Ruby - Misc #20387

Meta-ticket for ASAN support

03/22/2024 01:58 AM - kjtsanaktsidis (KJ Tsanaktsidis)

Status:	Assigned
Priority:	Normal
Assignee:	kjtsanaktsidis (KJ Tsanaktsidis)

Description

I was asked to provide a bit of information about the current status of ASAN in CRuby, so I thought I'd open this meta-ticket to track all of the work I've been performing on fixing up address sanitizer support.

So far, I have fixed the following issues related to ASAN support:

- https://bugs.ruby-lang.org/issues/20001 + https://github.com/ruby/ruby/pull/9505, which dealt with two main themes:
  - Pushing the logic for capturing the start of the machine stack much closer to the top of the call stack, so that VALUEs stored close to the top of the machine stack get marked properly
  - Marking VALUEs stored on ASAN fake stacks during machine stack marking
- https://bugs.ruby-lang.org/issues/20220 + https://github.com/ruby/ruby/pull/9734, which made M:N threading notify ASAN about stack switches in the same way that fibers do
  - Note: ASAN still doesn't work with M:N threading, but that actually has nothing to do with ASAN; it's because the most recent versions of Clang which are needed for ASAN just don't work with M:N threading either. See https://bugs.ruby-lang.org/issues/20243 for more info about that.
- https://bugs.ruby-lang.org/issues/20273 + https://github.com/ruby/ruby/pull/10012, which disables callcc (and the associated tests) when ASAN is enabled
  - callcc is very rarely used in real code and the way it works is just fundamentally incompatible with ASAN (it performs longjmp's which I think are technically undefined behaviour)
- https://bugs.ruby-lang.org/issues/20221 + https://github.com/ruby/ruby/pull/9865, which ignore some global symbols that ASAN defines from the global symbol leak checker
- https://bugs.ruby-lang.org/issues/20274 + https://github.com/ruby/ruby/pull/10087, which ignores some false positive tests about memory leaks when ASAN is enabl
- I updated the ASAN docs in https://github.com/ruby/ruby/pull/9922 to more closely reflect current reality

The current state of things is that, by following the instructions in https://github.com/ruby/ruby/blob/master/doc/contributing/building_ruby.md, you can successfully build Ruby with ASAN enabled, however, the test suite has several failures. I'm currently working on addressing these:

The next step is to merge https://github.com/ruby/ruby/pull/10122 (https://bugs.ruby-lang.org/issues/20310) which I plan to do next week (I'm currently away on a work trip). That makes sure that VALUEs stored in ASAN fake stacks from threads other than the currently running thread get marked during GC.

After that, I need to push up patches for the remaining few issues. I mostly have these patches ready to go already; in fact, last week I got the full make check suite passing all tests with ASAN enabled!

Once that's working, I'd like to investigate how ASAN can fit into CRuby's CI matrix somewhere so that it *stays* working, although I have not thought too deeply about this yet.

I will provide further updates on this ticket so anybody interested can stay in the loop.

Related issues:

Related to Ruby - Bug #20001: Make Ruby work properly with ASAN enabled	**Closed**
Related to Ruby - Bug #20220: M:N threading needs to tell ASAN about stack sw...	**Closed**
Related to Ruby - Feature #20273: Disable callcc when compiled with ASAN	**Closed**
Related to Ruby - Bug #20221: ASAN: make test-basic: un-prefixed symbol leakage	**Closed**
Related to Ruby - Feature #20274: Add RubyVM::ASAN.enabled?	**Closed**
Related to Ruby - Bug #20243: M:N threading VM_ASSERT failure in rb_current_e...	**Open**
Related to Ruby - Bug #20310: ASAN fake stacks need to be marked during GC fo...	**Closed**
Related to Ruby - Bug #20398: heap-buffer-overflow in numeric literal parsing	**Closed**
Related to Ruby - Bug #20402: Double-free in TestIseqLoad#test_stressful_roun...	**Closed**

History

#1 - 03/22/2024 01:59 AM - kjtsanaktsidis (KJ Tsanaktsidis)

- Related to Bug #20001: Make Ruby work properly with ASAN enabled added

#2 - 03/22/2024 01:59 AM - kjtsanaktsidis (KJ Tsanaktsidis)

- Related to Bug #20220: M:N threading needs to tell ASAN about stack switches added

#3 - 03/22/2024 01:59 AM - kjtsanaktsidis (KJ Tsanaktsidis)

