Ruby master - Bug #4237

SSL_shutdown should be called until it returns 0

01/06/2011 10:26 AM - tenderlovemaking (Aaron Patterson)

Status:	Closed		
Priority:	Normal		
Assignee:	nahi (Hiroshi Nakamura)		
Target version:	1.9.3		
ruby -v:	-	**Backport:**	

Description

=begin
The documentation says SSL_shutdown should be called until it returns 0. I believe this was causing heap corruption that can be demonstrated through the steps listed here:

http://intertwingly.net/blog/2010/12/07/SQLite3-CorruptException-database-disk-image-is-malformed

make sure you use Ruby r30293 or earlier. I believe r30294 only fixed a symptom is it closes the SSL socket twice (once from the call to close and once again during GC).

Please note that Sam sees the problem because postfix is establishing a TLS connection.
=end

Related issues:

Related to Ruby master - Bug #4241: IMAPTest#test_imaps_verify_none does not ...	**Closed**	**01/06/2011**

Associated revisions

Revision 369b0950 - 06/24/2011 07:01 AM - nahi (Hiroshi Nakamura)

- ext/openssl/ossl_ssl.c (ossl_ssl_shutdown): Try to shutdown SSL connection more gracefully. Call SSL_shutdown() max 4 times until it returns 1 (success). Bi-directional SSL close has several states but SSL_shutdown() kicks only 1 transition per call. Max 4 is from mod_ssl.c of Apache httpd that says 'max 2x pending * 2x data = 4'. See #4237.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@32219 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 32219 - 06/24/2011 07:01 AM - nahi (Hiroshi Nakamura)

- ext/openssl/ossl_ssl.c (ossl_ssl_shutdown): Try to shutdown SSL connection more gracefully. Call SSL_shutdown() max 4 times until it returns 1 (success). Bi-directional SSL close has several states but SSL_shutdown() kicks only 1 transition per call. Max 4 is from mod_ssl.c of Apache httpd that says 'max 2x pending * 2x data = 4'. See #4237.

Revision 32219 - 06/24/2011 07:01 AM - nahi (Hiroshi Nakamura)

- ext/openssl/ossl_ssl.c (ossl_ssl_shutdown): Try to shutdown SSL connection more gracefully. Call SSL_shutdown() max 4 times until it returns 1 (success). Bi-directional SSL close has several states but SSL_shutdown() kicks only 1 transition per call. Max 4 is from mod_ssl.c of Apache httpd that says 'max 2x pending * 2x data = 4'. See #4237.

Revision 32219 - 06/24/2011 07:01 AM - nahi (Hiroshi Nakamura)

- ext/openssl/ossl_ssl.c (ossl_ssl_shutdown): Try to shutdown SSL connection more gracefully. Call SSL_shutdown() max 4 times until it returns 1 (success). Bi-directional SSL close has several states but SSL_shutdown() kicks only 1 transition per call. Max 4 is from mod_ssl.c of Apache httpd that says 'max 2x pending * 2x data = 4'. See #4237.

Revision 32219 - 06/24/2011 07:01 AM - nahi (Hiroshi Nakamura)

- ext/openssl/ossl_ssl.c (ossl_ssl_shutdown): Try to shutdown SSL connection more gracefully. Call SSL_shutdown() max 4 times until it returns 1 (success). Bi-directional SSL close has several states but SSL_shutdown() kicks only 1 transition per call. Max 4 is from mod_ssl.c of Apache httpd that says 'max 2x pending * 2x data = 4'. See #4237.

Revision 32219 - 06/24/2011 07:01 AM - nahi (Hiroshi Nakamura)

- ext/openssl/ossl_ssl.c (ossl_ssl_shutdown): Try to shutdown SSL connection more gracefully. Call SSL_shutdown() max 4 times until it

returns 1 (success). Bi-directional SSL close has several states but SSL_shutdown() kicks only 1 transition per call. Max 4 is from mod_ssl.c of Apache httpd that says 'max 2x pending * 2x data = 4'. See #4237.

Revision 32219 - 06/24/2011 07:01 AM - nahi (Hiroshi Nakamura)

- ext/openssl/ossl_ssl.c (ossl_ssl_shutdown): Try to shutdown SSL connection more gracefully. Call SSL_shutdown() max 4 times until it returns 1 (success). Bi-directional SSL close has several states but SSL_shutdown() kicks only 1 transition per call. Max 4 is from mod_ssl.c of Apache httpd that says 'max 2x pending * 2x data = 4'. See #4237.

