Memory Safety SIG

Antitrust Policy Notice

Linux Foundation meetings involve participation by industry competitors, and it is the intention of the Linux Foundation to conduct all of its activities in accordance with applicable antitrust and competition laws. It is therefore extremely important that attendees adhere to meeting agendas, and be aware of, and not participate in, any activities that are prohibited under applicable US state, federal or foreign antitrust and competition laws. Examples of types of actions that are prohibited at Linux Foundation meetings and in connection with Linux Foundation activities are described in the Linux Foundation Antitrust Policy available at http://www.linuxfoundation.org/antitrust-policy. If you have questions about these matters, please contact your company counsel, or if you are a member of the Linux Foundation, feel free to contact Andrew Updegrove of the firm of Gesmer Updegrove LLP, which provides legal counsel to the Linux Foundation.

Code of Conduct

All OpenSSF Meetings must comply with the Code of Conduct:
https://openssf.org/community/code-of-conduct/

This group will propose to be part of the OpenSSF Best Practices WG.

2023-05-25

Attendees:

- Avishay Balter (Microsoft)
- Gabriel Dos Reis (Microsoft)
- Mikey Strauss (Scribe)
- CRob (Intel, TAC)
- David A. Wheeler (Linux Foundation)

Agenda:

- Welcome new friends
- Help Scribe
- Discuss/approve drafts?
- We can walk through some open PRs https://github.com/ossf/Memory-Safety/pulls
    - PR#4: https://github.com/ossf/Memory-Safety/pull/4 - WG decision today: Fix the markdown lint problems, then please merge. We'll track the TODOs as issues to

close them out. David A. Wheeler will help with 3 (C/C++), CRob will help with 4 (education).

- David A. Wheeler: Question: Are techniques to improve memory safety in C/C++ in scope?
    - The Linux kernel is taking steps to counter *some* memory safety issues, e.g., Kees Cook “Bounded Flexible Arrays in C” https://people.kernel.org/kees/bounded-flexible-arrays-in-c. These can’t provide the same level of memory safety guarantee, as you can still easily write non-memory-safe code. But by applying certain practices you can counter specific kinds of memory safety problems (like buffer overflows). It’s okay if this is out-of-scope… it’s just best to make it clear what the group’s intended scope is.
    - See also: OpenSSF compiler flags work
    - See also: https://discourse.llvm.org/t/rfc-enforcing-bounds-safety-in-c-fbounds-safety/70854
    - Boost C++ library - safe pointers, etc. Warning: some organizations prohibit the use of Boost, primarily because it’s a huge monolith, some parts are actively maintained while others are vestigial and aren’t maintained. For example, some parts that are now obsolete as their functionality is part of C++ itself. See: https://www.reddit.com/r/cpp/comments/gfowpq/why_you_dont_use_boost/
        - David A. Wheeler: Historically LibreOffice had many different C++ types for strings, which created a huge overhead & made maintenance hard. Reducing library usage can help improve maintenance by simplifying the code. In short, a library to implement a function can be helpful, but having *multiple* libraries that implement the same functionality within an application can be a real problem.
    - Note: many items are dependent on specific hardware or specific compilers.
    - Yes, this is in scope.
    - Basically, this WG would create guide (or spec) for C/C++ to reduce the likelihood of memory safety problems in real-world applications. It’d note the above.
    - Maybe separate C from C++ guidance/spec.
-

2023-04-27

Attendees:

- Nell Shamrell-Harrington (she/they, Microsoft, Rust Foundation)
- Gabriel Dos Reis (Microsoft)
- Jay White (Microsoft)
- Avishay Balter (Microsoft)
- David A. Wheeler (Linux Foundation)
- CRob (Intel)
- Charles C Palmer (IBM Research and Dartmouth)
- Matt Rutkowski (IBM)

- Randall T. Vasquez (LF/SKF/Gentoo)
- Jeff Borek (IBM)

