

Open Source Summit Japan 2022 (LinuxCon)

# **Delivering Security Fixes Continuously into Your Embedded Linux**



Hiraku Toyooka  
Cybertrust Japan

Hiraku Toyooka <hiraku.toyooka@{cybertrust.co.jp,miraclelinux.com}>

- Living in Yokohama
- Senior Manager in EMLinux dev team at Cybertrust Japan
  - EMLinux: Yocto-based Linux environment for embedded/IoT systems
- Maintainer of [meta-emlinux](#), [meta-debian-extended](#)
- (A little) contributing to [CIP Testing](#), [meta-debian](#)
  - CIP: Civil Infrastructure Platform
- Previously also contributed to Linux kernel

- Background
- Considerations to get prepared for security updates
- How EMLinux provides continuous security updates

# Background

# Security updates for embedded / IoT systems

---

- Security updates are becoming recognized as essential
- Lots of cyberattacks on embedded/IoT devices have already been observed
  - from familiar items (home appliances and cars), to infrastructures (buildings, factories and plants)

- Standards, guidelines, regulations require security update processes
  - IoT devices (in general)
    - NISTIR 8259 series (US), EN 303 645 (EU), MIC Ordinance No.12 of H31 (JP)
  - Automotive
    - UNECE WP29 UN-R155/R156, ISO/SAE 21434
  - Medical
    - IMDRF Principles and Practices for Medical Device Cybersecurity
  - Industrial Control Systems
    - ISA/IEC 62443
- Procurement requirements may include such standards
  - You may be involved even if those standards are not legally enforced yet

- Example in Linux kernel
  - 150-350 vulnerabilities(CVEs) are reported annually
  - Greg KH, “[Linux Kernel Development](#)” says:
    - > Average fix date, -100 days
    - > Bugs are fixed before you realize it
- Thankfully, Linux community release security fixes quickly in most cases

# What about “your” embedded Linux?

---

- Thankfully, Linux community release security fixes quickly in most cases
  - But often observed that product developers are unable to leverage those fixes for their own products
- You may not able to leverage those fixes for your product when ...
  - you don't have an appropriate maintenance plan or processes
  - your embedded Linux is inconsistent with open-source (upstream) ecosystem

How do such situations happen?

What should we do to change such situations?

# Considerations to get prepared for security updates

# Do you have any maintenance plan or process?

---

- Too late to think about them after shipment (or just before shipment)
  - You can not respond quickly to sudden incidents
  - Your embedded Linux will become difficult to maintain (described later)
- You'd better to consider the appropriate maintenance plan and processes **from the beginning of development**
  - Maintenance period, cycle
  - Update model
  - How to check vulnerabilities
  - How to prepare the fix
  - How to verify the update
  - How to deploy the update

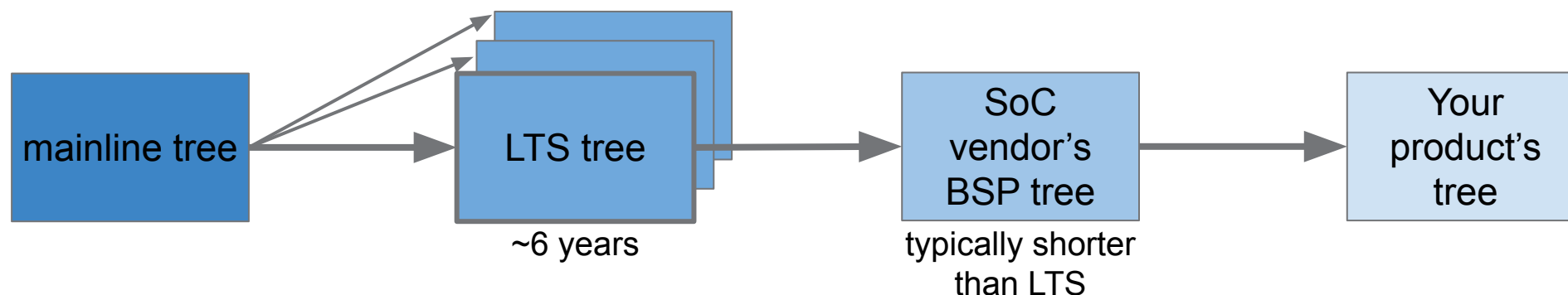
# Does your update process work?

