CIP Kernel Team Activities to Accomplish Super Long Term Support

SZ Lin (林上智) @ Moxa Inc.
Masashi Kudo @ Cybertrust Japan
June 29th, 2020
About US

• **SZ Lin (林上智) <sz.lin@moxa.com>**
  - Working for Moxa Inc.
  - CIP Technical Steering Committee Member
  - OpenChain Project Governing Board Member
  - Debian Developer

• **Masashi Kudo <masashi.kudo@miraclelinux.com>**
  - Working for Cybertrust Japan Co., Ltd.
  - Acted as OpenDaylight (LF Networking) Ambassador
  - CIP Kernel Team Chair
What is CIP?
Speed and efficiency: focus on differentiating parts

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.
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

picture taken from Pinterest
https://www.pinterest.de/pin/554646510344033382/
CIP is the Solution

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)”?

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

**system-specific middleware and applications**

**OSBL**

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

SLTS Super Long Term Support
Scope of activities

1. **Super Long Term Supported Kernel (STLS)**
   - User space: App container infrastructure (mid-term)
   - Kernel space: Domain Specific communication (e.g. OPC UA)

2. **Real-time support**
   - Monitoring
   - Real-time / safe virtualization

3. **Test automation**
   - Tracing & reporting tools
   - Configuration management
   - Device management (update, download)
   - Application life-cycle management

4. **Build environment**
   - (e.g. bitbake, dpkg)
   - Tools

5. **Security**
   - Shared config. & logging
   - Multimedia

6. **Safe & Secure Update**
   - Middleware/Libraries

7. **CIP Core Packages**

---

**Tools**

- **Build environment**
- **Test automation**
- **Security**
- **Real-time support**

---

**Concepts**

1. **Long-term support Strategy**: security patch management
   - Functional safety architecture/strategy, including compliance w/ standards (e.g., NERC CIP, IEC61508)

2. Standardization collaborative effort with others
3. Export Control Classification
4. License clearing
CIP governance structure and projects

Governing Board (GB)

Technical Steering Committee (TSC)

CIP Projects and its scopes

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

Security
Sustainability
Industrial grade

(*) Workgroup
The backbone of CIP are the member companies

[Diagram showing budget, developers, maintainers, and open source projects (upstream work) with logos of companies like Renesas, Siemens, Toshiba, CodeThink, cybertrust, Hitachi, and Moxa.]

- Optional: funding of selected projects
- Contribution & usage / integration
- Open Source Projects (Upstream work)
Mapping CIP into the company

- **CIP Core Packages** *(tens)*
- **CIP Kernel** *(10+ years maintenance)*
- **additional packages** *(hundreds)*

**Business Units / Products**

**Companies / Divisions**

**Corporate team / central project**

**“distribution”**

- Domain-specific extensions
- Domain-specific extensions
- ...

**Kernel and base packages, SDK, Build chain, QA**

**Firmware Update**

**Security Hardening**

**Container Runtime**

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.
Upstream First
CIP Kernel Team

• 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
  • SZ Lin – Kernel Developer
  • Chen-Yu Tsai – Kernel Developer
Development Models

“Own Community” Model

The project branches its base from upstream and evolves by its own.

“Upstream First” Model

The project 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

Contribute, Collaborate and use by CIP

Contributing by CIP members as future candidates

1. Upstream first
2. Use the upstream code
3. Integrate

CIP Open Source Base Layer (OSBL)
CIP SLTS kernel development

1. Upstream First
   - Mentor / Maintainer
   - Developers

2. Use
   - CIP SLTS kernel

3. Integrate
   - LTS kernel

Mainline / LTS

CIP kernel team
CIP SLTS kernel development

1. Contribute to upstream projects
2. Use
   - Feature up-streaming
   - First
   - Upstream
   - CIP SLTS kernel
3. Integrate
   - Automated testing
   - Release CIP kernels based on LTS kernels
   - Integrate with userland packages

Open Source Tools

Mentor / Maintainer

Developers

CIP kernel team

Mainline / LTS

LTS kernel
Contributions – Statistics

![Histogram showing contribution counts]

606 for v4.4, 480 for v4.9, 274 for v4.14, 178 for v4.19, 61 for v5.4

as of June 7, 2020
## Contributions - Details

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></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>429</td>
<td>319</td>
<td>137</td>
<td>74</td>
<td>24</td>
<td>983</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>78</td>
<td>81</td>
<td>53</td>
<td>33</td>
<td>14</td>
<td>259</td>
</tr>
<tr>
<td>Acked-by:</td>
<td>26</td>
<td>29</td>
<td>33</td>
<td>43</td>
<td>11</td>
<td>142</td>
</tr>
<tr>
<td>Reviewed-by:</td>
<td>2</td>
<td>4</td>
<td>9</td>
<td>6</td>
<td>3</td>
<td>24</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>103</td>
<td>96</td>
<td>70</td>
<td>49</td>
<td>23</td>
<td>341</td>
</tr>
<tr>
<td><strong>TOTAL / LTS</strong></td>
<td><strong>606</strong></td>
<td><strong>480</strong></td>
<td><strong>274</strong></td>
<td><strong>178</strong></td>
<td><strong>61</strong></td>
<td><strong>1599</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 June 7, 2020
## Use – Current SLTS Versions

