Ruby master - Bug #10019

segmentation fault/buffer overrun in pack.c (encodes)

07/09/2014 02:40 PM - wkwood (Will Wood)

| Status:    | Closed          |
| Priority:  | Normal          |
| Assignee:  |                 |
| Target version: |             |
| Backport:  | 2.0.0: REQUIRED, 2.1: DONE |

ruby -v:
```
ruby 2.1.2p168 (2014-07-06 revision 46721) [i386-mingw32]
```

Description

While working with an AWS sample I hit a segmentation fault. The same sample works under 1.9.3. It appeared to be coming from pack.c function encodes. After looking at the source there's a 4K buffer allocated on the stack. I made a minor change to base the buffer length off of the incoming buffer length with a pad and allocate it off the heap. Anyway, after fixing this my code sample runs fine. I'm including a patch file and the sample code.

Associated revisions

Revision 8a608d2b - 07/11/2014 01:09 AM - nobu (Nobuyoshi Nakada)
pack.c: fix buffer overrun

- pack.c (encodes): fix buffer overrun by tail_if. Thanks to Mamoru Tasaka and Tomas Hoger. [ruby-core:63604] [Bug #10019]
git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@46728 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 46778 - 07/11/2014 01:09 AM - nobu (Nobuyoshi Nakada)
pack.c: fix buffer overrun

- pack.c (encodes): fix buffer overrun by tail_if. Thanks to Mamoru Tasaka and Tomas Hoger. [ruby-core:63604] [Bug #10019]

Revision 46778 - 07/11/2014 01:09 AM - nobu (Nobuyoshi Nakada)
pack.c: fix buffer overrun

- pack.c (encodes): fix buffer overrun by tail_if. Thanks to Mamoru Tasaka and Tomas Hoger. [ruby-core:63604] [Bug #10019]

Revision 46778 - 07/11/2014 01:09 AM - nobu (Nobuyoshi Nakada)
pack.c: fix buffer overrun

- pack.c (encodes): fix buffer overrun by tail_if. Thanks to Mamoru Tasaka and Tomas Hoger. [ruby-core:63604] [Bug #10019]

Revision 46778 - 07/11/2014 01:09 AM - nobu (Nobuyoshi Nakada)
pack.c: fix buffer overrun

- pack.c (encodes): fix buffer overrun by tail_if. Thanks to Mamoru Tasaka and Tomas Hoger. [ruby-core:63604] [Bug #10019]

Revision 46778 - 07/11/2014 01:09 AM - nobu (Nobuyoshi Nakada)
pack.c: fix buffer overrun

- pack.c (encodes): fix buffer overrun by tail_if. Thanks to Mamoru Tasaka and Tomas Hoger. [ruby-core:63604] [Bug #10019]

Revision 46778 - 07/11/2014 01:09 AM - nobu (Nobuyoshi Nakada)
pack.c: fix buffer overrun

- pack.c (encodes): fix buffer overrun by tail_if. Thanks to Mamoru Tasaka and Tomas Hoger. [ruby-core:63604] [Bug #10019]

Revision 5c1a6945 - 07/13/2014 01:59 PM - nagachika (Tomoyuki Chikanaga)
merge revision(s) 46778: [Backport #10019]

* pack.c (encodes): fix buffer overrun by tail_if. Thanks to Mamoru Tasaka and Tomas Hoger. [ruby-core:63604] [Bug #10019]
pack.c (encodes): fix buffer overrun by tail_lf. Thanks to Mamoru Tasaka and Tomas Hoger. [ruby-core:63604] [Bug #10019]

History

#1 - 07/09/2014 05:06 PM - nobu (Nobuyoshi Nakada)
- Status changed from Open to Feedback
- Priority changed from Normal to 3

encodes() fills and flush buff gradually by the buffer size.
Could you show the message at the segmentation fault?
Anyway, your patch includes unnecessary garbage.

#2 - 07/10/2014 09:24 AM - mtasaka (Mamoru Tasaka)
Looking at encodes(), it seems at least when

- len = 3070 = ((4096 - 4) / 4 * 3 + 1)
- tail_lf = 1 encodes() causes segv at https://github.com/ruby/ruby/blob/trunk/pack.c#L987

#3 - 07/10/2014 09:28 AM - mtasaka (Mamoru Tasaka)
... and type is not ‘u’.

#4 - 07/10/2014 01:23 PM - nobu (Nobuyoshi Nakada)
But if tail_lf is 1, len is a multiple of 3, so it can’t be 3070.