- Now, you have maintenance processes prepared 5 years ago
  - But is the process itself still maintained after shipment?
- If the update is assumed as “only on-demand” or “once in a decade”,
  - Nobody may have experienced the actual update
  - Developer who created the process may no longer be employed
  - Software update feature/infrastructure may be broken
- The most reliable measure is to practice (or check) the maintenance process **continuously**



# Where do you get the security fix from?

- Linux kernel in embedded product is typically based on LTS (long-term stable) version
  - and customized for your hardware

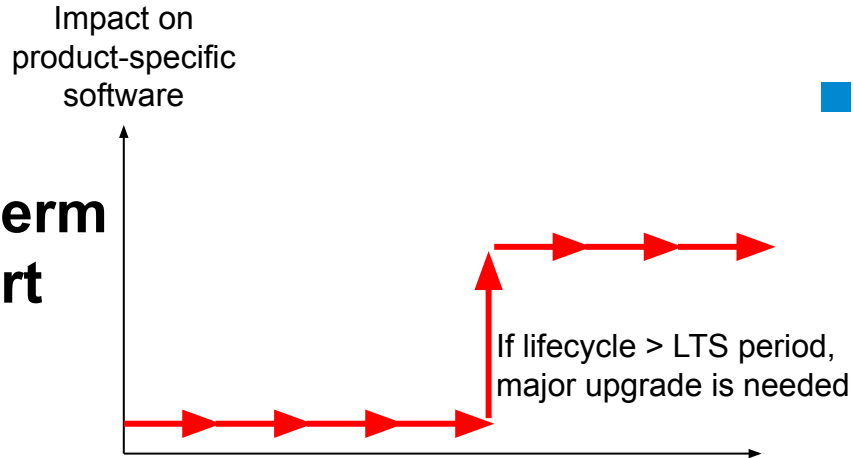


- Once the **maintenance period** of the upstream LTS version ends, it will be difficult to get security fixes that match your tree
  - You should always **use the (actively) maintained version**
  - or you need to backport the fixes to your tree by yourself (costly)

# How do you cover the long-term lifecycle of your product?

- Some kind of update model will be necessary (not limited to the following)

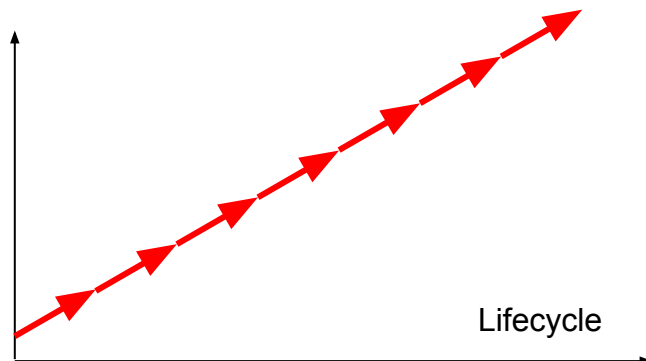
## Long-term Support model



- Keep using one LTS version as long as possible

- Pros: Less opportunities to modify product-specific software (drivers and applications)
- Cons: Large migration cost at version up, Possible delay in security fixes in later phase

## Rolling Update model



- Always use the latest LTS (or stable) version

- Pros: Quick response to security bugs, New feature utilization
- Cons: More opportunities to modify product-specific software

# How do you take the security fix into your product's tree?

---

- “cherry-picking” only the fix patch from the LTS tree?
  - Not recommended for Linux kernel
  - LTS trees merge many changes at a fast pace, so prerequisite code to apply the fix may be missing in your tree
  - If you cherry-pick, you need more time to confirm if the bug is really fixed
  - In addition, other known (security) bugs will remain in your product
- You should **take all LTS updates** into your product's tree
  - Also sync your tree with the latest state of the LTS tree **continuously**
    - Useful to keep your product ready for the next security updates

