Ruby master - Bug #9544

Ruby resolver not using autoport

02/21/2014 12:41 AM - samu (Jakub Szafranski)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:	2.2.0		
ruby -v:	ruby 2.1.0p0 (2013-12-25 revision 44422) [x86_64-freebsd9.1]	**Backport:**	2.0.0: DONE, 2.1: DONE

Description

Problem

On one of my production servers I've noticed that customers were failing to install anything using gem and the latest ruby. After a bit of debugging we've found out, that it's related to ruby resolve module:

```
p Resolv.getaddress "google.com"
Errno::EPERM: Operation not permitted - bind(2) for "0.0.0.0" port 62374
from /home/pudlobe/.rvm/rubies/ruby-2.1.0/lib/ruby/2.1.0/resolv.rb:654:in bind'
from /home/pudlobe/.rvm/rubies/ruby-2.1.0/lib/ruby/2.1.0/resolv.rb:654:inbind_random_port'
from /home/pudlobe/.rvm/rubies/ruby-2.1.0/lib/ruby/2.1.0/resolv.rb:747:in block in initialize'
from /home/pudlobe/.rvm/rubies/ruby-2.1.0/lib/ruby/2.1.0/resolv.rb:735:ineach'
...
```

The interesting part is bind_random_port function. What for? The standard way of binding to a random port for udp connection is to use port 0. And on that particular machine it fails because it's using a mac_portacl module to filter which user can bind to what ports. **However, port 0 is excepted from this rule, because it's the AUTOPORT** - practically every system that allows such port filtering also allows to set an exception for the autoport.

Docs

Purpose:

Port 0 is officially a reserved port in TCP/IP networking, meaning that it should not be used for any TCP or UDP network communications. However, port 0 sometimes takes on a special meaning in network programming, particularly Unix socket programming. In that environment, port 0 is a programming technique for specifying system-allocated (dynamic) ports.
Description:

Configuring a new socket connection requires assigning a TCP or UDP port number. Instead of hard-coding a particular port number, or writing code that searches for an available port on the local system, network programmers can instead specify port 0 as a connection parameter. That triggers the operating system to automatically search for and return the next available port in the dynamic port number range.

Impact

This bug affects every system that has a restricted port-binding policy, making ruby unavailable for security-freak admins ;)

Suggested fix:

Either use port 0 to bind to the port, or at least make an option for the system admin/end user to specify the port by himself.

Associated revisions

Revision db537c51 - 02/22/2014 05:38 PM - akr (Akira Tanaka)

- lib/resolv.rb (bind_random_port): Rescue EPERM for FreeBSD which security.mac.portacl.port_high is changed. See mac_portacl(4) for details. Reported by Jakub Szafranski. [ruby-core:60917] [Bug #9544]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@45144 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 45144 - 02/22/2014 05:38 PM - akr (Akira Tanaka)

- lib/resolv.rb (bind_random_port): Rescue EPERM for FreeBSD which security.mac.portacl.port_high is changed. See mac_portacl(4) for details. Reported by Jakub Szafranski. [ruby-core:60917] [Bug #9544]

Revision 45144 - 02/22/2014 05:38 PM - akr (Akira Tanaka)

- lib/resolv.rb (bind_random_port): Rescue EPERM for FreeBSD which security.mac.portacl.port_high is changed. See mac_portacl(4) for details. Reported by Jakub Szafranski. [ruby-core:60917] [Bug #9544]

Revision 45144 - 02/22/2014 05:38 PM - akr (Akira Tanaka)

- lib/resolv.rb (bind_random_port): Rescue EPERM for FreeBSD which security.mac.portacl.port_high is changed. See mac_portacl(4) for details. Reported by Jakub Szafranski. [ruby-core:60917] [Bug #9544]

Revision 45144 - 02/22/2014 05:38 PM - akr (Akira Tanaka)

- lib/resolv.rb (bind_random_port): Rescue EPERM for FreeBSD which security.mac.portacl.port_high is changed. See mac_portacl(4) for details. Reported by Jakub Szafranski. [ruby-core:60917] [Bug #9544]

Revision 45144 - 02/22/2014 05:38 PM - akr (Akira Tanaka)

