Backporting is so 1993

Ricardo Salveti - ricardo@foundries.io

Michael Scott - mike@foundries.io

Embedded Linux Conference & OpenIoT Summit - Edinburgh

Introduction

Contents

1. Common IoT Product Problems
2. Traditional Embedded Approach
3. Embedded Linux
4. Zephyr
5. Working with latest software

Common IoT Product Problems

FOUNDRIES.IO

- Long product lifetime maintenance (+10 Years)
- Always on / connected
- Large attack surface
- Complex architecture
- Continuous maintenance

Traditional Embedded Approach

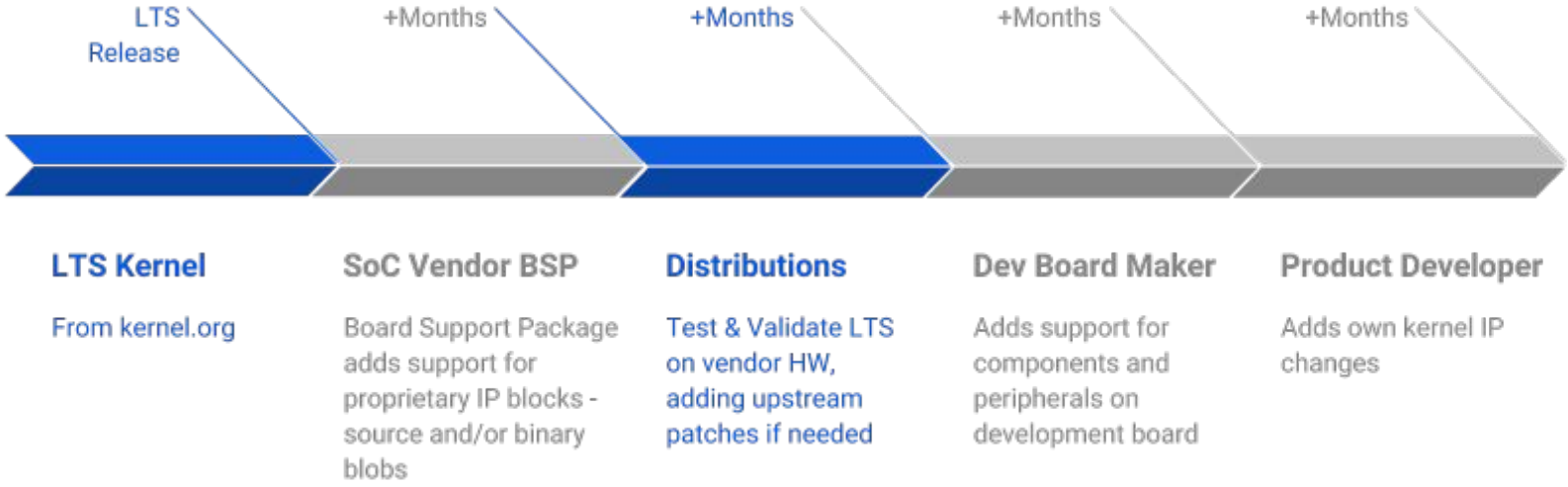
- Start out of vendor-specific BSPs
- Custom OS platform development
- Fork -> customize (hack)-> QA -> release
- In-house software update systems (when available!)

Traditional Product Maintenance

- React only when really needed (criticals only)
- Cherry-pick fixes from the project stable releases
  - Backport when no stable baseline is available
- Iterate on top of a customized software stack, QA and release!

Embedded Linux Development

FOUNDRIES.IO



Complex and long supply chain, hard to maintain and apply updates

Meltdown/Spectre?

Or just move to 4.14.y. Seriously, that's probably the safest thing in the long run for anyone here. And when you realize you can't do that, go yell at your SoC for forcing you into the nightmare that they conned you into by their 3+ million lines added to their kernel tree. You were always living on borrowed time, and it looks like that time is finally up.

-- Greg Kroah-Hartman

https://lkml.org/lkml/2018/3/5/338

Stable Maintenance Is Not Easy

- How to identify bug fixes that should be backported?
  - Which fixes are security related?
- Complex maintenance chain across SoC/Board vendors
- Backport can also introduce new bugs and regressions
- New features might also be desired
  - e.g. Kernel Self Protection

Stable Maintenance Is Not Easy

CVE-2017-18344 - https://seclists.org/oss-sec/2018/q3/76

Bug introduced in 3.10 and fixed in 4.15-rc4.

- Nov 30, 2017 - bug reported by syzbot (4.15)
- Dec 15, 2017 - fix committed upstream (4.15-rc4)
- Feb 17, 2018 - fix backported to the 4.4 LTS
- Mar 15, 2018 - fix added to the Ubuntu Xenial 4.4
- Jul 25, 2018 - CVE requested
- Aug 2, 2018 - notified linux-distros and oss-security

Stable Maintenance Is Not Easy

USN-3741-1 introduced mitigations in the Linux kernel for Ubuntu 14.04 LTS to address L1 Terminal Fault (L1TF) vulnerabilities (CVE-2018-3620, CVE-2018-3646). Unfortunately, the update introduced regressions that caused kernel panics when booting in some environments as well as preventing Java applications from starting.

Stable Maintenance Is Not Easy

USN-3522-1 fixed a vulnerability in the Linux kernel to address Meltdown (CVE-2017-5754). Unfortunately, that update introduced a regression where a few systems failed to boot successfully.

