Upstream First is Our Principle
– Toward Super Long-Term Support –

Chris Paterson @ Renesas
Masashi Kudo @ Cybertrust Japan
October 26th, 2020
About US

- **Masashi Kudo** <masashi.kudo@miraclelinux.com>
  - Working for Cybertrust Japan Co., Ltd.
  - Acted as OpenDaylight (LF Networking) Ambassador
  - CIP Kernel Team Chair

- **Chris Paterson** <chris.paterson2@renesas.com>
  - Working for Renesas Electronics Europe GmbH
  - CIP Testing Working Group Chair
Table of Contents

What is CIP?
Upstream First
CIP Kernel Team Activities
CIP Automated Testing
Summary
What is CIP?
Handling increasing complexity with constant development resources

Join forces by leveraging commodity components, partnering, and adapting open source software.

Open source software ensures long-term availability, flexibility, and maintainability without vendor lock-in.

**Differentiating**

Why to buy the product

**Commodity**

Invisible for customers

**Up to 2000**

- Proprietary application, proprietary operating system

**2000 – 2015**

- Proprietary application
- Operating System

**2016 and beyond**

- Proprietary application
- Middleware
- Communication
- Domain-specific frameworks
- HMI frameworks
- Augmented reality platforms
- Mobile interfaces
- Enterprise IT interfaces
- IoT backend
- Stream processing
- Cloud orchestration
- Cloud frameworks

**Speed and efficiency: focus on differentiating parts**
Facts and Issues: Silo Development

Facts

• Millions or trillions Industrial devices, including smart devices
• Similar software components (e.g. Linux)
• Industrial IoT requirements
  • Security
  • Sustainability
  • Industrial-gradeness

Issues

• A lot of products have to meet industrial requirements
• Same development and maintenance efforts spent by many companies or even business units
• **No common solution** for base building blocks
Establishing an Open Source Base Layer of industrial-grade software to enable the use and implementation of software building blocks for Civil Infrastructure Systems.
What is “Open Source Base Layer (OSBL)”?

**OSBL**

- **CIP SLTS kernel**
  - (10+ years maintenance, based on LTS kernels)
- **CIP Core packages**
  - (tens)
- **additional packages**
  - (hundreds)

System-specific middleware and applications

Scope of a typical Linux distribution

**CIP** Civil Infrastructure Platform Project ([https://www.cip-project.org/](https://www.cip-project.org/))

**SLTS** Super Long Term Support
CIP governance structure and projects

Governing Board (GB)

Technical Steering Committee (TSC)

CIP Projects and its scopes

Security
Sustainability
Industrial grade

(*): Workgroup

SLTS kernel
Real-time
Testing
CIP Core
Security WG(*)
Software update WG(*)

On-device software stack
Product development and maintenance

User space
Kernel space

Topics:

- Technical Steering Committee (TSC)
- SLTS kernel
- Real-time
- Testing
- CIP Core
- Security WG(*)
- Software update WG(*)

Governing Board (GB) with governing topics:
- ✔ Security
- ✔ Sustainability
- ✔ Industrial grade

(*) Workgroup for specific topics.
The backbone of CIP are the member companies

![CIP Member Companies](image)

Budget

Developers, maintainers

Optional: funding of selected projects

Contribution & usage / integration

Open Source Projects (Upstream work)
Mapping CIP into the company

Up to 70% effort reduction achievable for OSS license clearing and vulnerability monitoring, kernel and package maintenance, application adaptation and testing for an individual product.

OSS Open Source Software  QA quality assurance  SDK software development kit
Upstream First
Development Models

“Own Community” Model
Branches its base from upstream and evolves by its own.

“Upstream First” Model
Only allows patch commits if those patches are already in the upstream.
Commit Counts per LTS

Note: If a patch has an original patch, the date of the patch is that of the original one.
Collaborative development with other OSS projects

Upstream Projects

1. Upstream first

2. Use the upstream code

3. Integrate

Contribute, Collaborate and use by CIP

CIP Open Source Base Layer (OSBL)
How CIP Artifacts can be used

CIP Core packages
- CIP kernel
- additional packages

middleware and applications
- CIP Core packages
- CIP kernel

Yocto/Poky
Build framework from source code for embedded systems

CIP Source Packages
Compile and optimize for embedded devices
Apply CIP Linux

Debian Source Packages
Build
Binary Packages (deb)
Install
Repository

Debian User
apt/apt-get

source packages
CVE patches

referencing

CIP User 😊
CIP
Kernel Team
Activities
Primary Goal

- Provide CIP SLTS kernels with ten+ years maintenance period by fixing versions to fulfill the required level of reliability, sustainability, and security

