I could be totally wrong, but it seems the standard library doesn't provide a reliable way of comparing hashes in constant-time.

- The docs for OpenSSL::HMAC encourage the use of Digest#to_s (see: 
  http://ruby-doc.org/stdlib-2.1.0/libdoc/openssl/rdoc/OpenSSL/HMAC.html#method-c-new)
- Ruby's string comparison uses memcmp, which isn't timing safe (see: http://rxr.whitequark.org/mri/source/string.c#L2382)

With this patch I propose to add an additional method, OpenSSL::HMAC#verify, which takes a binary string with a digest and compares it against the computed hash.

History

#1 - 07/28/2014 03:13 PM - arrtchiu (Matt U)
- File hmac-timing.patch added

Modified misleading error message - new patch attached.

#2 - 07/29/2014 01:25 AM - nobu (Nobuyoshi Nakada)

- Indent style mismatch
- Should try to convert the argument with StringValue()
- Why HMAC only? Other digests don't need it?
- Probably we should provide timing-safe binary compare function?

#3 - 07/29/2014 01:39 AM - arrtchiu (Matt U)
Thanks for the feedback!

Nobuyoshi Nakada wrote:

- Indent style mismatch
- Should try to convert the argument with StringValue()

Will fix - sorry, this is my first contribution!

- Why HMAC only? Other digests don't need it?

Good point, I thought since HMAC is for both redundancy and message authentication it made most sense on HMAC rather than all digests - but - I can see there would be use-cases for comparing other digests or other strings in a timing-safe manner.

- Probably we should provide timing-safe binary compare function?

I think this makes the most sense ;)
How about moving this to a method on String - e.g. String#slow_eq? or String::timing_safe_eq?? I'm terrible at naming things ;)

#4 - 07/29/2014 02:01 AM - nobu (Nobuyoshi Nakada)
Matt U wrote:

How about moving this to a method on String - e.g. String#slow_eq? or String::timing_safe_eq?? I'm terrible at naming things ;)

slow is not the main concern here, IMHO.
The latter is more descriptive, but seems less pretty a little.
Nobuyoshi Nakada wrote:

slow is not the main concern here, IMHO.
The latter is more descriptive, but seems less pretty a little.

Cleaned up (hopefully correctly) and moved to String#tsafe_eql?. Any ideas for a better name?

#6 - 07/29/2014 05:26 AM - nobu (Nobuyoshi Nakada)
Seems correct.

I made a benchmark for it:

```
# bm-tsafe_eql.rb: [Feature #10098]
require 'benchmark'
a = "x"*1024_000
b = a="y"
c = "y"+a
a << "x"
n = (ARGV.shift || 100000).to_i
Benchmark.bm(15) { |bm|  
  bm.report("a==b") { n.times{a==b} }
  bm.report("a==c") { n.times{a==c} }
  bm.report("a.tsafe_eql?(b)") { n.times{a.tsafe_eql?(b)}
  bm.report("a.tsafe_eql?(c") { n.times{a.tsafe_eql?(c)}
}

It seems quite slow.

<table>
<thead>
<tr>
<th></th>
<th>user</th>
<th>system</th>
<th>total</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>a==b</td>
<td>4.660000</td>
<td>0.000000</td>
<td>4.660000</td>
<td>( 4.672629)</td>
</tr>
<tr>
<td>a==c</td>
<td>0.010000</td>
<td>0.000000</td>
<td>0.010000</td>
<td>( 0.007154)</td>
</tr>
<tr>
<td>a.tsafe_eql?(b)</td>
<td>34.330000</td>
<td>0.020000</td>
<td>34.350000</td>
<td>( 34.366759)</td>
</tr>
<tr>
<td>a.tsafe_eql?(c)</td>
<td>34.450000</td>
<td>0.020000</td>
<td>34.470000</td>
<td>( 34.480267)</td>
</tr>
</tbody>
</table>
```

With the following patch,

```
diff --git a/string.c b/string.c
index 5c2852f..90865c0 100644
--- a/string.c
+++ b/string.c
@@ -2504,7 +2504,7 @@ static VALUE
     rb_str_tsafe_eql(VALUE str1, VALUE str2)
     {
         long len, idx;
-     char result;
     +     VALUE result;
         const char *buf1, *buf2;

         str2 = StringValue(str2);
@@ -2516,7 +2516,13 @@ rb_str_tsafe_eql(VALUE str1, VALUE str2)
         buf2 = RSTRING_PTR(str2);
         
         result = 0;
-     for (idx = 0; idx < len; idx++) {
```
According to notes on timingsafe_memcmp, OpenBSD has timingsafe_memcmp(), and NetBSD has consttime_memequal().

