Ruby master - Bug #11030

Ruby 2.2.1 fails to compile with hardened GCC

04/03/2015 09:57 AM - mva (Vadim A. Misbakh-Soloviov)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	2.2.1	**Backport:**	2.0.0: UNKNOWN, 2.1: DONE, 2.2: DONE

Description

Hi there!
I've discovered, that Ruby 2.2.1 can't be built using Hardened GCC (4.8 and 4.9).
Probably, that was introduced in that commit:
http://svn.ruby-lang.org/cgi-bin/viewvc.cgi/tags/v2_2_1/thread_pthread.c?r1=48992&r2=49578&diff_format=h

Additional info and build logs can be found on downstream bug tracker: https://bugs.gentoo.org/show_bug.cgi?id=542610
(unfortunately, ruby maintainers in downstream are slackers, so I going to report that to upstream myself).

Related issues:

Related to Ruby master - Bug #11001: 2.2.1 Segmentation fault in reserve_stac...	**Closed**

Associated revisions

Revision 78c75612 - 04/14/2015 10:34 PM - nobu (Nobuyoshi Nakada)

thread_pthread.c: keep sp safe zone

- thread_pthread.c (reserve_stack): keep sp safe zone to get rid of crash by -fstack-check. [ruby-core:68740] [Bug #11030]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@50316 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 50316 - 04/14/2015 10:34 PM - nobu (Nobuyoshi Nakada)

thread_pthread.c: keep sp safe zone

- thread_pthread.c (reserve_stack): keep sp safe zone to get rid of crash by -fstack-check. [ruby-core:68740] [Bug #11030]

Revision 50316 - 04/14/2015 10:34 PM - nobu (Nobuyoshi Nakada)

thread_pthread.c: keep sp safe zone

- thread_pthread.c (reserve_stack): keep sp safe zone to get rid of crash by -fstack-check. [ruby-core:68740] [Bug #11030]

Revision 50316 - 04/14/2015 10:34 PM - nobu (Nobuyoshi Nakada)

thread_pthread.c: keep sp safe zone

- thread_pthread.c (reserve_stack): keep sp safe zone to get rid of crash by -fstack-check. [ruby-core:68740] [Bug #11030]

Revision 50316 - 04/14/2015 10:34 PM - nobu (Nobuyoshi Nakada)

thread_pthread.c: keep sp safe zone

- thread_pthread.c (reserve_stack): keep sp safe zone to get rid of crash by -fstack-check. [ruby-core:68740] [Bug #11030]

Revision 50316 - 04/14/2015 10:34 PM - nobu (Nobuyoshi Nakada)

thread_pthread.c: keep sp safe zone

- thread_pthread.c (reserve_stack): keep sp safe zone to get rid of crash by -fstack-check. [ruby-core:68740] [Bug #11030]

Revision 513b313d - 04/28/2015 05:14 AM - usa (Usaku NAKAMURA)

merge revision(s) 50316: [Backport #11030]

```
    * thread_pthread.c (reserve_stack): keep sp safe zone to get rid
      of crash by -fstack-check.  [ruby-core:68740] [Bug #11030]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_1@50396 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 50396 - 04/28/2015 05:14 AM - usa (Usaku NAKAMURA)

merge revision(s) 50316: [Backport #11030]

```
* thread_pthread.c (reserve_stack): keep sp safe zone to get rid
  of crash by -fstack-check.  [ruby-core:68740] [Bug #11030]
```

Revision 0224bb03 - 05/13/2015 03:28 PM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 50316: [Backport #11030]

```
    * thread_pthread.c (reserve_stack): keep sp safe zone to get rid
      of crash by -fstack-check.  [ruby-core:68740] [Bug #11030]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_2@50484 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 50484 - 05/13/2015 03:28 PM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 50316: [Backport #11030]

```
* thread_pthread.c (reserve_stack): keep sp safe zone to get rid
  of crash by -fstack-check.  [ruby-core:68740] [Bug #11030]
```

History

#1 - 04/03/2015 11:15 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Feedback

Can't you show the process memory map at the failure?
The call seems to try expanding the stack to 0x7fffff000200, but there needs more margin?