History

#1 - 01/06/2011 10:39 AM - nahi (Hiroshi Nakamura)

=begin
Here's a link to OpenSSL doc Aaron said: http://www.openssl.org/docs/ssl/SSL_shutdown.html

You should read it directly, since it's not simple as "SSL_shutdown should be called until it returns 0". We should handle -1 as well, if we want to ensure bi-directional SSL close. Plus we should check OpenSSL versions behavior.

See also: https://www.honeynet.org/node/436
=end

#2 - 01/06/2011 10:51 AM - tenderlovemaking (Aaron Patterson)

=begin
On Thu, Jan 06, 2011 at 10:39:11AM +0900, Hiroshi NAKAMURA wrote:

> Issue #4237 has been updated by Hiroshi NAKAMURA.
>
> Here's a link to OpenSSL doc Aaron said: http://www.openssl.org/docs/ssl/SSL_shutdown.html
>
> You should read it directly, since it's not simple as "SSL_shutdown should be called until it returns 0". We should handle -1 as well, if we want to ensure bi-directional SSL close. Plus we should check OpenSSL versions behavior.

This happens during GC. What should we do during GC if it returns -1?

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

Attachment: (unnamed)
=end

#3 - 03/26/2011 10:25 PM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Assigned

#4 - 06/22/2011 11:08 PM - nahi (Hiroshi Nakamura)

Aaron, I'm sorry for late response, but can you point the original report of this issue? The link ' http://intertwingly.net/blog/2010/12/07/SQLite3-CorruptException-database-disk-image-is-malformed' looks not related to openssl. Am I misunderstanding something?

#5 - 06/23/2011 12:29 AM - tenderlovemaking (Aaron Patterson)

- ruby -v changed from ruby 1.9.3dev (2011-01-06 trunk 30471) [x86_64-darwin10.5.0] to -

On Wed, Jun 22, 2011 at 11:08:08PM +0900, Hiroshi NAKAMURA wrote:

> Issue #4237 has been updated by Hiroshi NAKAMURA.
>
> Aaron, I'm sorry for late response, but can you point the original report of this issue? The link ' http://intertwingly.net/blog/2010/12/07/SQLite3-CorruptException-database-disk-image-is-malformed' looks not related to openssl. Am I misunderstanding something?

Sam's report does not look like it's related to openssl. The case that he was encountering was an SSL socket failure. Please look at the change I committed: r30294.

Before my change, the post_connection_check in tlsconnect would fail, and the ssl socket would never be closed. I believe this was causing heap corruption which lead to the issues that Sam was seeing.

After studying the openssl code, I saw that we only call SSL_shutdown once[1] and we do not check the return value[2]. This seems like a problem as the OpenSSL documentation says[3]:

It is therefore recommended, to check the return value of SSL_shutdown() and call SSL_shutdown() again, if the bidirectional shutdown is not yet complete (return value of the first call is 0). As the shutdown is not specially handled in the SSLv2 protocol, SSL_shutdown() will succeed on the first call.

Like I said in the original report:

I believe r30294 only fixed a symptom as it closes the SSL socket twice (once from the call to close and once again during GC).

I believe we need to be checking the return value of SSL_shutdown from ossl_ssl_shutdown, and possibly calling it multiple times. I made that change in r30451, but it was later reverted because we were seeing segvs on the CI machine.

I'll try to come up with a test case to reproduce, but it seems very difficult to me. :-(

1. https://github.com/ruby/ruby/blob/trunk/ext/openssl/ossl_ssl.c#L970-977
2. https://github.com/ruby/ruby/blob/trunk/ext/openssl/ossl_ssl.c#L974
3. http://www.openssl.org/docs/ssl/SSL_shutdown.html

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

#6 - 06/23/2011 02:27 PM - nahi (Hiroshi Nakamura)

Thanks for the explanation. I think r30294 (fix for lib/net/smtp.rb) is good. It solved the original problem, right? Of course, ext/openssl should try to shutdown the connection gracefully to the extent possible though.

