LLVM SECURITY GROUP

History, achievements, remaining challenges
FOSDEM 2024 – LLVM dev room
Kristof Beyls (@kbeyls)
You can reach me on LLVM Discourse/Discord/…

What is the LLVM security group? https://llvm.org/docs/Security.html

- Enable responsible disclosure of security issues related to LLVM projects
  - *Focus on security work that cannot immediately be done publicly.*
  - *Not focussed on other security-related things that can be done publicly like implementing improvements, new ideas, …*

History

November 2019: RFC Posted

July 2020: Security group formed; process still need to be developed

January 2021: group is fully operational

Today, early 2024: group has been operating for 3 years

- https://discourse.llvm.org/t/rfc-llvm-security-group-and-process/53707

Who is/can be on the group?

- Individual contributors
- Security Researchers
- Vendor Contacts

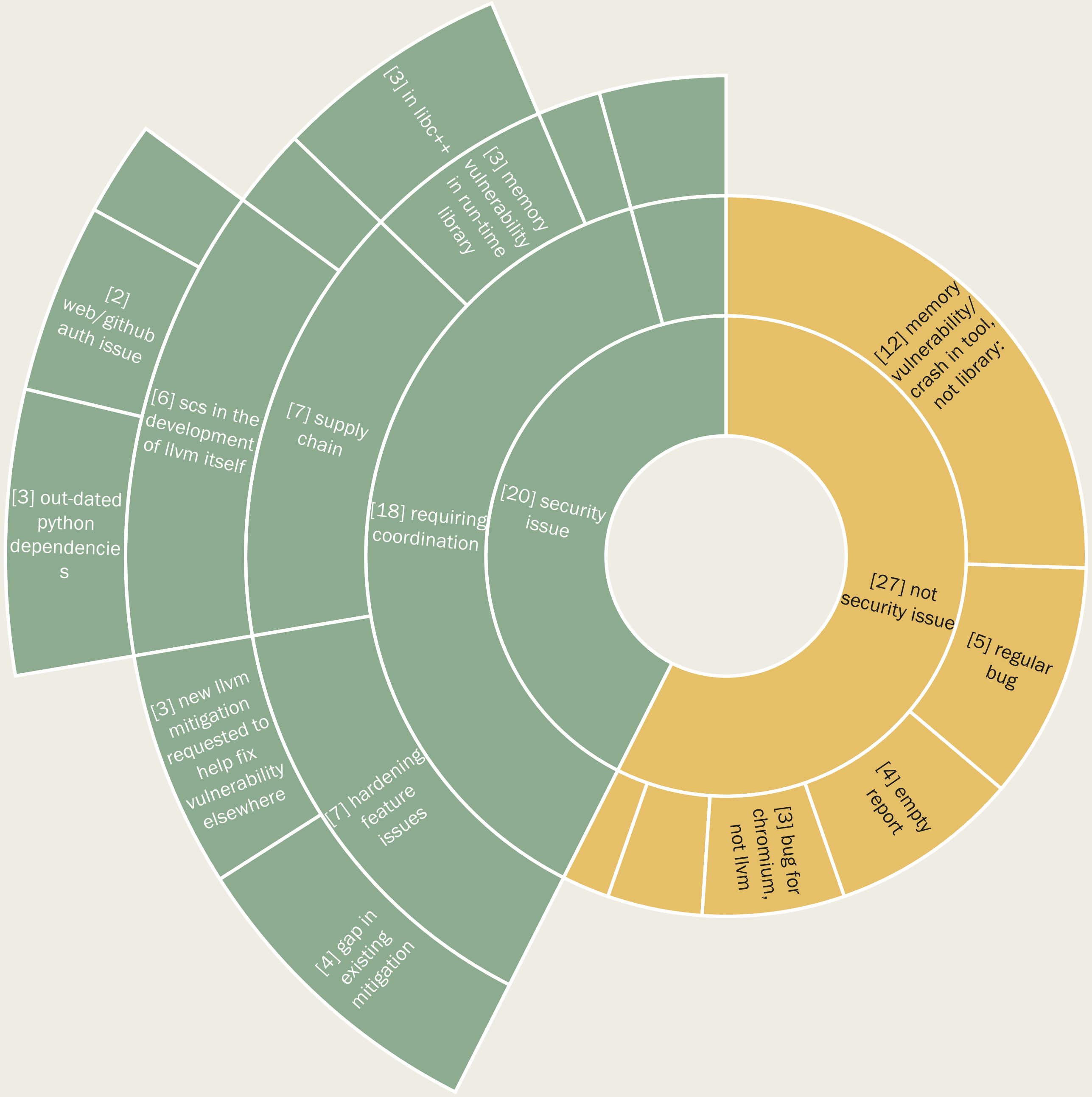
- Currently 20 members, mostly vendor contacts.

How to report security issues?

- Using the Chromium issue tracker, because it enables good access control to issues reported in confidence.
- Planning to soon move over to something different; most likely using github’s mechanism to report issues in confidence.
- Look out for an upcoming announcement/RFC.