Team Members

- Masashi Kudo – Chairperson
- Nobuhiro Iwamatsu – Kernel Maintainer
- Pavel Machek – Kernel Maintainer
- Ben Hutchings – Kernel Mentor
- Chen-Yu Tsai – Kernel Developer
CIP SLTS kernel development

Mainline / LTS

LTS kernel

Contributions

Patch Review

CVE Check

Kernel Releases

Kernel Team

Developers
Stable Patch Review

- Reviews for -rc
  - Review results are posted to ML
- Reviews for stable releases
Gather kernel CVE Information

cip-kernel-sec I/F

Analyze CVEs to determine necessities for contributions

Refer
cip-kernel-sec

- Tracks the status of security issues, identified by CVE ID, in mainline, stable, and other configured branches.
Necessity of contributions (backporting) is determined to be fixed base on kernel configurations provided by CIP members.
Contributions to LTS

Chart showing contributions to various LTS versions (4.4, 4.9, 4.14, 4.19, 5.4) from June to October, with increases marked (+12, +16, +15, +17, +16) in contributions from June to October 1599 to 1685.
## Contributions to LTS – Details

<table>
<thead>
<tr>
<th></th>
<th>v4.4.238</th>
<th>v4.9.238</th>
<th>v4.14.200</th>
<th>v4.19.149</th>
<th>v5.4.69</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suggested-by:</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>Reported-by:</td>
<td>44</td>
<td>35</td>
<td>29</td>
<td>16</td>
<td>6</td>
<td>130</td>
</tr>
<tr>
<td>Signed-off-by:</td>
<td>440</td>
<td>334</td>
<td>149</td>
<td>88</td>
<td>41</td>
<td>1052</td>
</tr>
<tr>
<td>Debugged-by:</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Author:</td>
<td>80</td>
<td>83</td>
<td>55</td>
<td>39</td>
<td>23</td>
<td>280</td>
</tr>
<tr>
<td>Acked-by:</td>
<td>26</td>
<td>29</td>
<td>33</td>
<td>44</td>
<td>13</td>
<td>145</td>
</tr>
<tr>
<td>Reviewed-by:</td>
<td>2</td>
<td>4</td>
<td>10</td>
<td>7</td>
<td>6</td>
<td>29</td>
</tr>
<tr>
<td>Tested-by:</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td></td>
<td>17</td>
</tr>
<tr>
<td>Cc:</td>
<td>104</td>
<td>97</td>
<td>72</td>
<td>51</td>
<td>28</td>
<td>352</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>618</strong></td>
<td><strong>496</strong></td>
<td><strong>289</strong></td>
<td><strong>195</strong></td>
<td><strong>87</strong></td>
<td><strong>1685</strong></td>
</tr>
</tbody>
</table>

**Note:** There could be multiple contributions by a same personnel in one commit. such duplicates are eliminated in total numbers. Therefore, the summation of each item may not equal to “Total”. 

as of October 6, 2020
1. Review stable patches – status tracked in Gitlab [1]
   • Mark the review and the name of the worker under the commit.
   • **Start to review stable kernel patches in rc stage**
2. Review patch from CIP members via cip-dev [2]
   • Update the status of the commit in patchwork
3. Start testing
4. Tag release candidate
5. Ack by other maintainers
6. Release and send the news to cip-dev

## CIP SLTS Kernel Release Policy

<table>
<thead>
<tr>
<th>Current Releases</th>
<th>Life-Cycle</th>
<th>Release Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>First Release</td>
<td>Projected EOL</td>
</tr>
<tr>
<td>SLTS 4.4</td>
<td>SLTS 4.4</td>
<td>2017-01-17</td>
</tr>
<tr>
<td></td>
<td>SLTS 4.4-rt</td>
<td>2017-11-16</td>
</tr>
<tr>
<td>SLTS 4.19</td>
<td>SLTS 4.19</td>
<td>2019-01-11</td>
</tr>
<tr>
<td></td>
<td>SLTS 4.19-rt</td>
<td>2019-01-11</td>
</tr>
</tbody>
</table>

**Note:** Difficult to estimate actual release date because of number of patches depends on each stable release

- SLTS 4.4:
  - First Release: 2017-01-17
  - Projected EOL: 2027-01
  - Regular Release: once a month
  - Release on Demand: Depends on criticality of bug / security fixes

- SLTS 4.4-rt:
  - First Release: 2017-11-16
  - Projected EOL: 2027-01
  - Regular Release: once every two months

- SLTS 4.19:
  - First Release: 2019-01-11
  - Projected EOL: 2029-01
  - Regular Release: twice a month
  - Release on Demand: Depends on criticality of bug / security fixes