# Some sayings (cherry-picked from other presentations)

- Small fraction of kernel fixes get CVEs
- Cherry-picking CVEs result in insecure system

Cited from: Greg Kroah-Hartman, "[Linux Kernel Release Model](#)"

(en)

- “We are using LTS!” is not enough. It doesn't make sense if you don't keep incrementing “x” (revision) of “4.19.x”

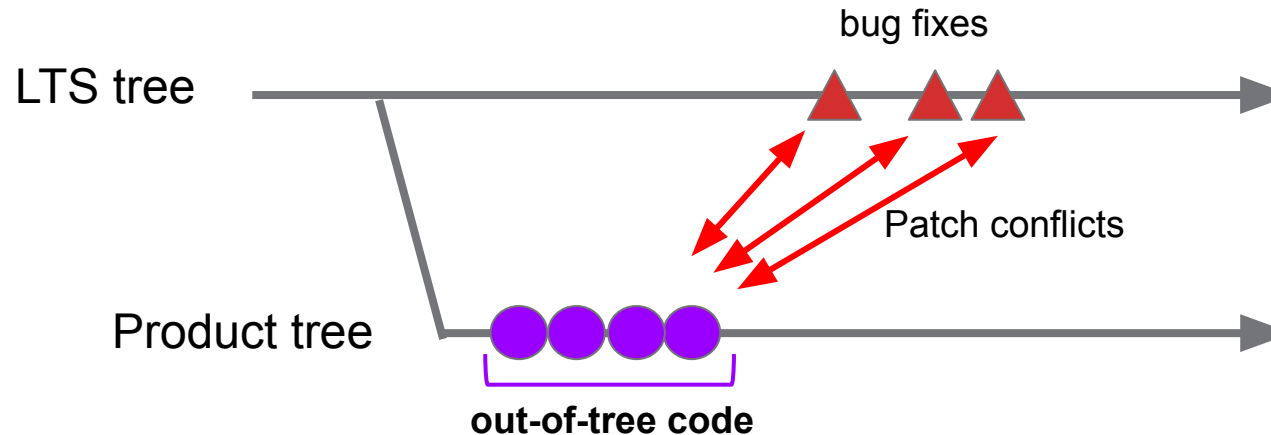
(ja)

- **LTS使ってます！, ではダメで, 4.19.x の x (リビジョン) を上げ続けないと意味がない**

Cited from: Shinsuke Kato, "[Linux Kernel versions and Longterm Stable Kernel \(LTS\)](#)",  
(Translated to English by Hiraku Toyooka)

# How much out-of-tree code do you have?

- Product's kernel tree is often based on the BSP of the SoC evaluation board
  - Most BSPs contain out-of-tree(\*) code for quick evaluation of hardware functions  
(\* Not merged to mainline)
- Additional out-of-tree code is added when supporting product hardware
- Massive amount of out-of-tree code causes lots of conflicts when you take LTS fixes into your product's tree
  - Then, your embedded Linux will become difficult to maintain



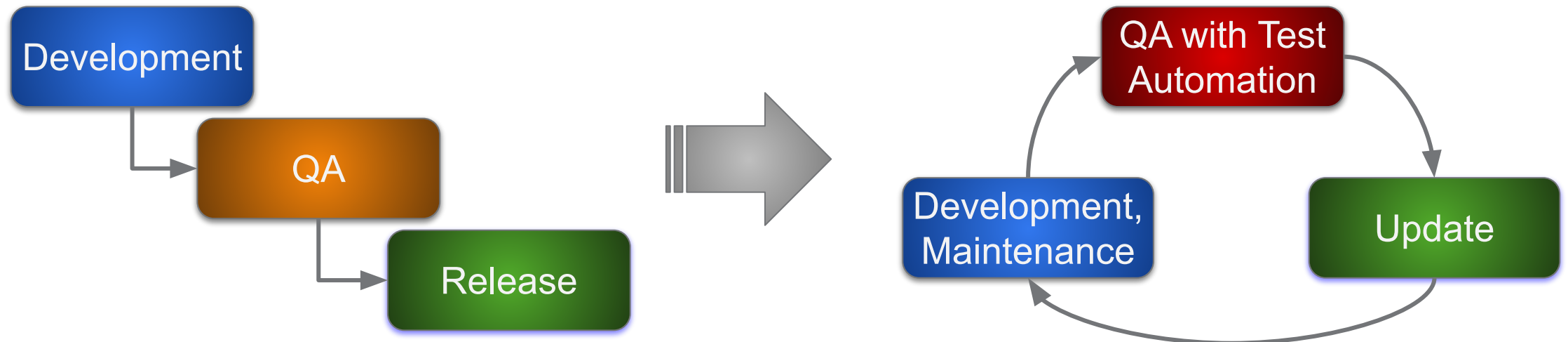
# Hardware selection considering software maintenance