And the second part, can you remember some more details of SEGV you got? If the error you got is timeout or something, it could be caused by infinite loop by r30451. Is this what you're saying?
http://www.rubyist.net/~akr/chkbuild/debian/ruby-trunk/log/20110105T190101Z.log.html.gz

As I wrote above, bi-directional SSL close is hard to control (opposite side is not controllable) so calling SSL_shutdown() several times without checking could be enough.

Here's what mod_ssl is doing.

int SSL_smart_shutdown(SSL *ssl)
{

int i;
int rc;

```
/*
 * Repeat the calls, because SSL_shutdown internally dispatches through a
 * little state machine. Usually only one or two interation should be
 * needed, so we restrict the total number of restrictions in order to
 * avoid process hangs in case the client played bad with the socket
 * connection and OpenSSL cannot recognize it.
 */
rc = 0;
for (i = 0; i < 4 /* max 2x pending + 2x data = 4 */; i++) {
    if ((rc = SSL_shutdown(ssl)))
        break;
}
return rc;
```

}

If my guess about the error you got is correct, I'll commit similar changes for SSL_shutdown().

#7 - 06/24/2011 02:23 AM - tenderlovemaking (Aaron Patterson)

On Thu, Jun 23, 2011 at 02:27:41PM +0900, Hiroshi NAKAMURA wrote:

> Issue #4237 has been updated by Hiroshi NAKAMURA.

> Thanks for the explanation. I think r30294 (fix for lib/net/smtp.rb) is good. It solved the original problem, right? Of course, ext/openssl should try to shutdown the connection gracefully to the extent possible though.

No problem. :-)

Yes it did solve the original problem, so I am happy.

> And the second part, can you remember some more details of SEGV you got? If the error you got is timeout or something, it could be caused by infinite loop by r30451. Is this what you're saying?
> http://www.rubyist.net/~akr/chkbuild/debian/ruby-trunk/log/20110105T190101Z.log.html.gz

Ah, I remembered incorrectly. It was an infinite loop, not a SEGV.
Sorry, I made this change a while ago and couldn't remember.

> As I wrote above, bi-directional SSL close is hard to control (opposite side is not controllable) so calling SSL_shutdown() several times without checking could be enough.
>
> Here's what mod_ssl is doing.
>
> int SSL_smart_shutdown(SSL *ssl)
> {
>
> int i;
> int rc;
>
> ```
> /*
>  * Repeat the calls, because SSL_shutdown internally dispatches through a
>  * little state machine. Usually only one or two interation should be
>  * needed, so we restrict the total number of restrictions in order to
>  * avoid process hangs in case the client played bad with the socket
>  * connection and OpenSSL cannot recognize it.
>  */
> rc = 0;
> for (i = 0; i < 4 /* max 2x pending + 2x data = 4 */; i++) {
>     if ((rc = SSL_shutdown(ssl)))
>         break;
> }
> return rc;
> ```
>
> }
>
> If my guess about the error you got is correct, I'll commit similar changes for SSL_shutdown().

Yes, you are correct. It was an infinite loop and not a segv. Thanks
for investigating this for me!

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

#8 - 06/24/2011 04:10 PM - nahi (Hiroshi Nakamura)

- Category set to ext

- Status changed from Assigned to Closed

- Target version set to 1.9.3

Hi,

On Fri, Jun 24, 2011 at 02:13, Aaron Patterson aaron@tenderlovemaking.com wrote:

> > And the second part, can you remember some more details of SEGV you got? If the error you got is timeout or something, it could be caused by infinite loop by r30451. Is this what you're saying?
> > http://www.rubyist.net/~akr/chkbuild/debian/ruby-trunk/log/20110105T190101Z.log.html.gz
>
> Ah, I remembered incorrectly. It was an infinite loop, not a SEGV.
> Sorry, I made this change a while ago and couldn't remember.

No problem! I should have investigated it earlier.

> If my guess about the error you got is correct, I'll commit similar changes for SSL_shutdown().

> Yes, you are correct. It was an infinite loop and not a segv. Thanks
> for investigating this for me!

Committed at r32219. Thanks for your patience for my late response!

Files

noname	500 Bytes	06/23/2011	tenderlovemaking (Aaron Patterson)
noname	500 Bytes	06/24/2011	tenderlovemaking (Aaron Patterson)