/20/2010 07:10 AM - nahi (Hiroshi Nakamura)

- Assignee set to nahi (Hiroshi Nakamura)

=begin
Anyone can reproduce this? On Snow Leopard only? It must be critical issue if it still exists.
=end

#2 - 08/20/2010 03:11 PM - tenderlovemaking (Aaron Patterson)

=begin
On Fri, Aug 20, 2010 at 07:12:47AM +0900, Hiroshi NAKAMURA wrote:

> Issue #3150 has been updated by Hiroshi NAKAMURA.
>
> Assigned to set to Hiroshi NAKAMURA
>
> Anyone can reproduce this? On Snow Leopard only? It must be critical issue if it still exists.

I am able to reproduce this on Snow Leopard. I will try to debug and provide more
info soon!

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

Attachment: (unnamed)
=end

#3 - 09/06/2010 11:25 AM - nahi (Hiroshi Nakamura)

=begin
Thanks Aaron. Hope we can identify the cause soon.
=end

#4 - 09/09/2010 03:42 AM - tenderlovemaking (Aaron Patterson)

=begin
On Mon, Sep 06, 2010 at 11:26:01AM +0900, Hiroshi NAKAMURA wrote:

> Issue #3150 has been updated by Hiroshi NAKAMURA.
>
> Thanks Aaron. Hope we can identify the cause soon.

Hi!

I *think* I have good news. I am starting to suspect this is not a bug
of Ruby.

First (for posterity), here is a chunk of Ruby code to reproduce the
problem without using net/http:

```
require 'socket'
require 'openssl'

s   = TCPSocket.new 'bugzilla.redhat.com', 443
ctx = OpenSSL::SSL::SSLContext.new
ctx.verify_mode = OpenSSL::SSL::VERIFY_PEER
s   = OpenSSL::SSL::SSLSocket.new s, ctx
s.connect
```

This code raises an exception on my linux machine, but not on my Snow
Leopard machine.

I wrote a (mostly) equivalent program in C here:

http://gist.github.com/570593

On my Snow Leopard machine, SSL_connect() returns a value greater than
0. On my linux machine, this program results in an error. This program
leads me to believe this is not a bug of Ruby, but I don't know why the
behavior is different on linux vs OS X. I am still researching.

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

Attachment: (unnamed)
=end

#5 - 09/14/2010 04:01 PM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Assigned

=begin

=end

#6 - 09/14/2010 05:20 PM - justincase (Justin Case)

=begin
Workaround with rvm:

$ openssl version
OpenSSL 0.9.8l 5 Nov 2009

$ rvm package install openssl
$ rvm remove 1.9.2
$ rvm install 1.9.2 -C --with-openssl-dir=$HOME/.rvm/usr

$ $HOME/.rvm/usr/bin/openssl version
OpenSSL 0.9.8n 24 Mar 2010

Output:

$ ruby ssltest.rb
verification succeeded: false
[...] http.rb:677:in `connect': SSL_connect returned=1 errno=0 state=SSLv3 read server certificate B: certificate verify failed
(OpenSSL::SSL::SSLError)
=end

#7 - 11/25/2010 08:48 PM - nahi (Hiroshi Nakamura)

=begin
I found that the OS X 10.6.5 includes security fix for OpenSSL. http://support.apple.com/kb/HT4435
And it looks similar to this issue for me. Can someone check if the behavior of above scripts changed on 10.6.5?

The ticket I file at developer.apple.com is not yet updated so it could be a different fix.
=end

#8 - 11/27/2010 03:16 AM - tenderlovemaking (Aaron Patterson)

=begin
On Thu, Nov 25, 2010 at 08:48:59PM +0900, Hiroshi NAKAMURA wrote:

> Issue #3150 has been updated by Hiroshi NAKAMURA.
>
> I found that the OS X 10.6.5 includes security fix for OpenSSL. http://support.apple.com/kb/HT4435
> And it looks similar to this issue for me. Can someone check if the behavior of above scripts changed on 10.6.5?
>
> The ticket I file at developer.apple.com is not yet updated so it could be a different fix.