Nobuyoshi Nakada wrote:

According to notes on timingsafe_memcmp, OpenBSD has timingsafe_memcmp(), and NetBSD has consttime_memequal().

Wow, thank you for such detailed and valuable feedback (and an awesome patch!)

What do you think about extracting this to an (inline) method like rb_timingsafe_memcmp(..) which can then use the system-provided ones if they exist? Since this is moving into distro/platform-specific territory I'm not sure how this fits with Ruby's coding guidelines.

Agree to extract a function, I meant it by "provide timing-safe binary compare function". But the name _memcmp doesn't look nice, as it looks like returning the same result as memcmp. The order on this comparison won't make sense, so _equal or _eql would be better.

rb_tsafe_eql() doesn't need to be VALUE, int is OK. Tests for timing-safeness are desirable, but it would be fragile by noise.

I don't like the proposed method name. tsafe? It should be timingsafe. Saving some keystrokes isn't worth it to have a less obvious name. Even C programmers chose a longer name!

I'm also not happy about eql? either. Then people expect it to work like String#eql?. Same argument requirements/conversion, same result, and it's even mentioned in the proposed docs ("similarly" is vague)! But it doesn't. For example:

If something is going to be added to String, then encodings need to be considered. Or:

rb_tsafe_eql() doesn't need to be VALUE, int is OK. Tests for timing-safeness are desirable, but it would be fragile by noise.

Nobuyoshi Nakada wrote:

rb_tsafe_eql() doesn't need to be VALUE, int is OK. Tests for timing-safeness are desirable, but it would be fragile by noise.
I'll get these done. Your benchmark code demonstrated this pretty well so (if it's ok with you) I'll use that as a starting point.

cremno phobia wrote:

I don't like the proposed method name.

tsafe? It should be timingsafe. Saving some keystrokes isn't worth it to have a less obvious name. Even C programmers chose a longer name! I'm also not happy about eql? either. Then people expect it to work like String#eql?. Same argument requirements/conversion, same result, and it's even mentioned in the proposed docs ("similarly" is vague)! But it doesn't. For example:

Thanks for taking the time to test out this patch!
You have some really good points to consider. My thoughts in response:

- Since Ruby has only the one String class for text and data, I think it does make sense to keep this method on the String class.
- The behaviour you've demonstrated is intended, we care about the bytes in the buffer; not the encoding.
- Name and documentation is terrible, I agree :)

I think the easiest way to resolve this would be to come up with a better name, and to explain more clearly in the documentation.

For a starting point, how about timingsafe_bytes_eq? ? I'll improve the documentation while making the fixes as suggested by Nobu :)

#14 - 07/30/2014 12:56 PM - cremno (cremno phobia)
Matt U wrote:

- Since Ruby has only the one String class for text and data, I think it does make sense to keep this method on the String class.

I agree with you:

- The behaviour you've demonstrated is intended, we care about the bytes in the buffer; not the encoding.
- Name and documentation is terrible, I agree :)

Then that encoding will be ignored has to be mentioned in the documentation. What happens if the length of both strings differ, too.

I don't know yet how I feel about (silently) ignoring encoding. Maybe only ASCII-8BIT strings should be allowed? But having bytes in the name is good anyway! Naming things is hard. I don't have a better idea either (timingsafe_bytecmp?, but there's also casecmp…).

Are there any gems for this or is such a method part of one?

#15 - 08/23/2014 09:12 AM - arrtchiu (Matt U)

- File 0001-add-timing-safe-string-compare-method.patch added

Changelog:

- Renamed rb_tsafe_eql => rb_consttime_memequal.
- Renamed rb_str_tsafe_eql => rb_str_consttime_bytes_eq.
- Renamed tsafe_eql? => consttime_bytes_eq?.
- rb_consttime_memequal now has return type int.
- Updated documentation to reflect that encodings are ignored, and removed reference to eql?.
- Added tests to ensure timing safety (delta of 0.25 sec allowed to account for GC/system noise).
- Build on Travis passing: https://travis-ci.org/ruby/ruby/builds/33351019

#16 - 09/18/2014 09:50 AM - arrtchiu (Matt U)
Keen to hear feedback if any. Completely understand there are many more important tickets than this one, but it would be great to see this feature in MRI soon!