#2 - 04/10/2015 05:00 PM - mva (Vadim A. Misbakh-Soloviov)

Uh... Sorry for delay! Somewhy I didn't get email notification from your redmine :(

So, I'm not sure, if this is exactly what you asked about (if it is not — I'd be happy to provide output of any gdb command you tell ;)), but here it is.

Although, that is output from my laptop (with 16GB RAM), and not from the build server of downstream bug reporter. So, memory address you're specified in the question would probably be invalid in that mapping...

```
(gdb)  cont
Continuing.

Program received signal SIGSEGV, Segmentation fault.
0x0000555555704b9f in reserve_stack ()

(gdb)  bt
#0  0x0000555555704b9f in reserve_stack ()
#1  0x0000555555707a4b in ruby_init_stack ()
#2  0x000055555557ce32 in main ()

(gdb) info all
rax            0x7fcca0 8375456
rbx            0x7fffff7ff100   140737479962880
rcx            0x7fffff7fed60   140737479961952
rdx            0x7fc000 8372224
rsi            0x7fffffffbd80   140737488338304
rdi            0x3      3
rbp            0x7fffffffbeb0   0x7fffffffbeb0
rsp            0x7fffff7fed60   0x7fffff7fed60
r8             0x5555559e3f20   93824997015328
r9             0x5555559e3f20   93824997015328
r10            0x5555559e3f10   93824997015312
r11            0x246    582
r12            0x7fdf00 8380160
r13            0x7fffffffc020   140737488338976
r14            0x0      0
r15            0x0      0
rip            0x555555704b9f   0x555555704b9f <reserve_stack+175>
eflags         0x10206  [ PF IF RF ]
cs             0x33     51
ss             0x2b     43
```

```
ds             0x0      0
es             0x0      0
fs             0x0      0
gs             0x0      0
st0            0        (raw 0x00000000000000000000)
st1            0        (raw 0x00000000000000000000)
st2            0        (raw 0x00000000000000000000)
st3            0        (raw 0x00000000000000000000)
st4            0        (raw 0x00000000000000000000)
st5            0        (raw 0x00000000000000000000)
st6            0        (raw 0x00000000000000000000)
st7            0        (raw 0x00000000000000000000)
fctrl          0x37f    895
fstat          0x0      0
ftag           0xffff   65535
fiseg          0x0      0
fioff          0x0      0
foseg          0x0      0
fooff          0x0      0
fop            0x0      0
mxcsr          0x1f80   [ IM DM ZM OM UM PM ]
ymm0           {v8_float = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}, v4_double = {0x0, 0x0, 0x0, 0x0}, v32_int
8 = {0x0, 0xff, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xff, 0x0 <repeats 20 times>}, v16_int16 = {0xff0
0,
    0x0, 0x0, 0x0, 0x0, 0xff00, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}, v8_int32 = {0xff00, 0x0, 0x
ff000000, 0x0, 0x0, 0x0, 0x0, 0x0}, v4_int64 = {0xff00, 0xff000000, 0x0, 0x0}, v2_int128 = {
    0x00000000ff000000000000000000ff00, 0x00000000000000000000000000000000}}
ymm1           {v8_float = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}, v4_double = {0x0, 0x0, 0x0, 0x0}, v32_int
8 = {0x0 <repeats 32 times>}, v16_int16 = {0x0 <repeats 16 times>}, v8_int32 = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
    0x0, 0x0}, v4_int64 = {0x0, 0x0, 0x0, 0x0}, v2_int128 = {0x00000000000000000000000000000000, 0x000000000000
00000000000000000000}}
ymm2           {v8_float = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}, v4_double = {0x0, 0x0, 0x0, 0x0}, v32_int
8 = {0x0 <repeats 32 times>}, v16_int16 = {0x0 <repeats 16 times>}, v8_int32 = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