#5 - 07/10/2014 01:32 PM - nobu (Nobuyoshi Nakada)
Am I missing something?

#6 - 07/10/2014 06:22 PM - thoger (Tomas Hoger)
Nobuyoshi Nakada wrote:

But if tail_lf is 1, len is a multiple of 3, so it can’t be 3070.

len in encodes() can be anything between 1 and len from pack_pack() (which is a multiple of 3). It is possible to trigger mentioned off-by-one overflow one of the following ways:

- the length of the input string is 3070 (or 3071) and count value for the m format directive is 3072 or more
- count value for the m format directive is exactly 3072 and the length of the input string is n*3072 - 2 (or -1)

I.e.:

- ['a'*3070].pack("m4000")
- ['a'*(3072*3-2)].pack("m3072")

Depending on platform, compiler, compiler flags, ... this may or may not produce reliable crash.

Anyway, it's unclear if that is the problem observed by reporter. The aws-sdk and its dependencies only seem to use pack("m0"), which can not trigger this overflow.

#7 - 07/11/2014 01:09 AM - nobu (Nobuyoshi Nakada)
- Status changed from Feedback to Closed
- % Done changed from 0 to 100

Applied in changeset r46778.
Backported into ruby_2_1 branch at r46806.

It seems that the (off-by-one) issue is also present in Ruby 1.9.3, requiring a backport too.

It may also worth confirming with the reporter if he continues to observe the crash after adding the fix, since as Tomas analysis of aws-sdk gem and its dependencies indicates it's unlikely this issue is being caused by any of these gems. So, it may be the case the reporter may be referring to a different issue.

After pulling the latest code, your fix still causes a seg fault. IMO you're still overrunning the buffer.

d:/ruby-2.1.2-i386-mingw32/lib/ruby/2.1.0/base64.rb:38: [BUG] Segmentation fault
ruby 2.1.2p176 (2014-07-19 revision 46869) [i386-mingw32]
-- Control frame information -----------------------------------------------
c:0027 p:0014 s:0014 CFUNC :pack
c:0026 p:0012 s:0139 e:000138 METHOD d:/ruby-2.1.2-i386-mingw32/lib/ruby/2.1.0/base64.rb:38
s/core/signers/base.rb:
s/core/signers/s3.rb:
c:0022 p:0019 s:0097 e:000096 METHOD d:/ruby-2.1.2-i386-mingw32/lib/ruby/gems/2.1.0/gems/aws-sdk-1.48.1/lib/aws/core/client.rb:391
s/core/client.rb:
c:0020 p:0028 s:0079 e:000078 METHOD d:/ruby-2.1.2-i386-mingw32/lib/ruby/gems/2.1.0/gems/aws-sdk-1.48.1/lib/aws/core/client.rb:429 [F
s/core/client.rb:
s/core/client.rb:
c:0016 p:0009 s:0059 e:000058 METHOD d:/ruby-2.1.2-i386-mingw32/lib/ruby/gems/2.1.0/gems/aws-sdk-1.48.1/lib/aws/core/client.rb:373
s/core/client.rb:
c:0014 p:0009 s:0052 e:000051 METHOD (eval):3
s/core/client.rb:
s/core/client.rb:
s/core/client.rb:
c:0008 p:0009 s:0031 e:000030 METHOD (eval):3
s/core/client.rb:
c:0006 p:0009 s:0024 e:000023 METHOD d:/ruby-2.1.2-i386-mingw32/lib/ruby/gems/2.1.0/gems/aws-sdk-1.48.1/lib/aws/core/client.rb:246
s/core/client.rb:
c:0004 p:0009 s:0017 e:000016 METHOD d:/ruby-2.1.2-i386-mingw32/lib/ruby/gems/2.1.0/gems/aws-sdk-1.48.1/lib/aws/core/client.rb:175
s/core/client.rb:
s/core/client.rb:
s/core/client.rb:
c:0000 p:0000 s:0000 e:000001 METHOD d:/ruby-2.1.2-i386-mingw32/lib/ruby/gems/2.1.0/gems/aws-sdk-1.48.1/lib/aws/core/client.rb:1
s/core/client.rb:

BucketCleaner.rb:13:in `main'
You may have encountered a bug in the Ruby interpreter or extension libraries.
Bug reports are welcome.
For details: http://www.ruby-lang.org/bugreport.html

This application has requested the Runtime to terminate it in an unusual way.
Please contact the application's support team for more information.

