Ruby trunk - Bug #14716

SecureRandom throwing an error in Ruby 2.5.1

04/27/2018 06:19 AM - snehavas (sneha vasanth)

Status:	Open		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	2.5.1	**Backport:**	2.3: DONTNEED, 2.4: DONTNEED, 2.5: REQUIRED

Description

Hi,

We recently upgraded from ruby 2.3.6 to 2.5.1.
We use SecureRandom.uuid to generate a random number for our session.
Post the upgrade we have been getting the following error intermittently

```
app error: failed to get urandom (RuntimeError)
E, [2018-04-27T04:55:08.741859 #16550] ERROR -- : /usr/lib/ruby/2.5.0/securerandom.rb:99:in `urand
om'
E, [2018-04-27T04:55:08.741898 #16550] ERROR -- : /usr/lib/ruby/2.5.0/securerandom.rb:99:in `gen_r
andom_urandom'
E, [2018-04-27T04:55:08.741932 #16550] ERROR -- : /usr/lib/ruby/2.5.0/securerandom.rb:129:in `rand
om_bytes'
E, [2018-04-27T04:55:08.741965 #16550] ERROR -- : /usr/lib/ruby/2.5.0/securerandom.rb:219:in `uuid
'
E, [2018-04-27T04:55:08.741997 #16550] ERROR -- : /usr/share/nginx/frontend/app/utilities/log.rb:7
4:in `create_session_info'
E, [2018-04-27T04:55:08.742036 #16550] ERROR -- : /usr/share/nginx/frontend/app/utilities/log.rb:1
1:in `context'
```

We understand that there was a change in ruby 2.5.1 where we now look at OS sources as the first point of contact to generate random numbers as opposed to OpenSSL.
Any idea why this could be happening?

Related issues:

Related to Ruby trunk - Bug #9569: SecureRandom should try /dev/urandom first	**Closed**

History

#1 - 04/27/2018 06:27 AM - nobu (Nobuyoshi Nakada)

- Related to Bug #9569: SecureRandom should try /dev/urandom first added

#2 - 04/27/2018 06:29 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Feedback

- Description updated

It depends on your OS.
What OS and the kernel version?

#3 - 04/27/2018 07:34 AM - snehavas (sneha vasanth)

We are using Ubuntu 14.04.5 LTS

#4 - 04/27/2018 07:36 AM - snehavas (sneha vasanth)

The Kernel version is 3.13.0-145-generic

#5 - 04/27/2018 08:45 AM - naruse (Yui NARUSE)

- Backport changed from 2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: UNKNOWN to 2.3: DONTNEED, 2.4: DONTNEED, 2.5: REQUIRED

- Status changed from Feedback to Open

r57307 changed to use __NR_getrandom (kernel header) from SYS_getrandom (glibc header).
But __NR_getrandom is from v3.17.
It needs to check both __NR_getrandom and SYS_getrandom for compatibility.

#6 - 04/27/2018 08:54 AM - naruse (Yui NARUSE)

naruse (Yui NARUSE) wrote:

> r57307 changed to use __NR_getrandom (kernel header) from SYS_getrandom (glibc header).
> But __NR_getrandom is from v3.17.
> It needs to check both __NR_getrandom and SYS_getrandom for compatibility.

genrandom(2) is introduced at v3.17.
On v3.13 kernel, it should fallback to /dev/urandom.
Maybe your environment doesn't have /dev/urandom?

#7 - 04/27/2018 09:23 AM - shyouhei (Shyouhei Urabe)

I can reproduce the situation using this Dockerfile https://github.com/shyouhei/docker-library/blob/master/%2314716/Dockerfile

The key point is to delete /dev/urandom from the image.

2.3.6 used to try openssl first, so if it had enough entropy that was okay. 2.5.1 prefers /dev/urandom (or getrandom(2), for newer kernels).

#8 - 04/27/2018 09:44 AM - snehavas (sneha vasanth)

We checked that by executing 'cat /dev/urandom' and we did see an output.
So dont think thats an issue.

We suspect that this issue occurs when there are many concurrent requests.

#9 - 08/01/2018 10:54 PM - patbl (Patrick Brinich-Langlois)

We also got this error after upgrading to 2.5.1 (we had been on 2.4.2). We're also on a v3.13 kernel (3.13.0-24-generic). cat /dev/urandom produces output, and the errors are intermittent. Over the past day, failed to get urandom was our most common error, but there were several hours when we didn't get any such errors. So it seems possible that it occurs only when there are large numbers of requests, but I didn't try to see whether the request rate was correlated with the error rate.

#10 - 09/06/2018 05:44 PM - patbl (Patrick Brinich-Langlois)

I worked around this problem by forcing SecureRandom to use the OpenSSL gem:

```
module SecureRandom
  if RUBY_VERSION == "2.5.1"
    class << self
      def gen_random(n)
        begin
          require 'openssl'
        rescue NoMethodError
          raise NotImplementedError, "No random device"
        else
          @rng_chooser.synchronize do
            class << self
              remove_method :gen_random
              alias gen_random gen_random_openssl
            end
          end
          return gen_random(n)
        end
      end
    end
  else
    raise "check whether this monkey patch is still correct"
  end
end
```