- lib/resolv.rb (bind_random_port): Rescue EPERM for FreeBSD which security.mac.portacl.port_high is changed. See mac_portacl(4) for details. Reported by Jakub Szafranski. [ruby-core:60917] [Bug #9544]

Revision 45144 - 02/22/2014 05:38 PM - akr (Akira Tanaka)

- lib/resolv.rb (bind_random_port): Rescue EPERM for FreeBSD which security.mac.portacl.port_high is changed. See mac_portacl(4) for details. Reported by Jakub Szafranski. [ruby-core:60917] [Bug #9544]

Revision 61754d1a - 07/23/2014 01:44 PM - nagachika (Tomoyuki Chikanaga)

merge revision(s) r45144: [Backport #9544]

```
    * lib/resolv.rb (bind_random_port): Rescue EPERM for FreeBSD which
      security.mac.portacl.port_high is changed.
      See mac_portacl(4) for details.
      Reported by Jakub Szafranski.  [ruby-core:60917] [Bug #9544]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_1@46909 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 46909 - 07/23/2014 01:44 PM - nagachika (Tomoyuki Chikanaga)

merge revision(s) r45144: [Backport #9544]

```
* lib/resolv.rb (bind_random_port): Rescue EPERM for FreeBSD which
  security.mac.portacl.port_high is changed.
  See mac_portacl(4) for details.
  Reported by Jakub Szafranski.  [ruby-core:60917] [Bug #9544]
```

Revision 96fac49c - 08/31/2014 07:22 AM - usa (Usaku NAKAMURA)

merge revision(s) 45144: [Backport #9544]

```
    * lib/resolv.rb (bind_random_port): Rescue EPERM for FreeBSD which
      security.mac.portacl.port_high is changed.
      See mac_portacl(4) for details.
      Reported by Jakub Szafranski.  [ruby-core:60917] [Bug #9544]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_0_0@47335 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 47335 - 08/31/2014 07:22 AM - usa (Usaku NAKAMURA)

merge revision(s) 45144: [Backport #9544]

```
* lib/resolv.rb (bind_random_port): Rescue EPERM for FreeBSD which
  security.mac.portacl.port_high is changed.
  See mac_portacl(4) for details.
  Reported by Jakub Szafranski.  [ruby-core:60917] [Bug #9544]
```

History

#1 - 02/21/2014 12:59 AM - akr (Akira Tanaka)

- Status changed from Open to Feedback

bind_random_port chooses more random than using the port 0.

The choosen ports by the port 0 is guessable from an attacker.
Some OS chooses it incrementaly.
So the attackker may be able to inject spoofed result.

What I'm not sure is why EPERM is occur.
If there is reasonable reason, we can add EPERM to the retry condition.

#2 - 02/21/2014 06:16 AM - akr (Akira Tanaka)

- Priority changed from 5 to Normal

I tested several OSs and I found NetBSD 6.1.3 still doesn't randomize port:

```
% ruby -rsocket -e '10.times { p Addrinfo.udp("0.0.0.0", 0).bind.local_address }'
#<Addrinfo: 0.0.0.0:65441 UDP>
#<Addrinfo: 0.0.0.0:65440 UDP>
#<Addrinfo: 0.0.0.0:65439 UDP>
#<Addrinfo: 0.0.0.0:65438 UDP>
#<Addrinfo: 0.0.0.0:65437 UDP>
#<Addrinfo: 0.0.0.0:65436 UDP>
#<Addrinfo: 0.0.0.0:65435 UDP>
#<Addrinfo: 0.0.0.0:65434 UDP>
#<Addrinfo: 0.0.0.0:65433 UDP>
#<Addrinfo: 0.0.0.0:65432 UDP>
% uname -mrsv
NetBSD 6.1.3 NetBSD 6.1.3 (GENERIC) amd64
```

#3 - 02/21/2014 11:01 AM - samu (Jakub Szafranski)

EPERM may happen if you're using (like me) bind port filter policy. For instance, FreeBSD mac_portacl (http://www.freebsd.org/doc/handbook/mac-portacl.html) provides such a feature, to prevent users from running their own daemons on ports we don't want them to.

I still think that this should be an OS concern to properly randomize source port, not really language case. BUT if you insist on handling this at the language level, then either catch this exception, or allow (by an environment variable, perhaps) to forcefully set the 0 port.