#12 - 07/19/2014 11:48 PM - nobu (Nobuyoshi Nakada)
- Status changed from Closed to Feedback
- Priority changed from 3 to Normal

Will Wood wrote:

After pulling the latest code, your fix still causes a seg fault. IMO you're still overrunning the buffer.

d:/ruby-2.1.2-i386-mingw32/lib/ruby/gems/2.1.0/gems/aws-sdk-1.48.1/lib/aws/core/signers/base.rb:29:in `signature'
d:/ruby-2.1.2-i386-mingw32/lib/ruby/2.1.0/base64.rb:38:in `encode64'

Can't you show the argument to encode64?

#13 - 07/21/2014 08:36 AM - thoger (Tomas Hoger)

Will Wood wrote:

d:/ruby-2.1.2-i386-mingw32/lib/ruby/gems/2.1.0/gems/aws-sdk-1.48.1/lib/aws/core/signers/s3.rb:59:in `signature'
signature = Base.sign(creds.secret_access_key, signature, 'sha1')

d:/ruby-2.1.2-i386-mingw32/lib/ruby/gems/2.1.0/gems/aws-sdk-1.48.1/lib/aws/core/signers/base.rb:29:in 'sign'


Base64.encode64(hmac(secret, string_to_sign, digest_method)).strip

where hmac() is:


def hmac(key, value, digest = 'sha256')
  OpenSSL::HMAC.digest(OpenSSL::Digest.new(digest), key, value)
end

d:/ruby-2.1.2-i386-mingw32/lib/ruby/2.1.0/base64.rb:38:in `encode64'

http://svn.ruby-lang.org/cgi-bin/viewvc.cgi/branches/ruby_2_1/lib/base64.rb?revision=44340&view=markup#l38

[bin].pack("m")

Value passed to encode64() should be short, and of fixed size for a given digest method. For SHA-1, that's 20 bytes, hence output size is 28+1 bytes. That should be far from overflowing buff[4096].

Hence the minimal reproducer should be:

require 'openssl'
digest = OpenSSL::Digest.new('sha1')
hmac_val = OpenSSL::HMAC.digest(digest, 'secret', 'value')
print hmac_val.pack('m')

As encodes() output size only depend on input size, and not input content, it should not matter what 'secret' and 'value' are.

#14 - 07/23/2014 01:25 AM - wkwood (Will Wood)
I agree the argument should be short. I'm working on it, but it's definitely strange. If I take the buffer and allocate it off the heap it's fine. Maybe it's a build issue with RubyInstaller? Anyway, I'll poke around with and get back with you in a day or two.

#15 - 07/28/2014 02:50 PM - wkwood (Will Wood)
- File pack.c.patch added

I took a look at it yesterday. Here's the issue. In the loop (len >= 3) you check to see if there's enough room in buff. Unfortunately if len < 3 we don't flush the buffer and then write additional bytes onto the end without checking. I added a check to flush the buffer if len < 3 and then there's enough space on exiting the loop for the remaining bytes and padding including linefeed. The patch attached fixes the problem. You also don't need the + 1 byte or a 4K buffer either, your call but patch is 256 bytes for the buffer. I don't like large objects on the heap, that's just me. No seg faults for me and it works with this patch just fine. Also the rb_bug test at the end isn't necessary, your call if you want to remove it but it'll never get executed.

#16 - 07/28/2014 03:42 PM - thoger (Tomas Hoger)
Will Wood wrote:

Here's the issue. In the loop (len >= 3) you check to see if there's enough room in buff. Unfortunately if len < 3 we don't flush the buffer and then write additional bytes onto the end without checking.

The check at the end of each iteration of the while loop ensures that there are at least 4 bytes left in the buff. So there is enough space to write additional 4 output bytes for the remaining 1 or 2 input bytes with padding if 0 < len < 3. Only the additional 'n' written in the尾循环 case was not accounted for, and was addressed by the +1.

You also don't need the + 1 byte or a 4K buffer either, your call but patch is 256 bytes for the buffer.

Can you test with just one of the two changes - either force flush if len < 3, or reduce buff_size?

#17 - 07/28/2014 10:11 PM - nobu (Nobuyoshi Nakada)
Tomas Hoger wrote:

Hence the minimal reproducer should be:
```ruby
require 'openssl'
digest = OpenSSL::Digest.new('sha1')
hmac_val = OpenSSL::HMAC.digest(digest, 'secret', 'value')
print [hmac_val].pack('m')
```

I couldn't reproduce it with that code, can you?