From a product perspective, people really have to rethink what they are doing. This technology is changing and evolving too fast for models which were invented when semiconductors were guaranteed to be available for 20 years and the change rate in technology was comparable to snail mail. **The illusion to support a product for 20 years with software from 20 years ago has been destroyed long ago, but still people cling to it for any price.**

-- Thomas Gleixner

https://lists.linuxfoundation.org/pipermail/ksummit-discuss/2018-September/005198.html

And now for something not Linux: The Zephyr™ Project

The Zephyr™ Project, is a Linux Foundation hosted Collaboration Project, an open source collaborative effort uniting leaders from across the industry to build a best-in-breed small, scalable, real-time operating system (RTOS) optimized for resource constrained devices, across multiple architectures.

Key components: Open Source, Secure, Modular and Connected

And now for something not Linux: The Zephyr™ Project

Rewind to October 2017:

Imagine we are a company making a connected wearable and want to use Zephyr™. We just started development and are planning to release in October 2018.

The Plan: Fork and forget! Right!? What could go wrong?

And now for something not Linux: The Zephyr™ Project

V1.10 released December 8th 2017: ~1800 commits

- Initial alpha-quality thread-level memory protection on x86, user-space and memory domains
- **Major overhaul to the build system and a switch from Kbuild to CMake.**
- HTTP API changed to use net-app API. Old HTTP API is deprecated.
- Deprecated ZoAP library in-favor of new CoAP library
- Various fixes for: TCP, RPL, ARP, DNS, LWM2M, Ethernet, net-app API, Network shell, and BSD socket API

And now for something not Linux: The Zephyr™ Project

V1.11 released March 9th 2018: ~1500 commits

- Thread-level memory protection on x86, ARC and Arm, userspace and memory domains
- Native development environment on Microsoft Windows.
- Thread support via integration with OpenThread.
- Lightweight flash storage layer for constrained devices.
- **LWM2M fixes and enhancements, CoAP fixes, TCP fixes, HTTP fixes, RPL fixes, Net-app API fixes, Net-shell fixes, BSD socket API fixes**

And now for something not Linux: The Zephyr™ Project

V1.12 released June 11th: ~1950 commits

- Support multiple concurrent filesystem devices, partitions, and FS types
- kernel/sched: Fix preemption logic
- **kernel: Scheduler rewrite**
- Add initial WiFi management API definitions.
- Network timeout fixes, ICMPv6 error check fixes, BSD socket sample application fixes.
- Multiple BLE Mesh bugfixes and improvements

And now for something not Linux: The Zephyr™ Project

V1.13 released Sept. 10th: ~1800 commits

- Support for TLS and DTLS using BSD socket API
- ADC: Introduced reworked API and updated Nordic, NXP, Atmel, and Synopsys DesignWare drivers
- **Bug fixes: Handle large IPv6 packets properly, IPv6 address lifetime fixes, IPv6 fragmentation fixes, DHCPv4 fixes, TCP retry, RST packet handling, and memory leak fixes, HTTP fix when sending the last chunk, MQTT fixes, LWM2M cleanups and fixes.**

And now for something not Linux: The Zephyr™ Project

Upcoming movement for v1.14:

- System Logger changed to Logger (including macro name changes)
- **Network APIs moving from linked list network buffer (net_app) APIs to POSIX socket APIs**
- Major rework of Timer APIs, and more!

Can you imagine shipping with Zephyr v1.9 from September 7th last year?

Our Belief

There is no such thing as secure software since security is an arms race. Therefore, the latest software is the most secure.

Latest Software

- Easier to report and fix issues
- Faster review & testing iteration
- Less load on upstream maintainers
- Easier integration of new features
- Smaller supply chain

Working with Latest: Zephyr

How do you manage large amounts of churn in a project like Zephyr?

It’s not easy, but it’s essential

- Updating a product after it’s been shipped has become the norm. Consumers have more confidence in companies that respond quickly to security issues.

Working with Latest: Zephyr

How do you manage large amounts of churn in a project like Zephyr?

#1 Take it in Bite-sized Chunks

- Bring in “small” batches of 100 to 200 commits at a time.
- Easier to bisect regressions and understand as major changes enter the codebase.

Working with Latest: Zephyr

#2 Who Tests the Testers?

- Design a test plan with a goal in mind. **Don't just test things at random.**
- Look at the project's unit tests and build those as often as you can.. Preferably on every upstream commit.
- Test mainline samples that demonstrate portions of your use-case.

Working with Latest: Zephyr

#3 Understand the Development Cycle

Read the Zephyr wiki program management page: https://github.com/zephyrproject-rtos/zephyr/wiki/Program-Management

There will be **A LOT** of “hazardous” commits going in during the first few weeks of new development. It then slows down and gets better as the RCs are released.

Working with Latest: Linux

- Separation of core platform from application code
  - Allow pieces to move independently
- CI and validation ahead of releases
  - Yocto / OpenEmbedded: master and master-next
  - KernelCI.org
- Joint effort across IP, SoC and Hardware vendors for continuous upstream support and validation

Move older systems to newer kernels (current long-term stable), it includes all the latest fixes and perhaps some useful features - some security hardening, other stuff you might want to have. It is the best kernel we know how to make.

I think it is the way we're going to end up eventually, doing this rather than trying to support ancient kernels.

-- Jonathan Corbet - The Kernel Report ELC-E 2018

https://www.youtube.com/watch?v=XQGUi5Gu0D8

Thank You!

FOUNDRIES.IO