<table>
<thead>
<tr>
<th>Version</th>
<th>Maintainer</th>
<th>First Release</th>
<th>Projected EOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4</td>
<td>Nobuhiro Iwamatsu &amp; Pavel Machek</td>
<td>2017-01-17 • v4.4.42-cip1</td>
<td>2027-01</td>
</tr>
<tr>
<td>4.4-rt</td>
<td>Pavel Machek</td>
<td>2017-11-16 • v4.4.75-cip6-rt1</td>
<td>2027-01</td>
</tr>
</tbody>
</table>
Use – CIP Kernel Release Process

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

---

## Use - CIP SLTS Kernel Release Policy

<table>
<thead>
<tr>
<th>Release regularly</th>
<th>Release on demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLTS 4.19: twice a month</td>
<td>Depends on criticality of bug / security fixes</td>
</tr>
<tr>
<td>SLTS 4.4: once a month</td>
<td></td>
</tr>
<tr>
<td>SLTS 4.19-rt: once a month</td>
<td>Ditto</td>
</tr>
<tr>
<td>SLTS 4.4-rt: once every two months</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** Difficult to estimate actual release date because of number of patches depends on each stable release
Use – CIP SLTS Kernel Release Statistics

as of June 7, 2020
Use – CIP SLTS Kernel Releases

Mainline

Stable (linux-stable-4.4)

CIP SLTS 4.4 (linux-4.4.y-cip)

CIP SLTS 4.19 (linux-4.19.y-cip)

Maintained by CIP Kernel Maintainers

Maintained by CIP

End of LTS

End of CIP SLTS

Maintained by CIP

4 years

6 years

4 years

10 years

10 years

6 years
Integrate

Yocto/Poky

Build framework from source code for embedded systems

CIP Source Packages

source packages
CVE patches

source packages
CVE patches

Compile and optimize for embedded devices

Apply CIP Linux

Middleware and applications
additional packages
CIP Core packages
CIP kernel

Debian Source Packages

Build

Binary Packages (deb)

Install

Repository

Debian User

apt/apt-get

User

Use Yocto/Poky
CIP
Open Source Tools
Open Source Tools for Backporting Process

 classify-failed-patches
Project ID: 12455807
- 66 Commits  1 Branch  0 Tags  52.2 MB Files  52.3 MB Storage

 cip-kernel-sec
Project ID: 3169019  Request Access
- 420 Commits  5 Branches  0 Tags  2.7 MB Files  2.7 MB Storage

Analysis by CIP Kernel Team and/or contributors

Backporting needed?

N
Y

ignore

Backporting by CIP Kernel Team

Backporting by CIP Kernel Team
cip-kernel-sec

- Tracks the status of security issues, identified by CVE ID, in mainline, stable, and other configured branches.
Backporting by CIP Kernel Team

Show via Web I/F
- Mainline/LTS
- ubuntu
- debian

Gather CVE Information for Kernel

Show via Command Line
- Webview
- Command line view
cip-kernel-sec Webview

CVE-2020-10720 - kernel: use-after-free read in...

Summary
Kernel: use-after-free read in napi_gro_free() in the Linux kernel

References
http://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/
CVE-2020-10720

Branches
- Linux-3.16-y
- Linux-4.4-y
- Linux-4.4-xmp
- Linux-4.4-xpkert
- Linux-4.9-
- Linux-4.19-
- Linux-4.19-xpkert
- Linux-5.4-
- Linux-5.6-
- mainline

CVE-2019-19073 - Memory leaks in...

Summary
Memory leaks in drivers/net/wireless/ath9k/htc.h in the Linux kernel through 3.11 allow attackers to cause a denial of service (memory corruption) by triggering netfor_compmon,timeout(), failures. This affects the htc_config_is_cpu() functions, the htc_send_complete() function, and the htc_connect_services() function, aka CSD-38347 (9a70).

References
http://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/
CVE-2019-19073

View branches View open issues View issues
classify-failed-patches

• This project tracks the status of failed patches, and classifies patches into “applied” and “ToApply” types.
classify-failed-patches