- SLTS 4.19-rt:
  - First Release: 2019-01-11
  - Projected EOL: 2029-01
  - Regular Release: once every two months

---

27
Progress this year

June 26 → October 46

CIP SLTS Kernel Release Statistics

as of 11 Oct. 2020
CIP SLTS Kernel Maintenance

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LTS</td>
<td>4.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIP SLTS</td>
<td>4.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Maintained by the LTS Project

Upstream First

Self-maintenance

We are here

Self-maintenance starts here
CIP
Automated Testing
Testing Goals

- **Centralised control / distributed testing**
  - CIP developers who are distributed over the world should be able to test CIP software on the CIP reference platforms, even if they don’t have a platform locally

- **Automated testing with Continuous Integration (CI)**
  - Sustain periodical and long-term kernel releases cost-effectively

- **Open Source collaboration**
  - Improve the whole ecosystem and avoid reinventing the wheel
Open Source Approach

1. Upstream First

2. Use

3. Integrate

- KernelCI
- LAVA
- lava-ciplatform.org
- lab-cip-cybertrust
- lab-cip-dnx
- lab-cip-mentor
- lab-cip-renesas
Open Source Approach – In Practice

1. **Upstream First**
   - Funding
   - CIP Testing WG

2. **Upstream Code & Code Reviews**
   - KernelCI
   - LAVA
   - lava-docker

3. **Use**
   - Build automated testing system

4. **Integrate**
   - Test CIP software
   - Test linux-stable release candidates

**Steps:**
1. Upstream First
2. Use
3. Integrate
Testing Architecture Overview

<table>
<thead>
<tr>
<th>Source</th>
<th>CIP Kernel</th>
<th>CIP-Core</th>
<th>stable-rc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build</td>
<td>GitLab runner @ k8s master</td>
<td>GitLab runner @ k8s master</td>
<td></td>
</tr>
<tr>
<td></td>
<td>k8s pod (build)</td>
<td>k8s pod (build)</td>
<td>k8s pod (test)</td>
</tr>
<tr>
<td>Test</td>
<td>Artifact Storage</td>
<td>Built Artifacts</td>
<td>LAVA Master</td>
</tr>
<tr>
<td></td>
<td>LAVA Worker</td>
<td>LAVA Worker</td>
<td>LAVA Worker</td>
</tr>
</tbody>
</table>

Location key:
- GitLab.com
- AWS EC2
- AWS S3
- AWS EC2 on-demand
- Local

CIP Reference Hardware
# CIP Reference Boards

## CIP Reference Boards

<table>
<thead>
<tr>
<th>Platform</th>
<th>Architecture</th>
<th>SLTS v4.4</th>
<th>SLTS v4.4-rt</th>
<th>SLTS v4.19</th>
<th>SLTS v4.19-rt</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM335x Beaglebone Black</td>
<td>Armv7</td>
<td>Y</td>
<td>Y¹</td>
<td>Y</td>
<td>Y¹</td>
</tr>
<tr>
<td>Cyclone V DE0-Nano-SoC Development Kit</td>
<td>Armv7</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y¹</td>
</tr>
<tr>
<td>QEMU</td>
<td>x86_64</td>
<td>Y</td>
<td>Y¹</td>
<td>Y</td>
<td>Y¹</td>
</tr>
<tr>
<td>RZ/G1M iWave Qseven Development Kit</td>
<td>Armv7</td>
<td>Y</td>
<td>Y¹,²</td>
<td>Y</td>
<td>Y¹,²</td>
</tr>
<tr>
<td>RZ/G2M HopeRun HiHope</td>
<td>Armv8</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y¹,²</td>
</tr>
<tr>
<td>SIMATIC IPC227E</td>
<td>x86_64</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y¹</td>
</tr>
<tr>
<td>OpenBlocks IoT VX2</td>
<td>x86_64</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y¹</td>
</tr>
</tbody>
</table>

## CIP Reference Board Candidates

<table>
<thead>
<tr>
<th>Platform</th>
<th>Architecture</th>
<th>SLTS v4.4</th>
<th>SLTS v4.4-rt</th>
<th>SLTS v4.19</th>
<th>SLTS v4.19-rt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zynq UltraScale+ MPSoC ZCU102 Evaluation Kit</td>
<td>Armv8</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y¹</td>
</tr>
</tbody>
</table>