I've updated to 10.6.5, and nothing has changed with this. :-(

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

Attachment: (unnamed)
=end

#9 - 11/27/2010 07:51 AM - nahi (Hiroshi Nakamura)

=begin
Thank you for sharing your time. It's bad to know that the problem still exists...
=end

#10 - 01/08/2011 02:59 AM - nbibler (Nathaniel Bibler)

=begin
I tested this again in OS X 10.6.6 yesterday, and the problem still exists. I used Aaron's code from 09/09/2010 03:42 AM and no exceptions were raised.

ruby 1.9.2p0 (2010-08-18 revision 29036) [x86_64-darwin10.4.0]
=> #OpenSSL::SSL::SSLSocket:0x000001008470a0

ruby 1.8.7 (2009-12-24 patchlevel 248) [i686-darwin10.2.0]
=> #OpenSSL::SSL::SSLSocket:0x1005b3be0
=end

#11 - 01/09/2011 11:44 PM - MartinBosslet (Martin Bosslet)

=begin
Using Ubuntu 10.04 and OpenSSL 1.0.0c, I get the expected exception.
=end

#12 - 03/06/2011 07:20 AM - nahi (Hiroshi Nakamura)

=begin
http://www.afp548.com/article.php?story=20091007164413755 (Thanks for finding this, Eric: https://twitter.com/#!/drbrain/status/43788281007194112
)

Apple looks to ship customized OpenSSL which integrates trusted root certificates from OS's system keychain. So we should test against not https://bugs.redhat.com/ but another SSL site which is using the certificate OS does not trust.

Hongli, can you confirm that the result of your program affected by OS's keychain setting?
=end

#13 - 03/16/2011 09:57 AM - justincase (Justin Case)

=begin
Heh. Actually, without the ((|verify_callback|)) I get consistent results.
net/https does a (({post_connection_check})) internally when (({use_ssl=true})).
(({VERIFY_PEER})) is also automatically set when (({use_ssl=true})) using
1.9.2-p180. Didn't check when that was introduced but on Ruby 1.8.7-p174
you have to explicitly set (({VERIFY_PEER})).

```
require 'net/http'
require 'net/https'
require 'uri'

url = URI.parse("https://github.com/")      # OK
#url = URI.parse("https://etutorials.org/")  # Hostname mismatch
http = Net::HTTP.new(url.host, url.port)
http.use_ssl = true
http.verify_mode = OpenSSL::SSL::VERIFY_PEER
request = Net::HTTP::Get.new(url.path)
response = http.request(request)
```

Result on both Mac OS 10.6.6 and Ubuntu 10.04.1 LTS:

```
=> OpenSSL::SSL::SSLError: hostname was not match with the server
certificate
```

((<Gist 871758|URL:https://gist.github.com/871758>))

Delving deeper, here's what I found. While I haven't found the root
cause, I think this might be interesting.

From the OpenSSL docs:

The verify_callback function is used to control the behaviour when the
SSL_VERIFY_PEER flag is set. It must be supplied by the
application and receives two arguments: preverify_ok indicates,
whether the verification of the certificate in question was passed
(preverify_ok=1) or not (preverify_ok=0).

[...]

The certificate chain is checked starting with the deepest nesting
level (the root CA certificate) and worked upward to the peer's
certificate. At each level signatures and issuer attributes are checked.
Whenever a verification error is found, the error number is stored in
x509_ctx and verify_callback is called with preverify_ok=0

[...]

If no error is found for a certificate, verify_callback is called with
preverify_ok=1 before advancing to the next level.

[...]

The return value of verify_callback controls the strategy of the
further verification process. If verify_callback returns 0, the
verification process is immediately stopped with ``verification failed''
state. If SSL_VERIFY_PEER is set, a verification failure alert is sent
to the peer and the TLS/SSL handshake is terminated. If verify_callback
returns 1, the verification process is continued. If verify_callback
always returns 1, the TLS/SSL handshake will not be terminated with
respect to verification failures and the connection will be established.
The calling process can however retrieve the error code of the last
verification error using SSL_get_verify_result(3) or by maintaining its
own error storage managed by verify_callback.

Ok. So it's pretty darn important what ((|verify_callback|)) actually
returns. Which in Hongli's test case is ((*nil*))! Let's revise
((|verify_callback|)) so that it returns ((|preverify_ok|)):

((<Gist 871668|URL:https://gist.github.com/871668>))

Now I get the same end results as before. Note that I say end results
because the validation process differs.

Per documentation ((|verify_callback|)) is called for each certificate
in the chain. This seems to happen successfully on Ubuntu but
not on OS X.