- Related to Feature #20273: Disable callcc when compiled with ASAN added

#4 - 03/22/2024 01:59 AM - kjtsanaktsidis (KJ Tsanaktsidis)

- Related to Bug #20221: ASAN: make test-basic: un-prefixed symbol leakage added

#5 - 03/22/2024 02:00 AM - kjtsanaktsidis (KJ Tsanaktsidis)

- Related to Feature #20274: Add RubyVM::ASAN.enabled? added

#6 - 03/22/2024 02:00 AM - kjtsanaktsidis (KJ Tsanaktsidis)

- Related to Bug #20243: M:N threading VM_ASSERT failure in rb_current_execution_context with clang 17 (on Linux) added

#7 - 03/22/2024 02:00 AM - kjtsanaktsidis (KJ Tsanaktsidis)

- Related to Bug #20310: ASAN fake stacks need to be marked during GC for non-current execution context added

#8 - 03/27/2024 10:50 PM - kjtsanaktsidis (KJ Tsanaktsidis)

I merged another few PR's related to unit tests in ASAN builds:

https://github.com/ruby/ruby/pull/10383
https://github.com/ruby/ruby/pull/10384
https://github.com/ruby/ruby/pull/10385
https://github.com/ruby/ruby/pull/10386

#9 - 03/28/2024 04:55 AM - kjtsanaktsidis (KJ Tsanaktsidis)

- Related to Bug #20398: heap-buffer-overflow in numeric literal parsing added

#10 - 03/30/2024 02:02 AM - kjtsanaktsidis (KJ Tsanaktsidis)

- Related to Bug #20402: Double-free in TestIseqLoad#test_stressful_roundtrip added

#11 - 03/31/2024 09:48 AM - kjtsanaktsidis (KJ Tsanaktsidis)

OK! Everything required to make make check pass on my machine with ASAN enabled has been merged! I opened https://github.com/ruby/ruby/pull/10412 to update the documentation to include the correct incantation.

The next things I would like to do:

- Find some gem maintainers interested in running their tests under ASAN to see if it shakes out anything interesting
- Could we get an ASAN build of ruby into setup-ruby? This would make it easier to use ASAN with github actions and the like.
- How could we get ASAN plugged into ruby-ci?

I'm going to investigate these things and open tickets as appropriate once I have an idea how they should be tackled.

#12 - 04/01/2024 01:20 PM - Eregon (Benoit Daloze)

kjtsanaktsidis (KJ Tsanaktsidis) wrote in #note-11:

> - Could we get an ASAN build of ruby into setup-ruby? This would make it easier to use ASAN with github actions and the like.

How about making ASAN enabled for ruby-debug builds at
https://github.com/ruby/ruby-dev-builder/blob/b0bf59a17c17985d4692243d4689c273f6348fa5/.github/workflows/build.yml#L126-L130 ?

#13 - 04/01/2024 10:17 PM - kjtsanaktsidis (KJ Tsanaktsidis)

> How about making ASAN enabled for ruby-debug builds

That's not a bad idea (avoid adding yet more combinations!) but do you know how widely these builds are currently used and what for? ASAN will

make them a heck of a lot slower so that might be undesirable for some uses.

#14 - 04/02/2024 02:35 PM - Eregon (Benoit Daloze)

kjtsanaktsidis (KJ Tsanaktsidis) wrote in #note-13:

> That's not a bad idea (avoid adding yet more combinations!) but do you know how widely these builds are currently used and what for? ASAN will make them a heck of a lot slower so that might be undesirable for some uses.

They are builds using -DRUBY_DEBUG=1 and optflags=-O3 -fno-inline" so I would think already quite a bit slower, so it sounds OK to me.
These ruby-debug builds are currently meant to help CRuby development, i.e., using them on your gem helps finds bugs in CRuby. I don't think there are used very often.
I think the use-case here is close enough and I am not so keen on adding a 3rd kind of CRuby dev build.

#15 - 04/03/2024 03:50 AM - hsbt (Hiroshi SHIBATA)

- Status changed from Open to Assigned

#16 - 05/23/2024 06:46 AM - kjtsanaktsidis (KJ Tsanaktsidis)

So I've been thinking about how I'm going to do the setup-ruby thing this week. *Technically* speaking, when using ASAN, you should compile all of your dependencies with ASAN as well. If you don't do this, ASAN might miss crashes where you pass an invalid pointer to a shared library, and *it* dereferences that.