Devise, one of the most popular frameworks currently implements a timing-safe string compare in Ruby manually:
https://github.com/plataformatec/devise/blob/66db52ca31b5d8629f5813a1d7f03a8bc17e5d52/lib/devise.rb#L480-L488

#17 - 10/29/2014 08:52 AM - nagachika (Tomoyuki Chikanaga)

- Category changed from ext/openssl to core
- Status changed from Open to Assigned
- Assignee set to matz (Yukihiro Matsumoto)

The latest patch seems satisfy nobu, doesn't it?
At last we need to get approved from Matz.

#18 - 09/13/2015 03:31 AM - zzak (Zachary Scott)
- Assignee changed from matz (Yukihiro Matsumoto) to 7150

#19 - 02/06/2016 05:28 PM - aledovsky (Aleksandrs Ļedovskis)
Can someone clarify, what state is this feature in? Do we still need to get Matz's approval of String API change, or in light of Zachary's change "openssl" group gives the final call?

#20 - 02/25/2016 06:42 AM - arrtchiu (Matt U)
Aleksandrs Ļedovskis wrote:

Can someone clarify, what state is this feature in? Do we still need to get Matz's approval of String API change, or in light of Zachary's change "openssl" group gives the final call?

While still useful with OpenSSL, I'd say that this feature has changed since it was initially reported and no longer relates to OpenSSL. It seems this falls under Ruby's standard String API, which I assume is up to Matz.

#21 - 03/02/2016 03:41 AM - naruse (Yui NARUSE)
- Assignee changed from 7150 to matz (Yukihiro Matsumoto)

#22 - 03/17/2016 05:42 AM - shyouhei (Shyouhei Urabe)
Anyways, this issue was discussed in this month's developer meeting. Attendees at there found that OpenSSL already provides a timing-safe binary comparison function in C (namely CRYPTO_memcmp). Just not currently usable from ruby. Why not export this to ruby-level so that application programmers can use?

#23 - 04/15/2016 12:32 PM - naruse (Yui NARUSE)
Following is a patch but I just found there's OPENSSL_memcmp, which is not timing safe...

diff --git a/ext/openssl/ossl.c b/ext/openssl/ossl.c
index d03dfa7..76333e2 100644
--- a/ext/openssl/ossl.c
+++ b/ext/openssl/ossl.c
@@ -551,6 +551,21 @@
 static void Init_ossl_locks(void)
  |
 static VALUE
+stat|o VALUE dummy, VALUE str1, VALUE str2)
+{
+  const unsigned char *p1 = (const unsigned char *)StringValuePtr(str1);
+  const unsigned char *p2 = (const unsigned char *)StringValuePtr(str2);
+  long len = RSTRING_LEN(str1);
+  long i;
+  unsigned char ret = 0;
+  if (len != RSTRING_LEN(str2)) return Qfalse;
+  for (i=0; i < len; i++) {
+    ret |= p1[i] ^ p2[i];
+  }
+ return ret ? Qfalse : Qtrue;
+}
+/
+/*
 OpenSSL provides SSL, TLS and general purpose cryptography. It wraps the
 OpenSSL[http://www.openssl.org/] library.
@@ -1088,6 +1103,7 @@
 
 * OpenSSL ruby extension version
 */

#24 - 07/05/2016 09:51 AM - arrtchiu (Matt U)
Yui, I'm a little confused. The patch you have in your comment looks timing-safe to me. Also I suggest taking a look at Nobu's improvements to my code, I definitely learned a lot more about speed after reading it.
Other things that don't use OpenSSL might benefit from this feature, so my vote is to add this to the String library rather than OpenSSL.

#25 - 09/07/2016 08:55 AM - naruse (Yui NARUSE)
- Status changed from Assigned to Feedback

Even though OpenSSL uses the name memcmp, I reconsidered it is a bad name. Therefore this is back to naming issue.