#18 - 07/29/2014 07:55 AM - thoger (Tomas Hoger)

Nobuyoshi Nakada wrote:

I couldn't reproduce it with that code, can you?

I can't, and I explained that the output of OpenSSL::HMAC.digest is way too short to fill or overflow buff[] in encodes(). It was added to make it possible for reporter to test if he can reproduce with it. No crash on that should indicate no issue in pack/encodes.

Will's attached patches suggest that moving buff from stack to heap, or significantly reducing its size solved the problem. Maybe 256 vs. 4096 is what makes a difference between exhausting and not exhausting all stack memory in his use case?

#19 - 07/30/2014 04:48 PM - wkwood (Will Wood)

Well the reproducibility issue appears to be a toolset problem. If I build with -O2 the code works fine. I think ultimately the optimizations (O3) for gcc 4.7.2 at least with the mingw toolset are causing the issue here for windows. I've subsequently built with the mingw 4.9.1 toolset and it works fine with O3, so I guess there's a 4.7.2 gcc bug that's in the way here.

#20 - 08/01/2014 09:57 AM - pwann (Poul Wann Jensen)

This crash only triggers with FORTIFY_SOURCE. When calling rb_str_buf_cat at the end of encodes for the situation where the len variable ends up as 4. As in the example in ["a"*(3072*3-2)].pack("m3072"). This causes 1 byte corruption of the stack, triggering __fortify_fail at the epilogue of rb_str_buf_cat and the rb_bug will never be executed in this case, unless it is compiled without FORTIFY_SOURCE.

Stack canary protection on Windows should produce the same crash I suspect, but this was tested on GCC 4.8.2.

#21 - 08/04/2014 09:15 AM - thoger (Tomas Hoger)

This seems to be getting off-topic, so just few quick notes:

- It seems -fstack-protector* (SSP) is what is referred to in the previous comment, not FORTIFY_SOURCE.
- If there is encodes()’s buff[] overflow, it corrupts encodes()’s SSP cookie, that is only checked on exit from encodes(). rb_str_buf_cat() called from encodes() after overflow does not matter, as it may or may not have it’s own SSP cookie, that is checked at its exit, and that’s not corrupted by buff[] overflow. So the check leading to rb_bug() is still expected to happen, as the corrupted SSP cookie is only checked later.
- The first byte of the SSP cookie is expected to be ‘\0’ on e.g. recent Linux systems (https://sourceware.org/bugzilla/show_bug.cgi?id=10149). Hence off-by-one overflow with ‘\0’ would not be detected.

#22 - 08/04/2014 11:46 AM - nobu (Nobuyoshi Nakada)

Tomas Hoger wrote:

- The first byte of the SSP cookie is expected to be ‘\0’ on e.g. recent Linux systems (https://sourceware.org/bugzilla/show_bug.cgi?id=10149). Hence off-by-one overflow with ‘\0’ would not be detected.

encodes() does uuencode and mime-encode, so never writes ‘\0’.

#23 - 08/04/2014 11:55 AM - thoger (Tomas Hoger)

Nobuyoshi Nakada wrote:

encodes() does uuencode and mime-encode, so never writes ‘\0’.

Sorry, my bad - I already forgot it was ‘\n’ that was written out of bounds, not ‘\0’.

#24 - 08/04/2014 12:38 PM - nobu (Nobuyoshi Nakada)

(2014/07/31 1:48), wkwood@gmail.com wrote:

Well the reproducibility issue appears to be a toolset problem. If I build with -O2 the code works fine. I think ultimately the optimizations (O3) for gcc 4.7.2 at least with the mingw toolset are causing the issue here for windows. I've subsequently built with the mingw 4.9.1 toolset and it works fine with O3, so I guess there's a 4.7.2 gcc bug that's in the way here.

Couldn't you try the trunk?
#25 - 07/22/2019 07:05 PM - jeremyevans0 (Jeremy Evans)
- Status changed from Feedback to Closed

**Files**

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>pack.patch</td>
<td>2.74 KB</td>
<td>07/09/2014</td>
<td>wkwood (Will Wood)</td>
</tr>
<tr>
<td>BucketTest.rb</td>
<td>326 Bytes</td>
<td>07/09/2014</td>
<td>wkwood (Will Wood)</td>
</tr>
<tr>
<td>pack.c.patch</td>
<td>769 Bytes</td>
<td>07/28/2014</td>
<td>wkwood (Will Wood)</td>
</tr>
</tbody>
</table>