The mechanics of doing this are kind of tricky though. Ruby's dependencies (and those of the default gems) dependencies are, I think, openssl, readline, ncurses/terminfo (through readline), libyaml, libffi, and gmdb. I can think of three main families of approaches, but I'm sure there are more variations.

1. Hack up Ruby's build system to build in-tree versions of all of these things and statically link against them (or install them to /usr/lib/ruby and put that in RPATH or something).
2. Declare /opt/asan or something as a home directory for ASAN libs. Put a bunch of ./configure CFLAGS="-fsanitize-address" --prefix=/opt/asan && make && make install in the setup-ruby github actions workflow.
3. Make a Ubuntu ppa archive, containing debs of libraries built with ASAN, and installed into /opt/asan. Then, the setup-ruby github actions workflow can just do apt-add-repository ppa:asan-libs && apt-get install libyaml-asan openssl-asan .... And I guess I become a package maintainer for the debs in the PPA (and bump them when they have releases, etc).

I'm honestly leaning towards doing 3 here. It's the most work, but Launchpad at least does have some tools for automating package builds when an upstream changes. And it means that if e.g. the mysql2 gem wants to test with ASAN, I can add a libmysqlclient-asan package to the PPA and then they can just use that, rather than having to build libmysqlclient in their github actions config.

I would like someone else's opinion on the subject though - what do you think @Eregon (Benoit Daloze)? Thanks!

EDIT: You could also convince me I should make a package archive for a different distro, not Ubuntu, but it seems to broadly be "the default" for github actions which is where most testing of Ruby extension gems probably happens these days.

#17 - 05/23/2024 10:15 AM - Eregon (Benoit Daloze)

> Technically speaking, when using ASAN, you should compile all of your dependencies with ASAN as well. If you don't do this, ASAN might miss crashes where you pass an invalid pointer to a shared library, and it dereferences that.

I am no ASAN expert but I think that might be overkill, especially initially. I.e. I think adding ASAN there would find very few issues, and OTOH compiling CRuby itself with ASAN would find most issues.
Of course C extensions from gems should also be compiled with ASAN (does that already work automatically?).

So my suggestion would be to either alter the ruby-debug build or add a new variant, simply setting the ASAN flags when building ruby. I believe that should work well and keep things simple.

The package approach sounds fine, but it sounds to me like high effort for potentially very little reward. It would also be slower to install (in the context of setup-ruby) since apt is slow (compared to download archive + extract).

#18 - 05/23/2024 10:20 AM - Eregon (Benoit Daloze)

Some more thoughts:

- How do other projects use ASAN? They must have similar situations regarding dependencies compiled or not with ASAN.
- Gentoo IIRC has this feature to easily set flags/features globally for all installed packages. Maybe a prebuilt docker image of Gentoo with CRuby + all other packages compiled with ASAN would be relatively easy to try? Less practical to run in GitHub Actions though.

#19 - 06/02/2024 12:56 AM - kjtsanaktsidis (KJ Tsanaktsidis)

Having spent a bit of time on this yesterday - I think there's some value in judiciously using ASAN versions *some* dependencies, but not the whole

tree. I think the best approach for these dependencies is simply to statically link them; that way, the built Ruby will be runnable exactly as it is now (no need to apt install special asan'd packages for consumers of setup-ruby).

I opened a PR in https://github.com/ruby/ruby-dev-builder/pull/10 to compile Ruby with ASAN, and selected these three dependencies to also compile with ASAN:

- OpenSSL - because we use it in non-GVL contexts to perform IO, so we want to catch if any buffers are GC'd and poisoned while OpenSSL is still using them
- libffi - because the whole point of it is to make arbitrary calls to C (in fact, pure-ruby code using fiddle or the ffi gem might even benefit from an ASAN'd libffi to find mistakes in Ruby code)
- libyaml - because it's often used to handle untrusted data and it's pretty small.

That seemed like a good balance to me. I've already found a bug in the OpenSSL gem's test suite because of this: https://github.com/ruby/openssl/pull/762

> Of course C extensions from gems should also be compiled with ASAN (does that already work automatically?).

It should do yes, because the CFLAGS/CC get saved in rbconfig (but i'm going to do some testing with that ruby-dev-builder branch to make sure that works in the way I expect

> How do other projects use ASAN? They must have similar situations regarding dependencies compiled or not with ASAN.

I really did try and look at how Firefox, Chromium and LLVM configure their build environments. But it was *wickedly* complicated and opaque. It doesn't *look* to me like they compile with ASAN'd dependencies, but i'm not really sure.