Zephyr® Project

Developer Summit

The Zephyr Project Security Overview, Progress, and Status

David Brown, *Linaro*
Flavio Ceolin, *Intel*

Zephyr Security Overview

- Introduction
- Lifecycle of a Vulnerability
- Current Status
- Discussion

Introduction

- Zephyr
- Importance of security
- Security Committee
- Security Working Group

What is Zephyr

- Open Source
- Lots of supported architectures and boards
- Lots of supported features
- Lots of code
  - Zephyr: 1.3 million lines of C
  - Modules: 20 million lines of C

Security Standards

- ETSI EN 303-645 Cybersecurity Standard for Consumer IoT Devices
- FIPS 140-3 Security Requirements for Cryptographic Modules
- SP 800-128 Secure Software Development Framework
- Annex K (C11 standard)

Security Committee

- Defined by project charter
- Has one rep from platinum members, an architect and a chair
- Architect: Flavio Ceolin, Chair: David Brown
- Meeting every two weeks
- Topics that are not public

Security Working Group

- Open to any participants
- First met regularly, with committee meeting on demand
- Now regular and interleaved with committee
- Many useful discussion have resulted
  - Security Standards
  - Security Processes
  - Code analysis tools
- Committee mostly dealing with vulnerabilities and sensitive information

Lifecycle of a vulnerability

- What is a vulnerability, in our context
- Why treated differently
- The process

Vulnerability

“A software vulnerability is a flaw or weakness in a software system that could be exploited to compromise the system's security or functionality. It could be a bug, design flaw, or configuration oversight in the software's code, design, architecture, or user interface.

If a vulnerability is exploited, it could potentially lead to unauthorized access, data loss, or data theft. It could allow an attacker to install malicious software, gain access to sensitive data, or gain control over the system and its resources.”

- ChatGPT (yes, I did it)

Vulnerability for us

- Bugs in the code, flaws in the design, configuration issues, even problems with an implemented standard
- Difference from “normal” bugs: it can be exploited
- This allows an attacker to do something else
- To understand, we need a threat model
- Threat models are hard in Zephyr, we are a general system, and there are lots of uses
- The threats and use also change over time

Treating different than bugs

- Bugs are reported publicly, fixes reviewed and merged, all publicly
- Vulnerabilities are reported and kept private for an “embargo period”
- Fixes still happen publicly, including review, but relation to vulnerability is not overtly stated during embargo
- Theory is to allow certain parties time to mitigate and deal with fix before exploit is known
- Reality hasn’t really worked as well

The Process: Incoming

- Reports to zephyr-psirt mailing list or directly on github
- Issue is entered in as a draft security advisory into github
- Attempt to triage: severity, and what code and maintainers are affected
- Start notification process, maintainers need to know to get a fix started

Process: Fixing

- Maintainers gain visibility along with potentially additional developers to work on fix
- Eventually one or more PRs are created, reviewed and merged
- All of this needs to be tracked in github, currently text in the security advisory, but moving to a project-based tracking system
- Once merged, it will get into a release

Process: Embargo Release

- Time based, 90-days
- Embargo can end, whatever state fixing the issue is in
- We report state of fix, point to patches, describe merged changes, or a release
- A CVE is published with MITRE to disclose vulnerability
- Release notes are updated to give information more than just the existence of the vulnerability

Process: Backports

- Zephyr has a good process for backports, often can happen with a few clicks
- If backport conflicts, security person may see if easy to resolve, otherwise give to maintainer
- Backports need to update release notes, CVE and possibly notification

Process: So How Is It Going?

- Largely manual
- Tried automating, web-scraping not reliable
- There are two types of tasks here:
  - Security “expert” type: analysis, triage, finding maintainers, looking at code
  - Process: notification, updating status based on merges, etc
- We kind of would like a “rotation” with rest of security committee, process things easier for someone to pick up, “expert” e.g. the process tasks are fungible.

Status: Committee

- Originally, just committee, met every two weeks
- Limited audience, started to realize lots of other people would be interested in most topics
- Created working group (March, 2022), changed meeting to this. Committee was to be on demand
- Committee didn’t really happen, so now interleave

Working Group

- General security discussions, open to anyone
- Covered many topics:
  - Security standards
  - Coding guidelines
  - Analysis tools
  - Security labeled issues on github
- Has worked reasonably well

Committee

- Focuses on vulnerabilities, and things involving budget
- Trying to improve the vulnerability process
- Budget things: e.g. third party analysis, sponsoring security conferences, attendance

Cryptography Libraries

- Zephyr uses multiple crypto libraries, can we unify?
- Some subsystems currently use a specific library
- One library, in particular, has no upstream maintainer
- Decided on PSA API
- Actually implementing hasn’t happened, though

Concerns:

- How do we handle vulnerabilities on modules?
  - 3rd party code we have fork in our repo
  - where do fixes go, do reports come from zephyr or forward
  - Can we automate it ? (google/osv-scanner, intel/cve-bin-tool, ...)
  - Do we need more SBOM information?
- module interdependencies
  - Shows up with TF-M and Mbed TLS
  - If someone needs TF-M, it demands a specific version of Mbed TLS
  - If not, Mbed TLS version can be flexible
  - Our last LTS update disabled tf-m due to this, fixes were needed in Mbed TLS
  - West doesn’t really handle this
  - What can we do better?

Questions/discussion