    0x0, 0x0}, v4_int64 = {0x0, 0x0, 0x0, 0x0}, v2_int128 = {0x00000000000000000000000000000000, 0x000000000000
00000000000000000000}}
ymm3           {v8_float = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}, v4_double = {0x0, 0x0, 0x0, 0x0}, v32_int
8 = {0x0 <repeats 32 times>}, v16_int16 = {0x0 <repeats 16 times>}, v8_int32 = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
    0x0, 0x0}, v4_int64 = {0x0, 0x0, 0x0, 0x0}, v2_int128 = {0x00000000000000000000000000000000, 0x000000000000
00000000000000000000}}
ymm4           {v8_float = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}, v4_double = {0x0, 0x0, 0x0, 0x0}, v32_int
8 = {0x0 <repeats 32 times>}, v16_int16 = {0x0 <repeats 16 times>}, v8_int32 = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
    0x0, 0x0}, v4_int64 = {0x0, 0x0, 0x0, 0x0}, v2_int128 = {0x00000000000000000000000000000000, 0x000000000000
00000000000000000000}}
ymm5           {v8_float = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}, v4_double = {0x0, 0x0, 0x0, 0x0}, v32_int
8 = {0x0 <repeats 32 times>}, v16_int16 = {0x0 <repeats 16 times>}, v8_int32 = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
    0x0, 0x0}, v4_int64 = {0x0, 0x0, 0x0, 0x0}, v2_int128 = {0x00000000000000000000000000000000, 0x000000000000
00000000000000000000}}
ymm6           {v8_float = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}, v4_double = {0x0, 0x0, 0x0, 0x0}, v32_int
8 = {0x0 <repeats 32 times>}, v16_int16 = {0x0 <repeats 16 times>}, v8_int32 = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
    0x0, 0x0}, v4_int64 = {0x0, 0x0, 0x0, 0x0}, v2_int128 = {0x00000000000000000000000000000000, 0x000000000000
00000000000000000000}}
ymm7           {v8_float = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}, v4_double = {0x0, 0x0, 0x0, 0x0}, v32_int
8 = {0x0 <repeats 32 times>}, v16_int16 = {0x0 <repeats 16 times>}, v8_int32 = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
    0x0, 0x0}, v4_int64 = {0x0, 0x0, 0x0, 0x0}, v2_int128 = {0x00000000000000000000000000000000, 0x000000000000
00000000000000000000}}
ymm8           {v8_float = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}, v4_double = {0x0, 0x0, 0x0, 0x0}, v32_int
8 = {0x0 <repeats 32 times>}, v16_int16 = {0x0 <repeats 16 times>}, v8_int32 = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
    0x0, 0x0}, v4_int64 = {0x0, 0x0, 0x0, 0x0}, v2_int128 = {0x00000000000000000000000000000000, 0x000000000000
00000000000000000000}}
ymm9           {v8_float = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}, v4_double = {0x0, 0x0, 0x0, 0x0}, v32_int
8 = {0x0 <repeats 32 times>}, v16_int16 = {0x0 <repeats 16 times>}, v8_int32 = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
    0x0, 0x0}, v4_int64 = {0x0, 0x0, 0x0, 0x0}, v2_int128 = {0x00000000000000000000000000000000, 0x000000000000
00000000000000000000}}
ymm10          {v8_float = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}, v4_double = {0x0, 0x0, 0x0, 0x0}, v32_int
8 = {0x0 <repeats 32 times>}, v16_int16 = {0x0 <repeats 16 times>}, v8_int32 = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
    0x0, 0x0}, v4_int64 = {0x0, 0x0, 0x0, 0x0}, v2_int128 = {0x00000000000000000000000000000000, 0x000000000000
00000000000000000000}}
ymm11          {v8_float = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}, v4_double = {0x0, 0x0, 0x0, 0x0}, v32_int
8 = {0x0 <repeats 32 times>}, v16_int16 = {0x0 <repeats 16 times>}, v8_int32 = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
    0x0, 0x0}, v4_int64 = {0x0, 0x0, 0x0, 0x0}, v2_int128 = {0x00000000000000000000000000000000, 0x000000000000
00000000000000000000}}