#26 - 09/27/2016 09:07 AM - shyouhei (Shyouhei Urabe)

(Just to be clear) what is bad about the word "memcmp" is that the "cmp" part implies returning integers, rather than true/false. One of such example that returns integer is String#casecmp. But almost nobody wants integers for timing-safe comparisons, meseems at least. We should provide a method that returns true/false, and in doing so "memcmp" is inappropriate.

#27 - 09/27/2016 12:20 PM - aledovsky (Aleksandrs Ļedovskis)

Ok, here are couple suggestions, to keep the carriage rolling.

#safe.eql?
#secure.eql?
#timesafe.eql?
#timingsafe.eql?

#28 - 09/27/2016 12:44 PM - znz (Kazuhiro NISHIYAMA)

Rack and Rails uses secure_compare.

- http://www.rubydoc.info/github/rack/rack/Rack/Utils.secure_compare

#29 - 09/27/2016 01:37 PM - nobu (Nobuyoshi Nakada)

secure sounds vague, and compare doesn't address the point by shyouhei, I think.

#30 - 09/28/2016 12:57 AM - shyouhei (Shyouhei Urabe)

Are we going to add this method under OpenSSL namespace? If so "secure" describes almost nothing.

My suggestion is to add String#casecmp? (proposed in #12798) first, then introduce this one with OpenSSL.memcmp? naming.

#31 - 06/18/2018 01:49 AM - shyouhei (Shyouhei Urabe)

String#casecmp has already landed. It seems no blocker are there for implementing this one, except the name.

#32 - 06/25/2018 08:43 PM - bdewater (Bart de Water)

I agree Rails' secure_compare name is not great. 'Secure' is a result of the fact it's a constant-time comparison. How about String#const_time.eql? or something along those lines?

#33 - 02/04/2019 12:55 PM - shevegen (Robert A. Heiler)

I think String#const_time.eql? is not an ideal name either.

A problem with "secure" is that it can mean different things in different contexts; on class String this may be a bit more difficult since Strings in ruby can be so general purpose. Perhaps the OpenSSL namespace could have somewhat more freedom to also accept names that are not absolutely perfect? Just as comparison - although I don't think #const_time eql? is a very good name, I think it would fit a lot better into OpenSSL than it may fit towards class String as such.

'Secure' is a result of the fact it's a constant-time comparison.

Now I think that will surprise some people since they may wonder what secure has to do with any constant-time comparison per se. :)

(By the way, I think the word "compare", also as method name, is easier to reason for than e. g. "secure"; what would a "secure string" mean, for example? Would that be different or the same to a "tainted" string or is it a separate aspect? But that is just a side comment, I think Bart would like to see any forward decision towards the issue at hand considering he added it to the next upcoming developer meeting.)

#34 - 02/07/2019 07:40 AM - naruse (Yui NARUSE)
- Assignee changed from matz (Yukihiro Matsumoto) to 7150

03/08/2022
This feature is now considered up to rhenium (Kazuki Yamaguchi).

#35 - 08/18/2019 05:17 AM - bdewater (Bart de Water)
New patch over at https://github.com/ruby/openssl/pull/269

#36 - 10/20/2019 06:21 PM - bdewater (Bart de Water)
The above PR has been merged into the OpenSSL gem 🤘

#37 - 10/21/2019 02:09 AM - shyouhei (Shyouhei Urabe)
- Status changed from Assigned to Closed

Implemented in upstream. Closing.

Files

<table>
<thead>
<tr>
<th>Patch Name</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>hmac-timing.patch</td>
<td>2.5 KB</td>
<td>07/28/2014</td>
<td>arrtchiu (Matt U)</td>
</tr>
<tr>
<td>hmac-timing.patch</td>
<td>2.48 KB</td>
<td>07/28/2014</td>
<td>arrtchiu (Matt U)</td>
</tr>
<tr>
<td>tsafe_eql.patch</td>
<td>2.48 KB</td>
<td>07/29/2014</td>
<td>arrtchiu (Matt U)</td>
</tr>
<tr>
<td>tsafe_inline.patch</td>
<td>3.51 KB</td>
<td>07/29/2014</td>
<td>arrtchiu (Matt U)</td>
</tr>
<tr>
<td>0001-add-timing-safe-string-compare-method.patch</td>
<td>4.31 KB</td>
<td>08/23/2014</td>
<td>arrtchiu (Matt U)</td>
</tr>
</tbody>
</table>