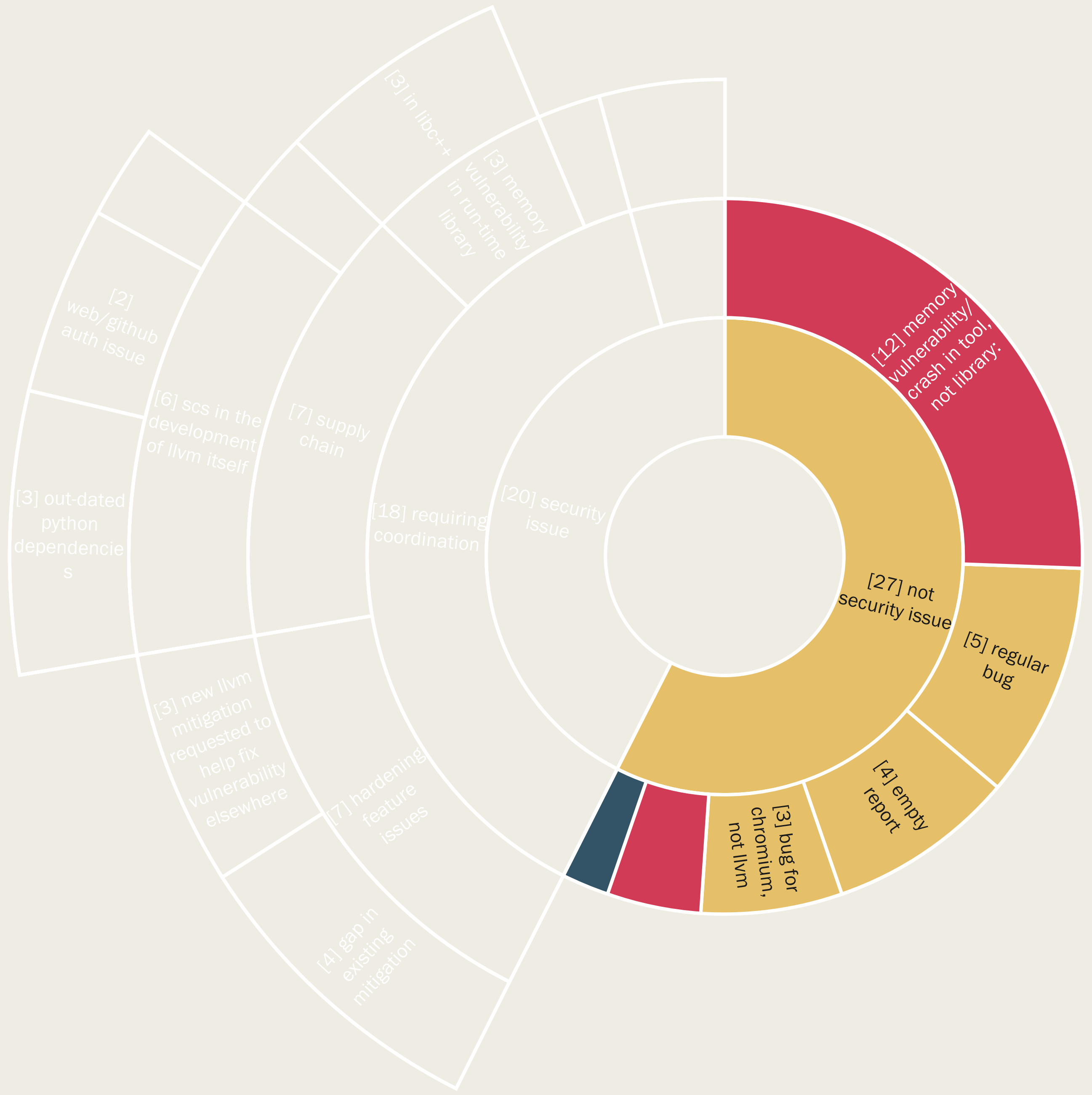
ANALYSIS OF REPORTS RECEIVED SO FAR

47 issues reported in 3 years: 2021-2023.



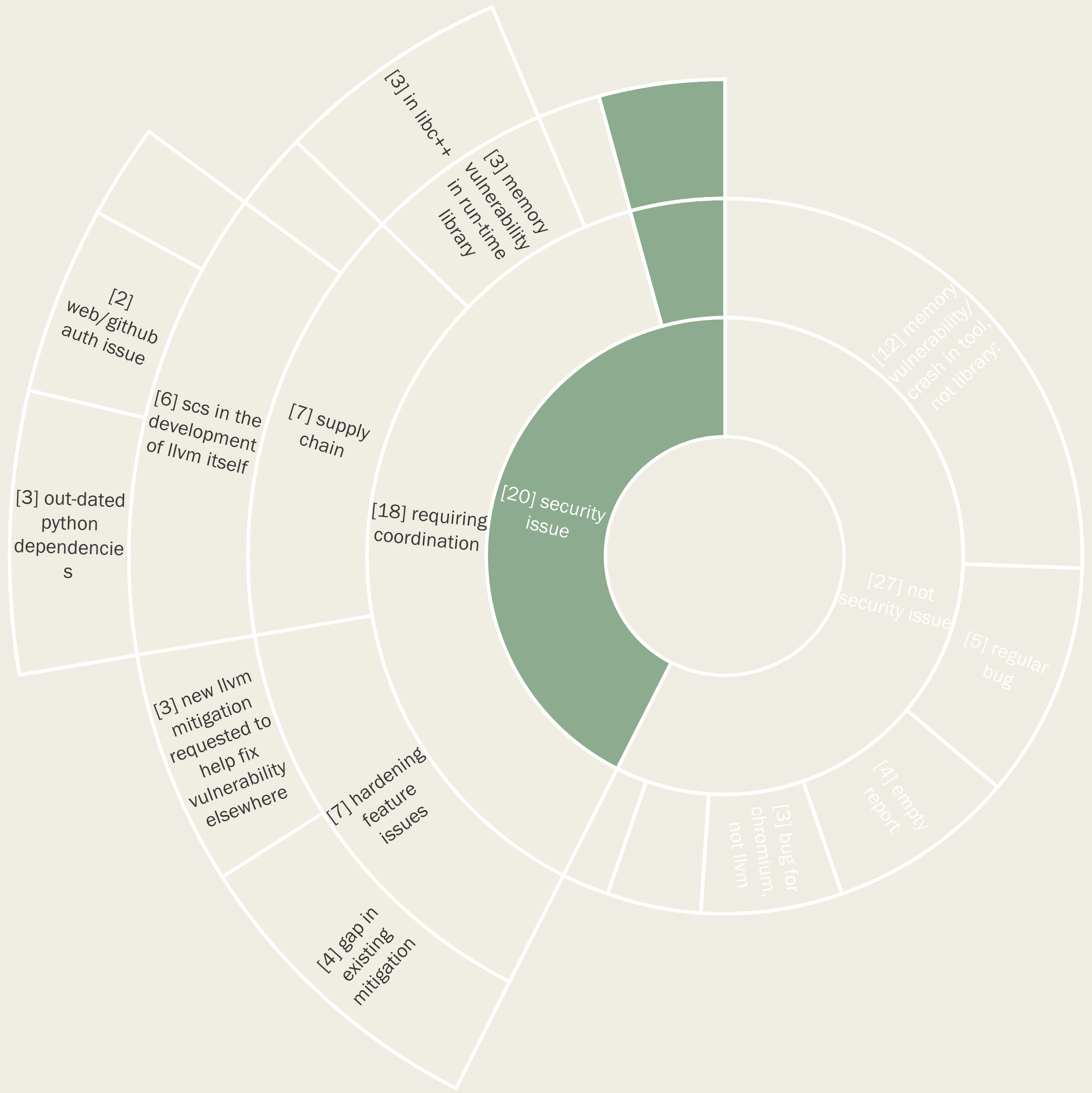
27 (57%) not deemed security issues:

- 4 empty reports
- 3 chromium issues
- 5 regular bugs
- 12 (26%) mem vulnerability in tool, not library
- 2 (4%) undefined behaviour in source code
- 1 discussion on improving supply chain security



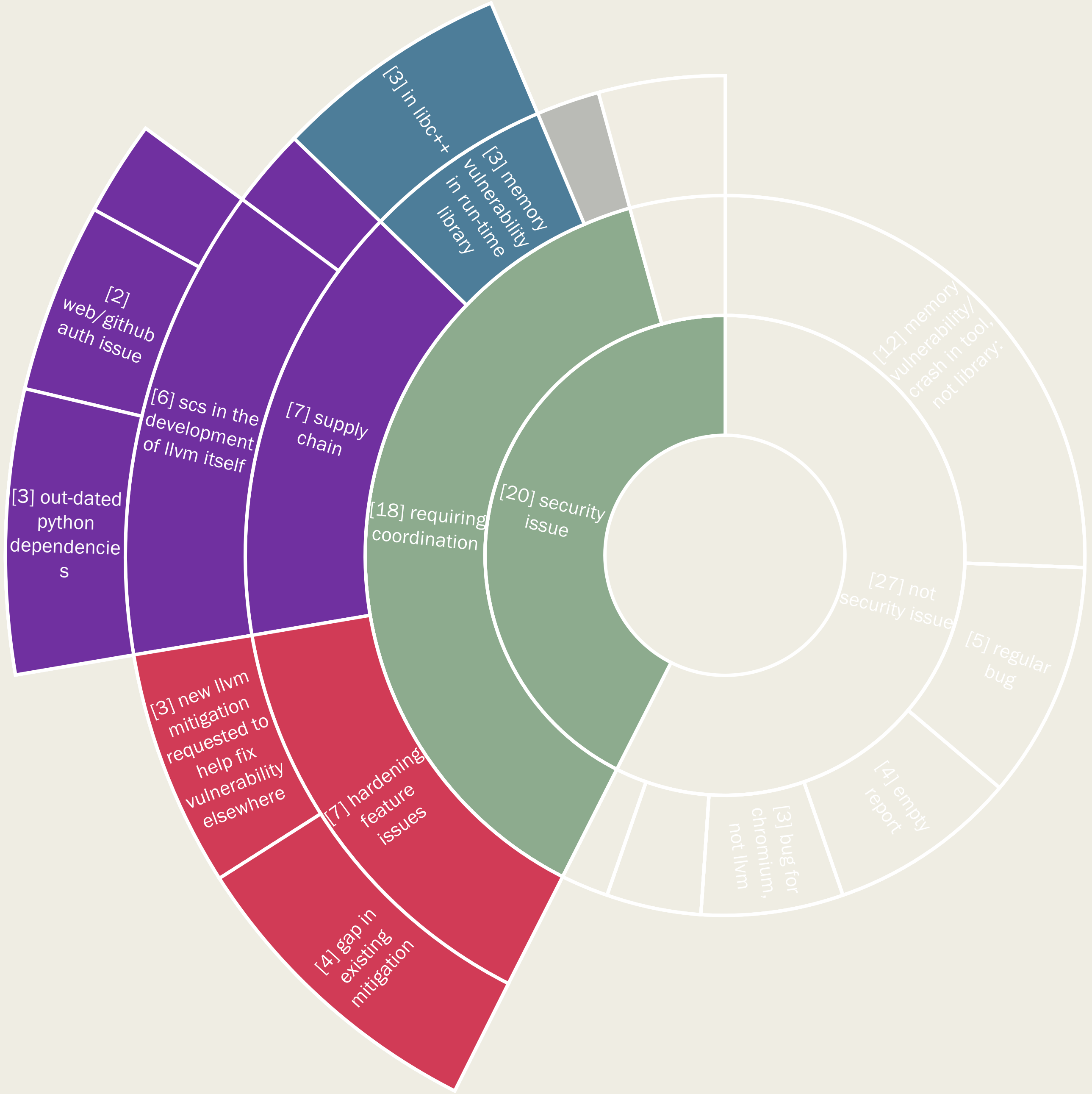
2 (5%) deemed security issues, but not requiring co-ordinated actions:

- 1 a sanitizer not reporting an issue
- 1 a clang warning not being enabled by default



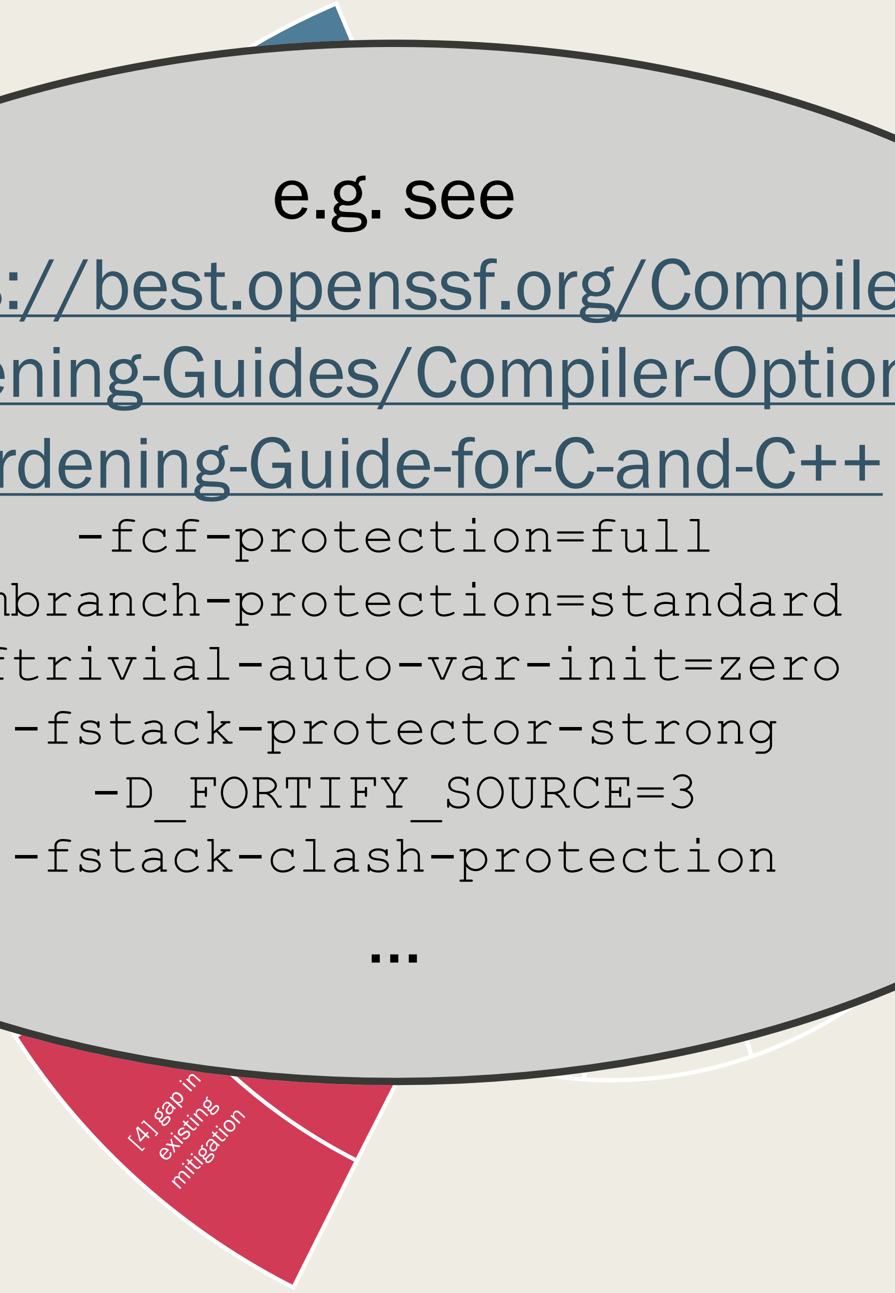
18 (38%) deemed security issues, requiring co-ordinated actions:

- 1 incorrect codegen
- 3 memory vuln in libc++
- 7 supply chain
- 7 gaps in hardening features



18 (38%) deemed security issues, requiring co-ordinated actions:

- 1 incorrect codegen
- 3 memory vuln libc++
- 7 supply chain
- 7 gaps in hardening features