---

- Some hardware vendors are doing upstreaming (mainlining) activities in the Linux kernel community
  - to have their own hardware support code integrated into the mainline tree
- You can avoid huge maintenance costs **by selecting hardware whose support code is well integrated into the mainline**
  - Software ecosystem is also important when selecting hardware

# Is your QA system ready for continuous updates?

- QA process will be required before the update in the field
  - More QA opportunities, more cost
  - Lead time is also an issue
- Test automation or CI/CD is required for **continuous and quick updates at a feasible cost**



# How EMLinux provides continuous security updates

# What is EMLinux?



- Yocto/OE-based embedded Linux environment
  - Developed and maintained by Cybertrust Japan
  - Used by various embedded product vendors
- Purpose
  - To deliver security-fixes and bug-fixes to users continuously

# Maintenance plan and processes for EMLinux

## ■ Brief overview

Note: EMLinux is supposed to be customized, so it is not directly deployed in the final product.

|                              |                                                                                            |
|------------------------------|--------------------------------------------------------------------------------------------|
| Maintenance period, cycle    | 10 years per major version: standard (5y) + late phase (5y)<br>Monthly maintenance updates |
| Update model                 | Based on LTS model<br>Major version upgrade every (2-)4 years                              |
| How to check vulnerabilities | Yocto cve-check extended for EMLinux<br>In-house CVE review process                        |
| How to prepare the fix       | Take all upstream fixes from CIP-SLTS* kernel and Debian pkgs                              |
| How to verify the update     | Verify changes continuously with in-house “EMLinuxCI”                                      |
| How to deploy the update     | (Depends on users)                                                                         |