ymm12          {v8_float = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}, v4_double = {0x0, 0x8000000000000000, 0x0
, 0x0}, v32_int8 = {0x0, 0xff, 0x0, 0x0, 0x0, 0xff, 0x0, 0x0, 0x0, 0xff, 0x0, 0x0, 0x0, 0x0, 0x0, 0xff,
    0x0 <repeats 16 times>}, v16_int16 = {0xff00, 0x0, 0xff00, 0x0, 0xff00, 0x0, 0x0, 0xff00, 0x0, 0x0, 0x0, 0x0, 0
x0, 0x0, 0x0, 0x0, 0x0, 0x0}, v8_int32 = {0xff00, 0xff00, 0xff00, 0xff000000, 0x0, 0x0, 0x0, 0x0}, v4_int64 = {
```

```
    0xff000000ff00, 0xff0000000000ff00, 0x0, 0x0}, v2_int128 = {0xff0000000000ff000000ff000000ff00, 0x00000000
00000000000000000000000}}
ymm13          {v8_float = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}, v4_double = {0x0, 0x0, 0x0, 0x0}, v32_int
8 = {0x0 <repeats 32 times>}, v16_int16 = {0x0 <repeats 16 times>}, v8_int32 = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
    0x0, 0x0}, v4_int64 = {0x0, 0x0, 0x0, 0x0}, v2_int128 = {0x00000000000000000000000000000000, 0x00000000000
00000000000000000000}}
ymm14          {v8_float = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}, v4_double = {0x0, 0x0, 0x0, 0x0}, v32_int
8 = {0x0 <repeats 32 times>}, v16_int16 = {0x0 <repeats 16 times>}, v8_int32 = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
    0x0, 0x0}, v4_int64 = {0x0, 0x0, 0x0, 0x0}, v2_int128 = {0x00000000000000000000000000000000, 0x00000000000
00000000000000000000}}
ymm15          {v8_float = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}, v4_double = {0x0, 0x0, 0x0, 0x0}, v32_int
8 = {0x0 <repeats 32 times>}, v16_int16 = {0x0 <repeats 16 times>}, v8_int32 = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
    0x0, 0x0}, v4_int64 = {0x0, 0x0, 0x0, 0x0}, v2_int128 = {0x00000000000000000000000000000000, 0x00000000000
00000000000000000000}}

(gdb) info proc all
process 73683
cmdline = '/var/tmp/portage/dev-lang/ruby-2.2.1/work/ruby-2.2.1/miniruby'
cwd = '/var/tmp/portage/dev-lang/ruby-2.2.1/work/ruby-2.2.1'
exe = '/var/tmp/portage/dev-lang/ruby-2.2.1/work/ruby-2.2.1/miniruby'
Mapped address spaces:

          Start Addr           End Addr       Size     Offset objfile
      0x555555554000     0x5555557cd000   0x279000        0x0 /var/tmp/portage/dev-lang/ruby-2.2.1/work/ruby-2
.2.1/miniruby
      0x5555559cc000     0x5555559d2000     0x6000   0x278000 /var/tmp/portage/dev-lang/ruby-2.2.1/work/ruby-2
.2.1/miniruby
      0x5555559d2000     0x5555559d3000     0x1000   0x27e000 /var/tmp/portage/dev-lang/ruby-2.2.1/work/ruby-2
.2.1/miniruby
      0x5555559d3000     0x555555a04000    0x31000        0x0 [heap]
      0x7ffff6b88000     0x7ffff6e42000   0x2ba000        0x0 /usr/lib64/locale/locale-archive
      0x7ffff6e42000     0x7ffff6ffa000   0x1b8000        0x0 /lib64/libc-2.19.so
      0x7ffff6ffa000     0x7ffff71f9000   0x1ff000   0x1b8000 /lib64/libc-2.19.so
      0x7ffff71f9000     0x7ffff71fd000     0x4000   0x1b7000 /lib64/libc-2.19.so
      0x7ffff71fd000     0x7ffff71ff000     0x2000   0x1bb000 /lib64/libc-2.19.so
      0x7ffff71ff000     0x7ffff7203000     0x4000        0x0
      0x7ffff7203000     0x7ffff7306000   0x103000        0x0 /lib64/libm-2.19.so
      0x7ffff7306000     0x7ffff7505000   0x1ff000   0x103000 /lib64/libm-2.19.so
      0x7ffff7505000     0x7ffff7506000     0x1000   0x102000 /lib64/libm-2.19.so
      0x7ffff7506000     0x7ffff7507000     0x1000   0x103000 /lib64/libm-2.19.so
      0x7ffff7507000     0x7ffff750f000     0x8000        0x0 /lib64/libcrypt-2.19.so
      0x7ffff750f000     0x7ffff770f000   0x200000     0x8000 /lib64/libcrypt-2.19.so
      0x7ffff770f000     0x7ffff7710000     0x1000     0x8000 /lib64/libcrypt-2.19.so