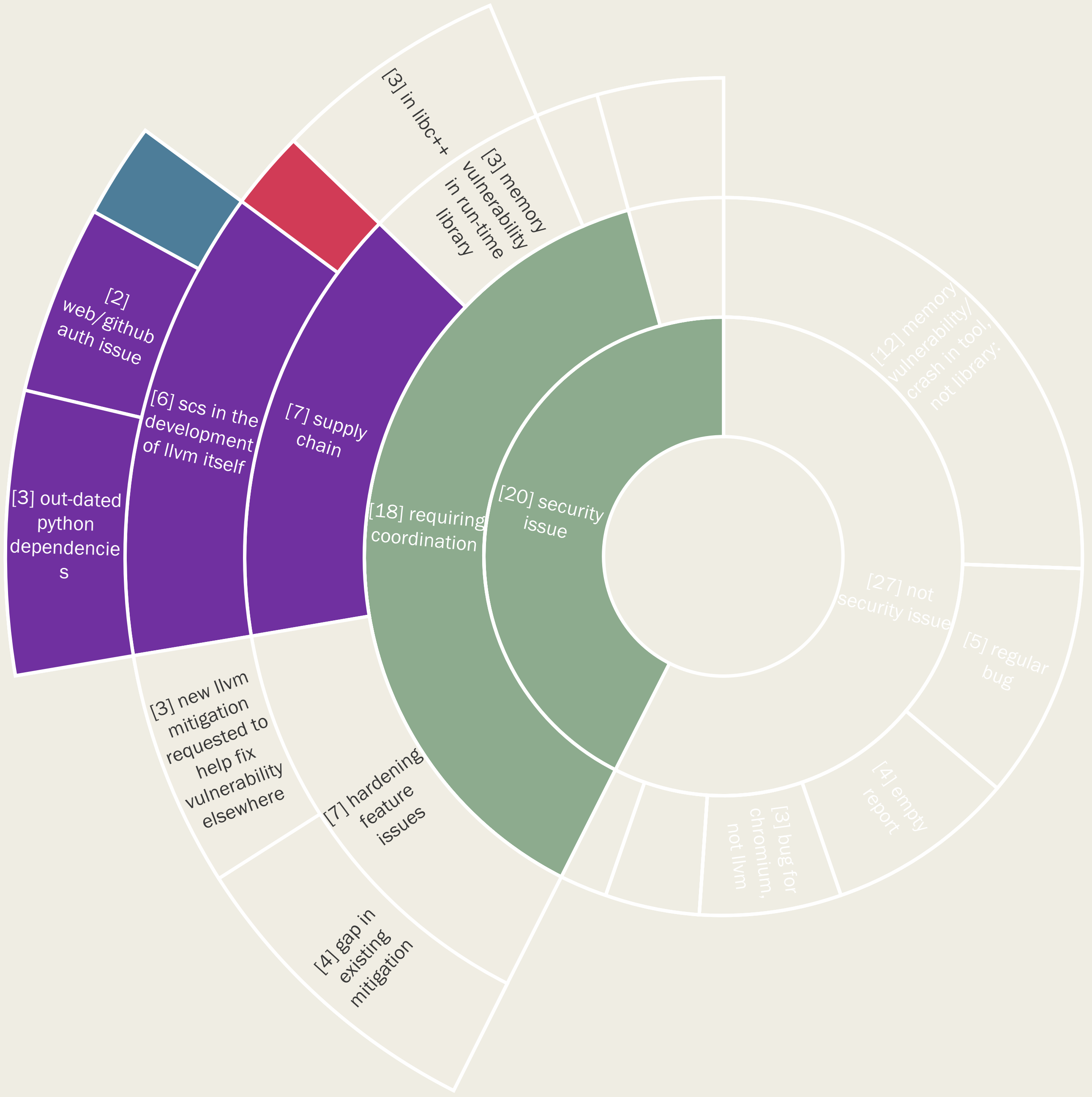
e.g. see https://best.openssf.org/Compiler-Hardening-Guides/Compiler-Options-Hardening-Guide-for-C-and-C++

```
-fcf-protection=full
-mbranch-protection=standard
-ftrivial-auto-var-init=zero
-fstack-protector-strong
-D_FORTIFY_SOURCE=3
-fstack-clash-protection
```

…

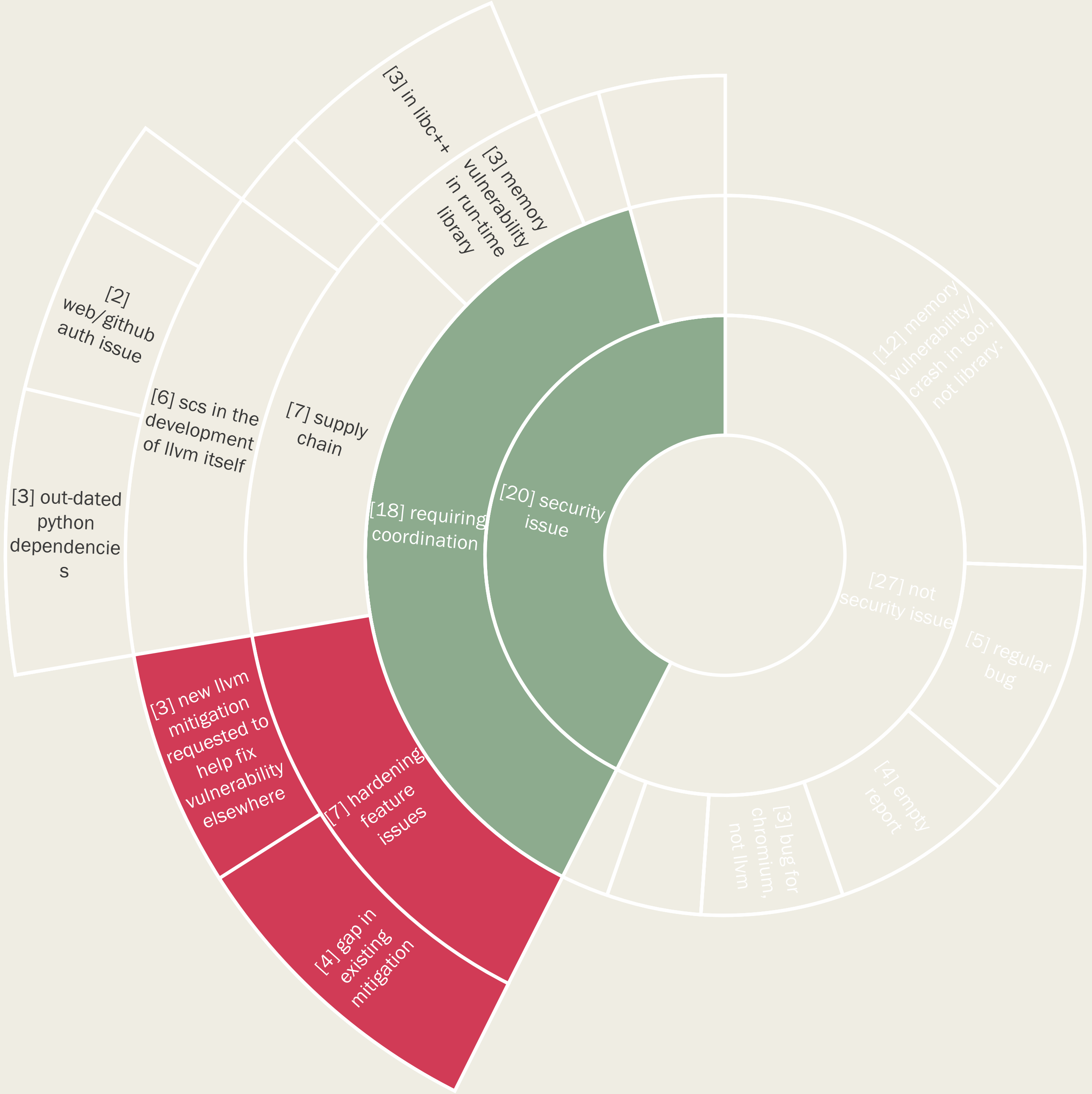
7 Supply chain issues requiring co-ordinated actions:

- 1 vscode clangd pot. trusting untrusted workspc
- 1 introducing back-door suspicion
- 2 github/website auth issue
- 3 out-dated python library dependencies



7 Hardening feature issues, requiring co-ordinated actions:

- 4x gaps in existing mitigations (e.g. CHOP, CFI, BTI)
- 3x request for new mitigation for vulnerability outside of LLVM (e.g. Retbleed, Ultimate SLH, Trojan Source)



Some take-aways from looking at stats: Achievements

- All reported issues seem to have been processed apropriately.

  => *The LLVM security group is working and adding value*

  => *The LLVM security group can be trusted to appropriately progress security issues to conclusion.*

- Yearly transparency reports
  https://llvm.org/docs/SecurityTransparencyReports.html

- Don’t know how many security issue were accidentally filed publicly instead of reporting to the security group…
  Feedback welcome on how to improve this.

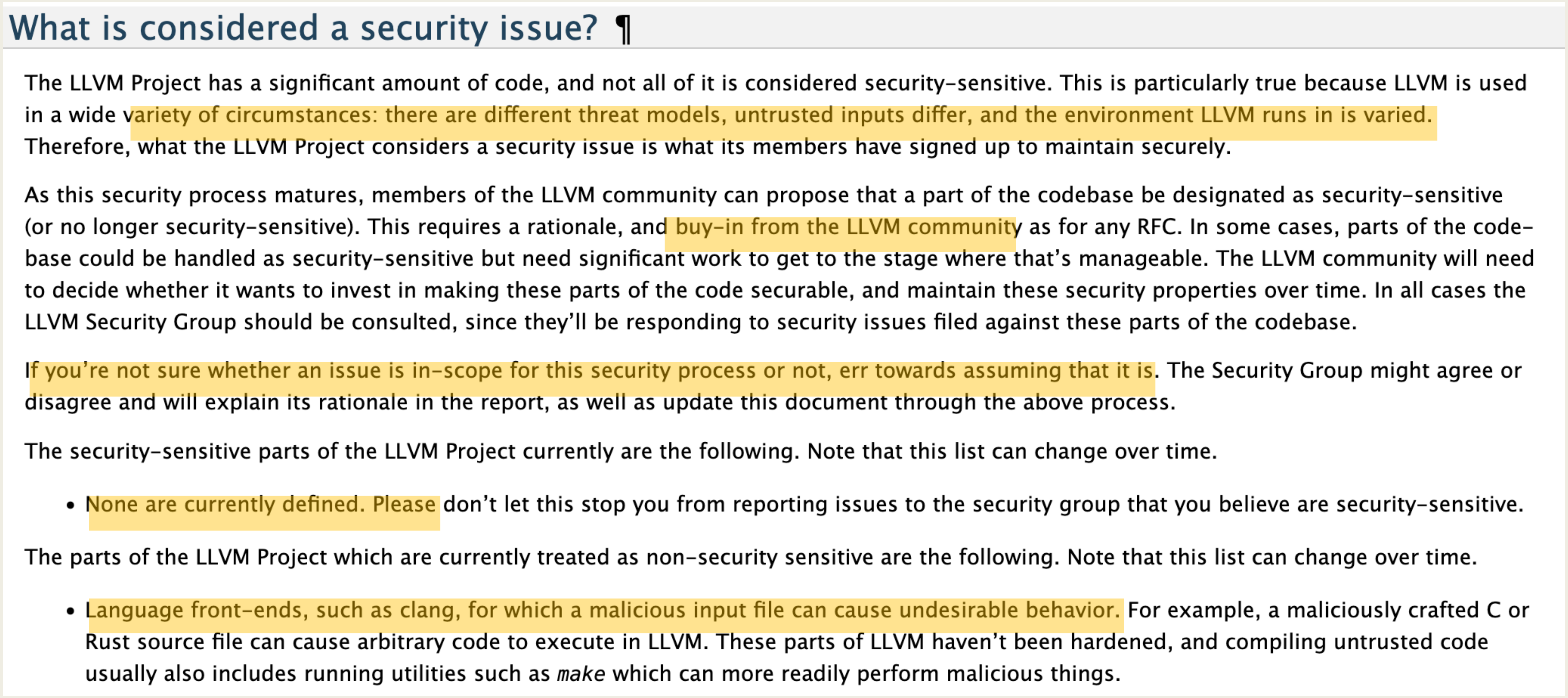
ROOM FOR IMPROVEMENT TOO

Areas for improvement

- More clarity on threat model/what is a security issue? (30% of all reports)
- Largest complexity: issues related mitigation implementations in code gen. (35% of security issues)
- Supply chain: few categories. (35% of security issues)
- Move away from chromium bug tracker (6% of all reports)
- How to communicate discovered security issues to people who need to know? (not easily seen from stats)