\* Super Long Term Stable

# EMLinux: Why we choose LTS model?

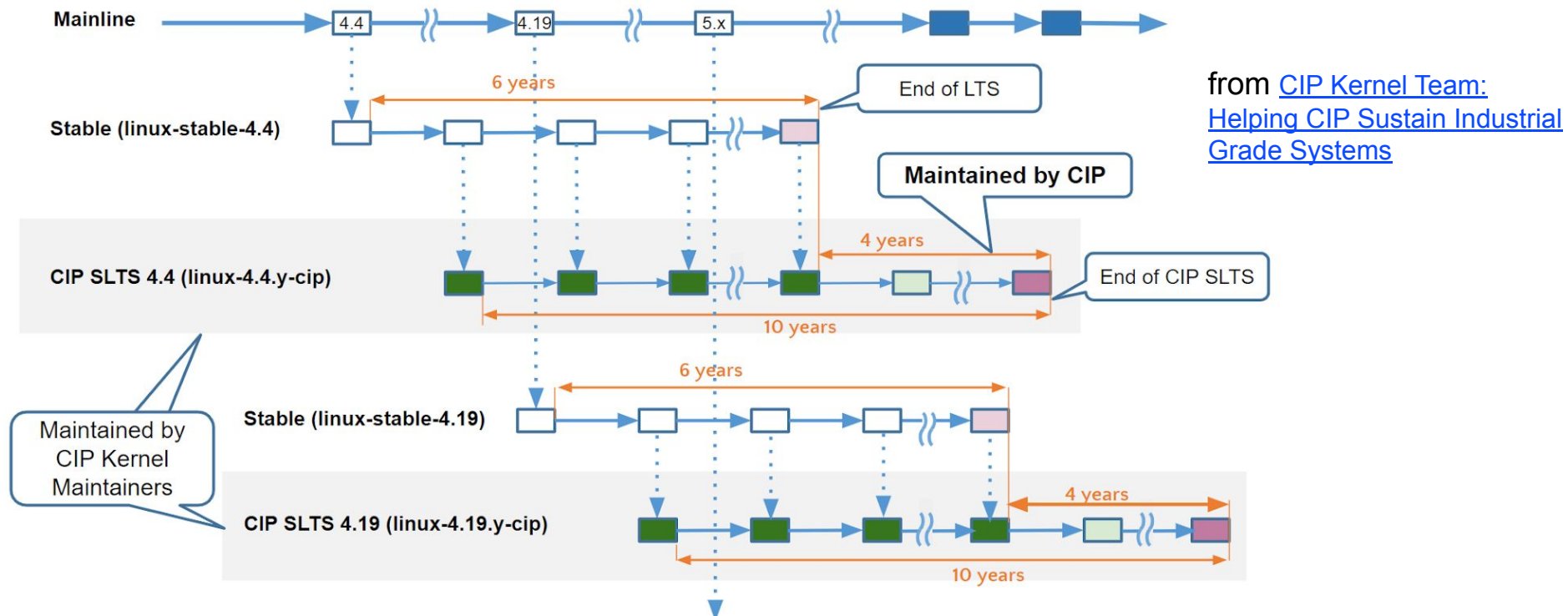
---

- LTS model is easier for most users to start the security maintenance process
  - including CI/CD processes
- Less changes in features or interfaces
  - Less impacts on product-specific code
  - (Automated) test cases are re-usable for a long time
- Major version upgrade will also provided for strict security requirements

# Leveraging CIP SLTS kernel

## ■ CIP Super Long-term Stable (SLTS) kernel

- maintained for 10+ years by CIP (Civil Infrastructure Platform) Project
  - Cybertrust Japan is a member of CIP
- Upstream first policy. All LTS bug-fixes are merged.
- Twice a month release for 4.19.y-cip, 5.10.y-cip (currently)



## ■ [cip-kernel-sec](#)

- CIP's Linux kernel CVE tracker
- Maintained by Masami Ichikawa(@Cybertrust Japan) frequently (every few days)
- Integrated to EMLinux to have the Yocto's cve-check feature output accurate CVE status for SLTS kernel

```
[Example: CVE-2022-41674.yml]

description: fix u8 overflow in cfg80211_update_notlisted_nontrans
...
introduced-by:
  mainline: [0b8fb8235be8be99a197e8d948fc0a2df8dc261a]
fixed-by:
  cip/5.10: [a6408e0b694c1bdd8ae7dd0464a86b98518145ec]
  mainline: [aebe9f4639b13a1f4e9a6b42cdd2e38c617b442d]
  stable/5.10: [a6408e0b694c1bdd8ae7dd0464a86b98518145ec]
  stable/5.15: [9a8ef2030510a9d6ce86fd535b8d10720230811f]
...
```