#4 - 02/21/2014 09:12 PM - samu (Jakub Szafranski)

I'd like to mention that you've got a hardcoded range of ports that can be used to bind(), however every system imeplents its own method to change that range - for instance to lower the minimum port, or keep users in ports from range 60000 - 65535.

Also, this resolver does not support nsswitch, which is a big, big minus.

#5 - 02/22/2014 12:51 PM - samu (Jakub Szafranski)

Akira Tanaka wrote:

> I tested several OSs and I found NetBSD 6.1.3 still doesn't randomize port:
>
> ```
> % ruby -rsocket -e '10.times { p Addrinfo.udp("0.0.0.0", 0).bind.local_address }'
> #<Addrinfo: 0.0.0.0:65441 UDP>
> #<Addrinfo: 0.0.0.0:65440 UDP>
> #<Addrinfo: 0.0.0.0:65439 UDP>
> #<Addrinfo: 0.0.0.0:65438 UDP>
> #<Addrinfo: 0.0.0.0:65437 UDP>
> #<Addrinfo: 0.0.0.0:65436 UDP>
> #<Addrinfo: 0.0.0.0:65435 UDP>
> #<Addrinfo: 0.0.0.0:65434 UDP>
> #<Addrinfo: 0.0.0.0:65433 UDP>
> #<Addrinfo: 0.0.0.0:65432 UDP>
> % uname -mrsv
> NetBSD 6.1.3 NetBSD 6.1.3 (GENERIC) amd64
> ```

ruby -rsocket -e '10.times { p Addrinfo.udp("0.0.0.0", 0).bind.local_address }'
#<Addrinfo: 0.0.0.0:65533 UDP>
#<Addrinfo: 0.0.0.0:65532 UDP>

#<Addrinfo: 0.0.0.0:65531 UDP>
#<Addrinfo: 0.0.0.0:65530 UDP>
#<Addrinfo: 0.0.0.0:65529 UDP>
#<Addrinfo: 0.0.0.0:65528 UDP>
#<Addrinfo: 0.0.0.0:65527 UDP>
#<Addrinfo: 0.0.0.0:65526 UDP>
#<Addrinfo: 0.0.0.0:65525 UDP>
#<Addrinfo: 0.0.0.0:65524 UDP>

BUT:

sysctl -w net.inet.udp.rfc6056.selected=random_pick
net.inet.udp.rfc6056.selected: bsd -> random_pick

ruby -rsocket -e '10.times { p Addrinfo.udp("0.0.0.0", 0).bind.local_address }'

#<Addrinfo: 0.0.0.0:56358 UDP>
#<Addrinfo: 0.0.0.0:52365 UDP>
#<Addrinfo: 0.0.0.0:58857 UDP>
#<Addrinfo: 0.0.0.0:53113 UDP>
#<Addrinfo: 0.0.0.0:49585 UDP>
#<Addrinfo: 0.0.0.0:62833 UDP>
#<Addrinfo: 0.0.0.0:65299 UDP>
#<Addrinfo: 0.0.0.0:53542 UDP>
#<Addrinfo: 0.0.0.0:60367 UDP>
#<Addrinfo: 0.0.0.0:52945 UDP>

uname -mrsv
NetBSD 6.1.3 NetBSD 6.1.3 (GENERIC) amd64

So basically, the system admin can change the random port alghoritm, and he can choose from a variety of alghoritms:

# sysctl net.inet.udp.rfc6056.available
net.inet.udp.rfc6056.available = bsd random_start random_pick hash doublehash randinc

Once again - I really think that it's not ruby case to randomize the port - in my opinion, this should always rely on the underlying system, and such thing shouldn't be forced by the language itself.

#6 - 02/22/2014 02:28 PM - akr (Akira Tanaka)

I found GNU/Hurd and Haiku also allocates ports sequentially.