Agenda

- Welcome new friends
- Scribe
- Discussion on revised wording of mobilization plan stream #4
    - Replace the wording for “known… vulnerabilities” to “classes of vulnerabilities”
    - Remove “known” as it may suggest purposeful action from developers
    - Should we add memory leaks
        - Java and C# can memory leak, but not in the same way as c/c++
        - However, memory leaks are an issue for some of the WG’s audience
    - Add link to NSA’s document on software memory safety
        - Switching to memory safe by default language, when possible
    - The emphasis should be what the language brings in terms of prevention, as opposed to name and blame specific technologies
    - Should the document reference specific tool or tool version or would that be too technical for the audience?
        - CRob suggests Mix people, process and tools
    - Discussion on declining suggestion for edit observing vulnerabilities in dependant code, since that relates more to SBOM/VEX and is handled by other WG/SIGs
- We want to have tooling that can prove memory safety in code, for instance in Scorecard.
    - Early discussion with Scorecard did not yet have a vision as to how that would look like in a Scorecard check. We should investigate on that more before reaching out again to the SC team
- We should expand the plan’s new items (3-5) and not only the original two points that were covered in the mobilization plan (MOVING CRITICAL SOFTWARE INFRASTRUCTURE TO MEMORY SAFE BY DEFAULT LANGUAGES and INVESTING IN SAFER SYSTEMS DEVELOPMENT TOOLS)
- We want to engage with other standard groups such as sigstore and Prossimo, through Josh to create an official liaison between the groups.
    - https://www.memorysafety.org/
    - We can work together with them, publish blogs together.
    - Nell will reach out to Josh to setup this allyship between the SIG and the foundation’s work.
- [joke] Propose to have tin foil hat as the official logo and have stickers with that image for all! [/joke]
- Next steps
    - Put together another draft.
    - Review the new draft revision.
    - Identity specific streams to work on and get funding on from the foundation.
    - Convert from GDoc to MD.

2023-04-13

- Nell Shamrell-Harrington (she/they, Microsoft, Rust Foundation)
- Gabriel Dos Reis (Microsoft)
- Jay White (Microsoft)
- Avishay Balter (Microsoft)
- David A. Wheeler (Linux Foundation)
- CRob (Intel)

Agenda

- https://github.com/ossf/Memory-Safety/pull/1
    - Jordan is working to clean up GitHub permissions. In general we want to form teams, and then assign permissions to teams
- https://docs.google.com/document/d/1e7-fUaPl1-uhXptsyD92MACvureXbMIUcmcD-ZyOd6M/edit
- Group agreed this initial SIG (co) leadership:
    - Nell Shamrell-Harrington (she/they, Microsoft, Rust Foundation) - @nellshamrell
    - Avishay Balter (Microsoft) - @balteravishay
- How do we define “memory safe language”
    -
    - Key issue: “by default”.
    - We should try to reuse existing definitions.
    - https://www.memorysafety.org/docs/memory-safety/ - Memory safety is a property of some programming languages that prevents programmers from introducing certain types of bugs related to how memory is used. Since memory safety bugs are often security issues, memory safe languages are more secure than languages that are not memory safe.
    - Rina Diane Caballar definition (IEEE): “Memory safety is a feature of programming languages that prevents certain types of memory-access bugs, such as out-of-bounds reads and writes, and use-after-free bugs. In an app that manages a list of to-do items, for example, an out-of-bounds read could involve accessing the nonexistent sixth item in a list of five, while a use-after-free bug could involve accessing one of the items on an already deleted to-do list. These bugs could lead to accessing private data, corrupting data, or even executing code that isn’t part of a program.” https://spectrum.ieee.org/memory-safe-programming-languages
    - Most languages are memory safe. Languages that are *not* memory-safe include C, standard C++, Objective-C, Vala, Forth, Assembly, Machine language,.
    - Memory-safe languages include Rust, Go, Ada, Python, JavaScript, TypeScript, Ruby, and many others.

    - There's a subset of C++ that can be enforced by rules that counters memory safety problems. E.g., no new/delete, can't access beyond memory bounds, etc.
    - Gdr will work out a definition of "memory safe language"
- Also: Quote the various citations for %s of vulnerabilities due to memory safety
- We won't be able to convert EVERYTHING to memory safe languages in any near term. However, we can encourage using them, as they eliminate problems at their source
- If you keep using C or C++, the compiler options work will help to counter some memory safety issues.
- Linux kernel experience: They're adding Rust for device drivers. In practice, they're finding they're changing their C interfaces to make them safer for all, e.g., preferring interfaces that are read-only. Linux is also changing how they use C to reduce the likelihood of memory safety problems, which is different but helpful.
- It's misleading to say "safe" languages - say "memory safe"
-

2023-03-30

Attendees:

-

- Nell Shamrell-Harrington (she/they, Microsoft, Rust Foundation)
- Gabriel Dos Reis (Microsoft)
- Christine Abernathy (F5)
- Josh Aas (he/him, ISRG/Prossimo)
- CRob (he/him, Intel/OSSF)
- Jonathan Leitschuh (he/him) OpenSSF
- Jay White (Microsoft)
- Randall T. Vasquez (SKF/Gentoo)
- Walter Pearce (he/him, Rust Foundation Security Engineer)
- Avishay Balter (Microsoft)

Agenda

- We are officially part of the Developer Best Practices SIG!
- We have a shiny new Git Repo! Let's plan how we fill this out!
    - https://github.com/ossf/Memory-Safety
    - Filling out Nell, Walter, Avishay
- Ideas for first project
    - Rewrite memory safety language in OSS Mobilization Plan https://8112310.fs1.hubspotusercontent-na1.net/hubfs/8112310/OpenSSF/OSS%20Mobilization%20Plan.pdf?hsCtaTracking=3b79d59d-e8d3-4c69-a67b-6b87b325313c%7C7a1a8b01-65ae-4bac-b97c-071dac09a2d8
    - Google doc - start with commenting?
        - Will create one after the meeting

- https://docs.google.com/document/d/1e7-fUaPl1-uhXptsyD92MACvureXbMIUcmcD-ZyOd6M/edit

2023-03-02

Attendees:

- Nell Shamrell-Harrington (she/they, Microsoft, Rust Foundation)
- David A. Wheeler (Linux Foundation)
- Josh Aas (ISRG)
- Art Manion (self, contractor to but do not speak for CISA))
- Walter Pearce - Rust Foundation, focused on security
- Daniel Frampton (Microsoft)
- Randall T. Vasquez (SKF/Gentoo)
- Avishay Balter (Microsoft)
- Jay White (Microsoft)
-

Agenda

- Further define goals as a SIG
    - This is not just a Rust group. The goal is to encourage memory safety. It includes how to make C/C++ better, since we can't rewrite everything instantly.
    - We need to crisply define goal. Maybe start with mobilization plan.
    - The purpose of memory safety SIG is to encourage the transition to memory safe programming languages and approaches in open source software.
    - The purpose of memory safety SIG is to improve memory safety in OSS.
    - Develop and promote standards and guidelines for memory safe programming practices
    - Purpose: Eliminate memory safety vulnerabilities in OSS.
- Then define specific tasks to start, e.g., documents, programs, etc.
- Prep for presentation to Developer Best Practices WG [propose to join them?]

From https://openssf.org/oss-security-mobilization-plan/:
Stream 4: Eliminate Root Causes of Many Vulnerabilities Through Replacement of Non-Memory-Safe Languages

Discussion about how to do this:

- Understanding factors that contribute to
    - Memory safety problems
    - Improvements
- Select and rewrite software (completely or modules) - e.g., Prossimo
- Remove roadblocks
    - CPU support
    - Inadequate performance of some languages
    - Difficulty in connecting with existing code (to enable incremental changes)

- Understand and document practical places to move away from memory unsafe languages
- Guidance, standards, practices, tools, advocacy
- Highlight successes, particularly running code
- Do we include/address confidential computing? Possibly a different class/layer of memory security problems?

Discussion on prevalence of memory-safety issues:

- Microsoft ~70%
- Chrome ~70%
- https://advocacy.consumerreports.org/wp-content/uploads/2023/01/Memory-Safety-Convening-Report-1-1.pdf
    - Alex Gaynor
- Across all NVD, ~25% (Art's quick estimate with data with problems)
    - https://docs.google.com/spreadsheets/d/1LIT_KusTuQSOka73Z3WUHHY1O0kcUuCnItA8wkX22L4
    - https://noncombatant.org/2022/04/22/itw-taxonomy/

Vision: Eliminate memory safety vulnerabilities (in OSS).

Mission: Understand and reduce memory safety vulnerabilities in OSS.

Scope/Purpose: Develop pragmatic guidance, standards, and software (including tools, tool improvements, and rewrites), along with advocating such changes, to systematically reduce memory safety vulnerabilities through the use of memory-safe programming languages and techniques, all informed by real-world data and risks.

Will present this to the Best Practices WG to be accepted.

Next meeting, 4 weeks from today - March 30. Keep talking on Slack.

we are in the OpenSSF Slack stream #stream-04-memory-safety channel
https://join.slack.com/t/openssf/shared_invite/zt-1o4q7bq81-rZTF_3le_4B5wGWwvLYqoA