# Leveraging Debian (userland) pkgs via meta-debian

---

## ■ [meta-debian](#)

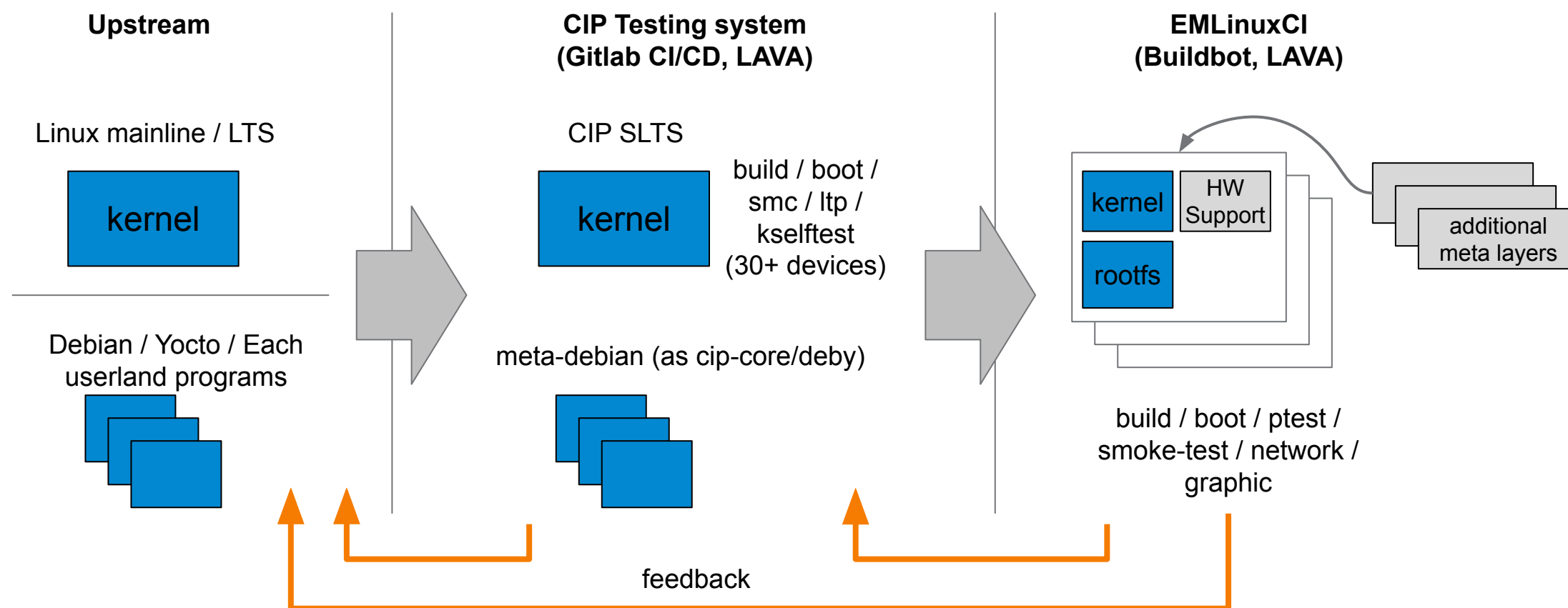
- “Yocto Project extension for using Debian source packages”
- Created by Toshiba people
- Debian packages:
  - Stable version typically accepts only bug-fixes
  - 3 + 2(LTS) years maintenance period (and Extended LTS adds a few years)
  - Some packages will be maintained as CIP Core Packages for 10 years
- Package version can be updated to Debian’s latest version
- Debian security tracker’s data integrated to Yocto cve-check

## ■ [meta-debian-extended](#)

- Additional packages for meta-debian (Same recipe format)
- Created by Cybertrust Japan

# Test Automation

- EMLinux is continuously verified with in-house “EMLinuxCI”
  - to support monthly (or on-demand) updates and daily development
- Found bugs / issues are fed back to each upstream projects



- EMLinux depends on many open-source projects related to Test Automation
- We are making lots of contributions to build a better Test Automation ecosystem (also helpful to stabilize EMLinux)
  - KernelCI
    - Alice Ferrazzi(@Cybertrust Japan) is a TSC member, and a maintainer for some subprojects
  - CIP Testing
  - Buildbot
  - LAVA

# (Other) Contributions to open-source projects

---

- When we find issues or make bug-fixes, we try to give feedback to upstream projects as much as possible
  - Helpful to reduce self-maintenance over the long term
- Projects we have provided feedback on
  - Linux LTS
  - Debian
  - Yocto Project
  - OpenEmbedded
  - util-linux

Reference: [List of open-source projects Cybertrust Japan has contributed to](#)

# Recap

- To deliver security fixes to your embedded Linux, we need to
  - Consider a maintenance plan and processes from the beginning of development
  - Keep your embedded Linux consistent with open-source ecosystem
  - Make whole maintenance system ready for continuous updates