      0x7ffff7710000     0x7ffff7711000     0x1000     0x9000 /lib64/libcrypt-2.19.so
      0x7ffff7711000     0x7ffff773f000    0x2e000        0x0
      0x7ffff773f000     0x7ffff7742000     0x3000        0x0 /lib64/libdl-2.19.so
      0x7ffff7742000     0x7ffff7941000   0x1ff000     0x3000 /lib64/libdl-2.19.so
      0x7ffff7941000     0x7ffff7942000     0x1000     0x2000 /lib64/libdl-2.19.so
      0x7ffff7942000     0x7ffff7943000     0x1000     0x3000 /lib64/libdl-2.19.so
      0x7ffff7943000     0x7ffff79b9000    0x76000        0x0 /usr/lib64/libgmp.so.10.2.0
      0x7ffff79b9000     0x7ffff7bb8000   0x1ff000    0x76000 /usr/lib64/libgmp.so.10.2.0
      0x7ffff7bb8000     0x7ffff7bb9000     0x1000    0x75000 /usr/lib64/libgmp.so.10.2.0
      0x7ffff7bb9000     0x7ffff7bba000     0x1000    0x76000 /usr/lib64/libgmp.so.10.2.0
      0x7ffff7bba000     0x7ffff7bd4000    0x1a000        0x0 /lib64/libpthread-2.19.so
      0x7ffff7bd4000     0x7ffff7dd4000   0x200000    0x1a000 /lib64/libpthread-2.19.so
      0x7ffff7dd4000     0x7ffff7dd5000     0x1000    0x1a000 /lib64/libpthread-2.19.so
      0x7ffff7dd5000     0x7ffff7dd6000     0x1000    0x1b000 /lib64/libpthread-2.19.so
      0x7ffff7dd6000     0x7ffff7dda000     0x4000        0x0
      0x7ffff7dda000     0x7ffff7dfc000    0x22000        0x0 /lib64/ld-2.19.so
      0x7ffff7f7a000     0x7ffff7f7f000     0x5000        0x0
      0x7ffff7ff7000     0x7ffff7ff8000     0x1000        0x0
      0x7ffff7ff8000     0x7ffff7ffa000     0x2000        0x0 [vvar]
      0x7ffff7ffa000     0x7ffff7ffc000     0x2000        0x0 [vdso]
      0x7ffff7ffc000     0x7ffff7ffd000     0x1000    0x22000 /lib64/ld-2.19.so
      0x7ffff7ffd000     0x7ffff7ffe000     0x1000    0x23000 /lib64/ld-2.19.so
      0x7ffff7ffe000     0x7ffff7fff000     0x1000        0x0
      0x7ffffff7ff000     0x7fffffffff000   0x800000        0x0 [stack]
  0xffffffffff600000 0xffffffffff601000     0x1000        0x0 [vsyscall]
Name:   miniruby
State:  t (tracing stop)
Tgid:   73683
Ngid:   0
Pid:    73683
PPid:   73681
TracerPid:      73681
```

```
Uid:    0       0       0       0
Gid:    0       0       0       0
FDSize: 64
Groups: 0 1 2 3 4 6 10 11 20 26 27
VmPeak:    29900 kB
VmSize:    29900 kB
VmLck:         0 kB
VmPin:         0 kB
VmHWM:     10640 kB
VmRSS:     10640 kB
VmData:      456 kB
VmStk:      8196 kB
VmExe:      2532 kB
VmLib:      3552 kB
VmPTE:        76 kB
VmSwap:        0 kB
Threads:        1
SigQ:   0/63595
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 0000000001001000
SigCgt: 0000000180000000
CapInh: 0000000000000000
CapPrm: 0000003fffffffff
CapEff: 0000003fffffffff
CapBnd: 0000003fffffffff
Seccomp:        0
Cpus_allowed:   ff
Cpus_allowed_list:      0-7
Mems_allowed:   00000000,00000001
Mems_allowed_list:      0
voluntary_ctxt_switches:        8
nonvoluntary_ctxt_switches:     2
Process: 73683
Exec file: miniruby
State: t
Parent process: 73681
Process group: 73683
Session id: 61853
TTY: 34831
TTY owner process group: 73681
Flags: 0x40006100
Minor faults (no memory page): 2338
Minor faults, children: 0
Major faults (memory page faults): 0
Major faults, children: 0
utime: 0
stime: 0
utime, children: 0
stime, children: 0
jiffies remaining in current time slice: 20
'nice' value: 0
jiffies until next timeout: 1
jiffies until next SIGALRM: 0
start time (jiffies since system boot): 27131408
Virtual memory size: 30617600
Resident set size: 2660
rlim: 18446744073709551615
Start of text: 0x555555554000
End of text: 0x5555557cc3e4
Start of stack: 0x7fffffffc020
```