What is a security issue/threat model

https://llvm.org/docs/Security.html#what-is-considered-a-security-issue



What is considered a security issue? ¶

The LLVM Project has a significant amount of code, and not all of it is considered security-sensitive. This is particularly true because LLVM is used in a wide variety of circumstances: there are different threat models, untrusted inputs differ, and the environment LLVM runs in is varied. Therefore, what the LLVM Project considers a security issue is what its members have signed up to maintain securely.

As this security process matures, members of the LLVM community can propose that a part of the codebase be designated as security-sensitive (or no longer security-sensitive). This requires a rationale, and buy-in from the LLVM community as for any RFC. In some cases, parts of the code-base could be handled as security-sensitive but need significant work to get to the stage where that's manageable. The LLVM community will need to decide whether it wants to invest in making these parts of the code securable, and maintain these security properties over time. In all cases the LLVM Security Group should be consulted, since they'll be responding to security issues filed against these parts of the codebase.

If you're not sure whether an issue is in-scope for this security process or not, err towards assuming that it is. The Security Group might agree or disagree and will explain its rationale in the report, as well as update this document through the above process.

The security-sensitive parts of the LLVM Project currently are the following. Note that this list can change over time.

- None are currently defined. Please don't let this stop you from reporting issues to the security group that you believe are security-sensitive.

The parts of the LLVM Project which are currently treated as non-security sensitive are the following. Note that this list can change over time.

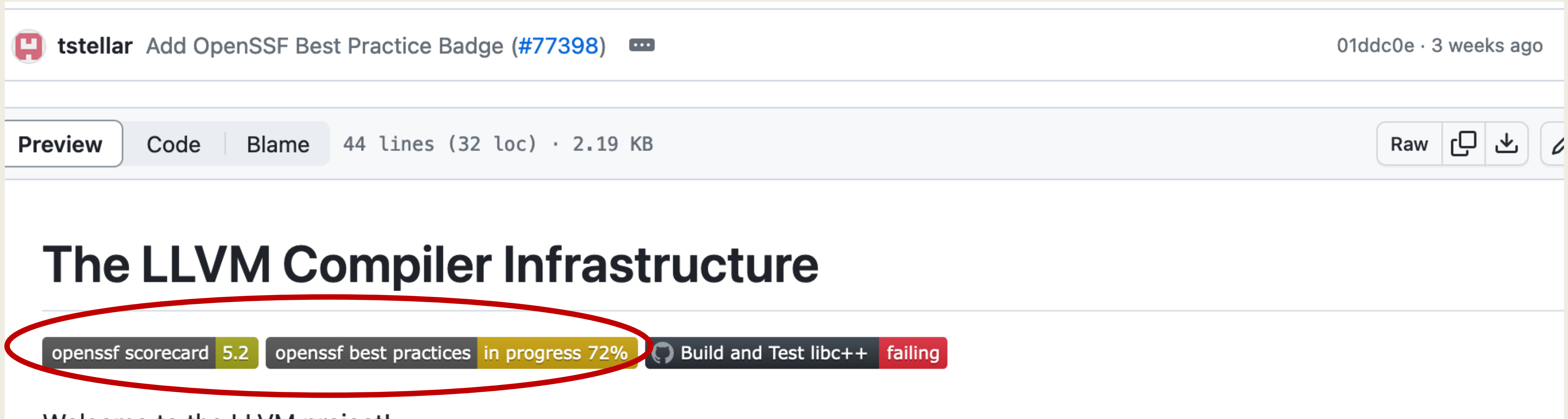
- Language front-ends, such as clang, for which a malicious input file can cause undesirable behavior. For example, a maliciously crafted C or Rust source file can cause arbitrary code to execute in LLVM. These parts of LLVM haven't been hardened, and compiling untrusted code usually also includes running utilities such as *make* which can more readily perform malicious things.

Can we improve on issues related to mitigation/hardening features?

- Better document what mitigations do exactly, so users know what protection they give exactly? (*2 or 3 reported issues?)*
- Increase quality-of-implementation of security hardening features. Ideas
  - *Building a binary scanner to verify correct application of security hardening. (I built a BOLT-based such scanner prototype, details for another presentation)*
  - *Improve design documentation and documentation of known gaps?*
  - *Help mitigations to become available to more LLVM-based languages (e.g. Rust?)*
  - *Help compiler developers know more about security and attacks, e.g. through https://llsoftsec.github.io/llsoftsecbook/ book?*

Supply chain

- A few categories:
  - *securing llvm.org web infrastructure.*
  - *Features in toolchain to help with developing software better from a supply chain security point-of-view.*
  - *Secure against malicious injection of code into llvm binaries.*



tstellar Add OpenSSF Best Practice Badge (#77398)