Running the gist against ((*github.com*)):

On Mac OS X 10.6.6 we get:

```
preverify_ok: false
error: 19
=> #<Net::HTTPOK 200 OK readbody=true>
```

On Ubuntu 10.04.1 LTS:

```
preverify_ok: true
error: 0
preverify_ok: true
error: 0
preverify_ok: true
error: 0
preverify_ok: true
error: 0
=> #<Net::HTTPOK 200 OK readbody=true>
```

The error on OS X means "(({X509_V_ERR_SELF_SIGNED_CERT_IN_CHAIN})):
((*self signed certificate in certificate chain*))" which is obviously
not the case since the certificate is valid.

Github's certificate chain is 4 deep and on Ubuntu no errors are found
while traversing the chain. On OS X the first certificate in the chain
couldn't be validated and stops but still makes a connection.

If we now run it against ((*etutorials.org*)) which has a hostname mismatch:

On Mac OS X 10.6.6 we get:

```
preverify_ok: false
error: 20
OpenSSL::SSL::SSLError: hostname was not match with the server certificate
```

On Ubuntu 10.04.1 LTS:

preverify_ok: true
error: 0
preverify_ok: true
error: 0
OpenSSL::SSL::SSLError: hostname was not match with the server certificate

Again we see that the certificate chain is successfully traversed on Ubuntu. AFAIK it doesn't throw an error code inside the ((|verify_callback|)) because the certificate itself is valid in the chain.

On OS X the first certificate in the chain couldn't be validated and stops but this time an exception is thrown! Hooray.

(The error on OS X means "(({X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT_LOCALLY})): ((*unable to get local issuer certificate*))" i.e. that it can't find a trusted root CA.)

The important bit here is that the connection failed and that we can trust the outcome even though the validation process doesn't seem to work the same.

((%$openssl version -d%)) will give you the path where OpenSSL looks for certs. On 10.6.6 thats /System/Library/OpenSSL/ which is empty. Wild guess: OpenSSL on OS X looks for certs in /System/Library/OpenSSL/, doesn't find any and fails over to the keychain. Or something to that effect.

You can try https://www.debian-administration.org/ for testing a self-signed certificate.

System:

Mac OS 10.6.6 / Build 10J567
OpenSSL 0.9.8l 5 Nov 2009

Ubuntu 10.04.1 LTS
OpenSSL 0.9.8k 25 Mar 2009

Also a good read ((<Braintree: SSL socket verify_mode doesn't verify|URL: http://www.braintreepaymentsolutions.com/devblog/sslsocket-verify_mode-doesnt-verify>))
=end

#14 - 06/26/2011 02:09 PM - akr (Akira Tanaka)

- Project changed from 8 to Ruby

#15 - 06/26/2011 04:48 PM - nahi (Hiroshi Nakamura)

- Target version set to 1.9.3

#16 - 06/29/2011 07:53 PM - nahi (Hiroshi Nakamura)

- Category set to ext

- Status changed from Assigned to Third Party's Issue

- Priority changed from 5 to Normal

Finally I found that Apple ships patched version of OpenSSL.

http://www.opensource.apple.com/source/OpenSSL098/OpenSSL098-27/src/crypto/x509/x509_vfy_apple.h

```
/*
* X509_verify_cert
*
* Originally located in x509_vfy.c.
*
* Verify certificate with OpenSSL created X509_verify_cert. If and only if
* OpenSSL cannot get certificate issuer locally then OS X security API will
* verify the certificate, using TrustEvaluationAgent.
*
* Return values:
* --------------
* -1: Null was passed for either ctx or ctx->cert.
```

* 0: Certificate is trusted.
* 1: Certificate is not trusted.
*/
int X509_verify_cert(X509_STORE_CTX *ctx);

So, with the OpenSSL on Show Leopard, a certificate is trusted if the certificate is trusted by system even if you don't set proper SSL_CERT_DIR.

You can see the original report have a verify callback and it reports 'false'. Here's what x509_apply_vfy.c is doing:

http://www.opensource.apple.com/source/OpenSSL098/OpenSSL098-27/src/crypto/x509/x509_vfy_apple.c
/* Try OpenSSL, if we get a local certificate issue verify against trusted roots */
ret = X509_verify_cert_orig(ctx);
if (ret != 1 && (ctx->error & X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT_LOCALLY)) {
...

So even though you return false from verify_callback, it could be trusted by system if the reason is
X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT_LOCALLY.

I close this as 'Third Party's Issue.' Yes, it's an *issue* of Snow Leopard.

Please let me know if I am wrong. I'll reopen this.