#3 - 04/11/2015 09:41 PM - mva (Vadim A. Misbakh-Soloviov)

Is it any more info I can provide? :)

#4 - 04/14/2015 04:52 PM - mva (Vadim A. Misbakh-Soloviov)

By the way, I just found (with help of the downstream's Ruby team), that adding -fno-stack-check to the CFLAGS makes it to build fine (and miniruby don't segafults). But I think, it is not the good way to solve this problem :-/

#5 - 04/14/2015 08:30 PM - mva (Vadim A. Misbakh-Soloviov)

Oh, and yes, adding -fstack-check to the CFLAGS even on non-Hardened system/toolchain reproduce the error ;)

P.S. As for now, latest Ruby-2.1 release is affected too!

#6 - 04/14/2015 10:35 PM - nobu (Nobuyoshi Nakada)

- Status changed from Feedback to Closed

- % Done changed from 0 to 100

Applied in changeset r50316.

---

thread_pthread.c: keep sp safe zone

- thread_pthread.c (reserve_stack): keep sp safe zone to get rid of crash by -fstack-check. [ruby-core:68740] [Bug #11030]

#7 - 04/15/2015 01:45 PM - nagachika (Tomoyuki Chikanaga)

- Backport changed from 2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN to 2.0.0: UNKNOWN, 2.1: REQUIRED, 2.2: REQUIRED

#8 - 04/28/2015 05:14 AM - usa (Usaku NAKAMURA)

- Backport changed from 2.0.0: UNKNOWN, 2.1: REQUIRED, 2.2: REQUIRED to 2.0.0: UNKNOWN, 2.1: DONE, 2.2: REQUIRED

ruby_2_1 r50396 merged revision(s) 50316.

#9 - 05/13/2015 03:28 PM - nagachika (Tomoyuki Chikanaga)

- Backport changed from 2.0.0: UNKNOWN, 2.1: DONE, 2.2: REQUIRED to 2.0.0: UNKNOWN, 2.1: DONE, 2.2: DONE

Backported into ruby_2_2 branch at r50484.

#10 - 06/04/2015 09:45 AM - nobu (Nobuyoshi Nakada)

- Related to Bug #11001: 2.2.1 Segmentation fault in reserve_stack() function. added

#11 - 06/14/2018 10:28 AM - knedlsepp (Josef Kemetmüller)

For me it helped to increase my stack size limit using ulimit -s unlimited.