01ddc0e · 3 weeks ago

44 lines (32 loc) · 2.19 KB

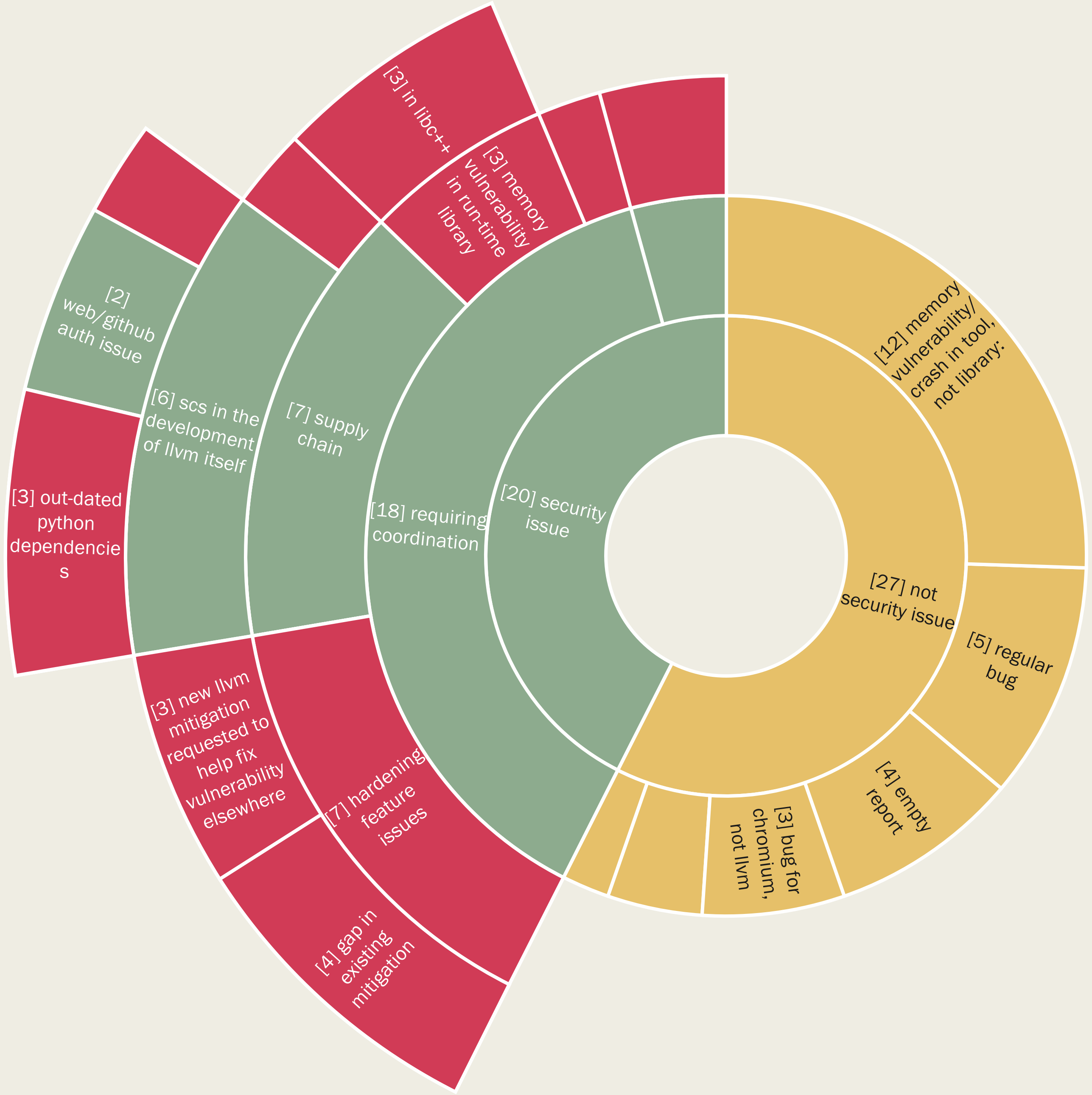
The LLVM Compiler Infrastructure

openssf scorecard 5.2 | openssf best practices in progress 72% | Build and Test libc++ failing

Move away from chromium tracker

- Expected to be happening in Q1 2024
- Probably moving to something based on github.

We should communicate about 18 out 20 issues, but none require CVE? So how?



Thoughts on better communicating security issues

- Most of these security issues aren’t worthy of a CVE (don’t leave a system immediately exploitable)
- Release notes don’t work well.
- Potential solutions:
  - *separate page on llvm.org documenting known public security issues.*
  - *Maybe use “security” label on github issues and that’s the way to publish known security issues?*
    Can interested people appropriately subscribe to changes there?
  - *Would need to document both known affected versions and fixed versions.*
  - *All potential solutions require community consensus;*
    maybe even adaptation of llvm developer policy?;
    maybe even ideally alignment with other compiler communities such as gcc?

WRAPPING UP

How can you take part/contribute?

- Report issues appropriately.
- When needed, spread the word LLVM has a process to responsibly disclose security issues.
- LLVM security group online sync-up.
  https://discourse.llvm.org/t/llvm-security-group-public-sync-ups/
  https://calendar.google.com/calendar/u/0/embed?src=calendar@llvm.org
- Feel welcome to join the LLVM security group and contribute

Summary/conclusions

- The LLVM security group has been running well for 3 years now.
  Transparency reports: https://llvm.org/docs/SecurityTransparencyReports.html
- Analyzing the Security groups' work indicates potential areas for further improvements:
  - *Improving description of threat model/what a security issue is.*
  - *Improving how toolchain-based mitigations are developed, documented, maintained*
  - *Improvements on supply chain security*
  - *Move reporting away from chromium bug tracker*
- If you do encounter a potential security issue requiring careful coordination and disclosure, please remember to report to security group instead of regular bug tracker.

Q&A