¹ Tested with standard Kernel configuration (non-RT)
² Tested with Real-Time enabled Kernel configuration
• Currently CIP is running the following tests:
  • Boot test
    • `uname -a`
  • Spectre/Meltdown checker
    • A shell script to tell if your system is vulnerable against the several "speculative execution" CVEs that were made public in 2018.
    • https://github.com/Linaro/test-definitions/tree/master/automated/linux/spectre-meltdown-checker-test
  • LTP
    • `ltp-cve-tests, ltp-dio-tests, ltp-fs-tests, ltp-ipc-tests, ltp-math-tests, ltp-open-posix-tests, ltp-sched-tests, ltp-syscalls-tests and ltp-timers-tests`
    • https://github.com/Linaro/test-definitions/tree/master/automated/linux/ltp
    • https://github.com/Linaro/test-definitions/tree/master/automated/linux/ltp-open-posix
  • Cyclictest+Hackbench
    • This test measures event latency in the Linux Kernel, with hackbench running in the background to stress the system.
    • https://gitlab.com/cip-project/cip-testing/linux-cip-ci/-/blob/master/lava_templates/test_cyclictest+hackbench.yaml
Example GitLab Pipeline
Test Results in LAVA
Collaboration with KernelCI

- CIP joined the KernelCI project at its inception in 2019.
- As premier members we help to manage and steer the project.
- We also contribute via code and code reviews, and plan to keep improving the project in this way.
- In next few months we plan to start using the KernelCI front end to help us visualise our build and test results.

- If you want to learn more about KernelCI please attend Guillaume’s BoF later today or Khouloud’s talk on Wednesday evening.
Summary
Summary

• CIP Kernel and Test Teams follows “Upstream First” principle, and contributes to upstream.

• By taking advantage of kernel LTS, the team steadily releases CIP SLTS kernels, and aims to maintain them for 10 years or more.

• To reduce CIP SLTS kernel release cost, the team is closely working with CIP testing team to build automated testing systems.
Please join us to sustain Civil Infrastructure together!
Weekly Regular Online Meeting

- CIP IRC weekly meeting – Every Thursday UTC (GMT) 09:00

<table>
<thead>
<tr>
<th>US-West</th>
<th>US-East</th>
<th>UK</th>
<th>DE</th>
<th>TW</th>
<th>JP</th>
</tr>
</thead>
<tbody>
<tr>
<td>02:00</td>
<td>05:00</td>
<td>10:00</td>
<td>11:00</td>
<td>17:00</td>
<td>18:00</td>
</tr>
</tbody>
</table>

- Channel:
  * irc:chat.freenode.net:6667/cip

- The meeting is used to share status among CIP developers (Kernel Team, Test Team, SW Update WG, Security WG)
CIP Kernel Workgroup Repositories

• CIP Linux kernel & real-time kernel
  • https://git.kernel.org/pub/scm/linux/kernel/git/cip/linux-cip.git

• CIP Linux kernel CVE tracker
  • https://gitlab.com/cip-project/cip-kernel/cip-kernel-sec

• CIP Linux kernel failed patches tracker
  • https://gitlab.com/cip-project/cip-kernel/classify-failed-patches
CIP Testing Workgroup Links

• CIP Testing WG wiki page
  • https://wiki.linuxfoundation.org/civilinfrastructureplatform/ciptesting/ciptestingwg

• CIP LAVA master
  • https://lava.ciplatform.org/

• CIP’s fork of lava-docker
  • https://gitlab.com/cip-project/cip-testing/lava-docker

• GitLab Cloud CI – manages our k8s build pods
  • https://gitlab.com/cip-project/cip-testing/gitlab-cloud-ci

• CIP CI – scripts used to build and test the Kernel
  • https://gitlab.com/cip-project/cip-testing/linux-cip-ci
Contact Information and Resources

To get the latest information, please contact:

• CIP Mailing List: cip-dev@lists.cip-project.org

Other resources

• Twitter: @cip_project
• CIP Web Site: https://www.cip-project.org
• CIP News: https://www.cip-project.org/news/in-the-news
• CIP Wiki: https://wiki.linuxfoundation.org/civilinfrastructureplatform/
• CIP Source Code
  • CIP repositories hosted at kernel.org: https://git.kernel.org/pub/scm/linux/kernel/git/cip/
  • CIP GitLab: https://gitlab.com/cip-project
Thank You

Upcoming CIP Sessions

○ **CIP Mini summit**
  - Friday, October 30 • 11:00 - 12:30

○ **Other CIP members talks**
  - The International Effort to Establish Open Source Base Layer of Cyber Security for IACS
    - Kento Yoshida, Renesas Electronics Corporation
    - Wednesday, October 28 • 16:15 - 17:05
  - Threat Modelling - Key Methodologies and Applications from OSS CIP Perspective
    - Dinesh Kumar, Toshiba Software India & SZ Lin, Moxa Inc
    - Tuesday, October 27 • 14:15 - 15:05