GNU/Hurd:

```
% ./ruby -v -rsocket -e '10.times { p Addrinfo.udp("0.0.0.0", 0).bind.local_address }'
ruby 2.2.0dev (2014-01-21 trunk 44678) [i686-gnu0.3]
#<Addrinfo: 0.0.0.0:32777 UDP>
#<Addrinfo: 0.0.0.0:32778 UDP>
#<Addrinfo: 0.0.0.0:32779 UDP>
#<Addrinfo: 0.0.0.0:32780 UDP>
#<Addrinfo: 0.0.0.0:32781 UDP>
#<Addrinfo: 0.0.0.0:32782 UDP>
#<Addrinfo: 0.0.0.0:32783 UDP>
#<Addrinfo: 0.0.0.0:32784 UDP>
#<Addrinfo: 0.0.0.0:32785 UDP>
#<Addrinfo: 0.0.0.0:32786 UDP>
% uname -mrsv
GNU 0.3 GNU-Mach 1.3.99-486/Hurd-0.3 i686-AT386
```

Haiku:

```
> ./ruby -v -rsocket -e '10.times { p Addrinfo.udp("0.0.0.0", 0).bind.local_address }'
ruby 2.1.0dev (2013-05-11 trunk 40642) [i586-haiku]
#<Addrinfo: 0.0.0.0:63208 UDP>
#<Addrinfo: 0.0.0.0:63209 UDP>
#<Addrinfo: 0.0.0.0:63210 UDP>
#<Addrinfo: 0.0.0.0:63211 UDP>
#<Addrinfo: 0.0.0.0:63212 UDP>
#<Addrinfo: 0.0.0.0:63213 UDP>
#<Addrinfo: 0.0.0.0:63214 UDP>
#<Addrinfo: 0.0.0.0:63215 UDP>
#<Addrinfo: 0.0.0.0:63216 UDP>
#<Addrinfo: 0.0.0.0:63217 UDP>
> uname -mrsv
```

```
Haiku 1 hrevr1alpha4-44702 Nov 14 2012  BePC
```

#7 - 02/22/2014 02:34 PM - samu (Jakub Szafranski)

So is it ruby's concern, or should the system developers make appropriate patches?

Currently you've reinvented the wheel in a very bad way - it is unusable with nsswitch, it is unusable with custom firewall policies, unusable with custom system port range.

A patch in one language won't fix the problem globally. It's system developers, who should think of improving their source port finding alghoritm.

#8 - 02/22/2014 05:39 PM - akr (Akira Tanaka)

- Status changed from Feedback to Closed

- % Done changed from 0 to 100

Applied in changeset r45144.

- lib/resolv.rb (bind_random_port): Rescue EPERM for FreeBSD which security.mac.portacl.port_high is changed. See mac_portacl(4) for details. Reported by Jakub Szafranski. [ruby-core:60917] [Bug #9544]

#9 - 02/22/2014 06:05 PM - akr (Akira Tanaka)

- Status changed from Closed to Feedback

If system's resolver, getaddrinfo(), is usable, you can use it.

#10 - 02/23/2014 01:29 AM - akr (Akira Tanaka)

- Status changed from Feedback to Closed

#11 - 02/24/2014 04:36 AM - shyouhei (Shyouhei Urabe)

Jakub Szafranski wrote:

> So is it ruby's concern, or should the system developers make appropriate patches?

It's not either-us-or-them problem. I strongly agree OS devs should take care,
but that doesn't always mean we shouldn't. We are portable project. We cant
ignore that wild OS out there now needs workarounds in our side.

#12 - 03/15/2014 08:17 PM - samu (Jakub Szafranski)

Just a little follow-up:

I still think that OS should take care of that, this is a wrong layer for me.

For instance: if a system is trying to choose a free port - either random or sequential - it will randomize from a pool of *free* ports. You do realize that at high traffic, the probability of ruby hitting into a free port is significantly lower? Also, if port_min/port_max has been altered (which isn't really any magic), that chance gets really dropped. Aren't you bothered by performance issues that your workaround introduced?

#13 - 07/02/2014 06:13 AM - usa (Usaku NAKAMURA)

- Backport changed from 1.9.3: UNKNOWN, 2.0.0: UNKNOWN, 2.1: UNKNOWN to 2.0.0: REQUIRED, 2.1: REQUIRED

#14 - 07/23/2014 01:44 PM - nagachika (Tomoyuki Chikanaga)

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

Backported into ruby_2_1 branch at r46909.

#15 - 08/31/2014 07:22 AM - usa (Usaku NAKAMURA)

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

backported into ruby_2_0_0 at r47335.