Stable Patches Archive
- mail for commit

Stable Kernel Tree
- commit log for 4.4 or 4.19

classify-failed-patches

classify patches

Already applied

To be applied
classify-failed-patches

**Applied patches**

```
[AAPPLIED] arm64: Disable unhandled signal log messages by default
  56b57/bd20f5bdcd353af7b70c41e18fffe0c963 arm64: Disable unhandled signal log messages by default
[AAPPLIED] ARC: hide unused function umw_hdr_alloc
  4d265152fcaaf456df656e80000018f080000 ARC: hide unused function umw_hdr_alloc
[AAPPLIED] btrfs: Ensure replaced device doesn’t have pending chunk allocation
  986543f508a3631be44c4ac2dc48f8e25ab34 btrfs: Ensure replaced device doesn’t have pending chunk allocation
[AAPPLIED] btrfs: Ensure replaced device doesn’t have pending chunk allocation
  986543f508a3631be44c4ac2dc48f8e25ab34 btrfs: Ensure replaced device doesn’t have pending chunk allocation
[AAPPLIED] btrfs: Ensure replaced device doesn’t have pending chunk allocation
  986543f508a3631be44c4ac2dc48f8e25ab34 btrfs: Ensure replaced device doesn’t have pending chunk allocation
[AAPPLIED] btrfs: Ensure replaced device doesn’t have pending chunk allocation
  986543f508a3631be44c4ac2dc48f8e25ab34 btrfs: Ensure replaced device doesn’t have pending chunk allocation
```

**To be Applied Patches**

```
[TOAPPLY] inet: update the IP ID generation algorithm to patch 355b985578b8b64ed97a8d801a619ff898471b92 standards.
[TOAPPLY] scsi: ufs: Fix RX_TERMINATION_FORCE_ENABLE define value
[TOAPPLY] inet: update the IP ID generation algorithm to patch 355b985578b8b64ed97a8d801a619ff898471b92 standards.
[TOAPPLY] iio/hf11: Failed to drain send queue when QP is put into error state
[TOAPPLY] arm64: mm: Ensure tail of unaligned inrdr is reserved
[TOAPPLY] fs/proc/task_mmu.c: fix uninitialized variable warning
[TOAPPLY] tpm: Fix the type of the return value in calc_tpm2_event_size()
[TOAPPLY] block: bio_map_user_iov should not be limited to BIO_MAX_PAGES
[TOAPPLY] clk: ingenic/jz4725bs Fix parent of pixel clock
[TOAPPLY] i2c-piix4: Add Hygon Dhyana SMbus support
[TOAPPLY] tty: serial core, add -dinstall
```
Necessity of backporting is determined to be fixed base on kernel configurations provided by CIP members.
CIP
Automated Testing
CIP Testing Goals

• Centralized control / distributed testing
  • CIP developers who are distributed over the world should be able to test CIP reference platforms which are hosted at 4 labs located in Europe, India and Japan.

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

• Support all CIP reference platforms
  • There are currently 7 different reference platforms
Upstream First

Use

Integrate

CIP Testing WG

lava-docker

lab-ciplatform.org

lab-cip-cybertrust

lab-cip-denk

lab-cip-mentor

lab-cip-renesas

CIP Testing Team
CIP Testing Team

Upstream First

1. Fund to KernelCI
2. Use Upstream Code & Code Reviews
3. Integrate
   - Test rc versions before kernel releases

Use automated testing systems

CIP Testing WG

lava.ciplatform.org

lab-cip-cybertrust
lab-cip-denk
lab-cip-mentor
lab-cip-renesas
## Testing Architecture Overview

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

**Location key**
- GitLab.com
- AWS EC2
- AWS EC2 on-demand
- Local

**CIP Reference Hardware**
# CIP Reference Boards

## CIP Reference Boards

<table>
<thead>
<tr>
<th>Platform</th>
<th>Architecture</th>
<th>Supported Kernels</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM335x Beaglebone Black</td>
<td>Armv7</td>
<td>SLTS v4.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SLTS v4.4-rt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SLTS v4.19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SLTS v4.19-rt</td>
</tr>
<tr>
<td>Cyclone V DE0–Nano–SoC Development Kit</td>
<td>Armv7</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y¹</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y¹</td>
</tr>
<tr>
<td>QEMU</td>
<td>x86_64</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y¹</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y¹</td>
</tr>
<tr>
<td>RZ/G1M iWave Qseven Development Kit</td>
<td>Armv7</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y¹,²</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y¹,²</td>
</tr>
<tr>
<td>RZ/G2M HopeRun HiHope</td>
<td>Armv8</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y¹,²</td>
</tr>
<tr>
<td>SIMATIC IPC227E</td>
<td>x86-64</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y¹,²</td>
</tr>
<tr>
<td>OpenBlocks IoT VX2</td>
<td>x86-64</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y¹,²</td>
</tr>
</tbody>
</table>

## CIP Reference Board Candidate

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

¹ Tested with standard Kernel configuration (non-RT)
² Tested with Real-Time enabled Kernel configuration
Automated Testing

• Currently CIP is running the following tests:
  • Boot test
    • `uname -a`
  • Spectre/Meltdown checker
  • 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`
  • Cyclictest+Hackbench
    • This test measures event latency in the Linux Kernel, with hackbench running in the background to stress the system.

• In Development:
  • Kselftest
Collaboration with KernelCI

- **kernelci.org** is now a Linux Foundation project, sponsored by Baylibre, CIP, Collabra, Foundries.io, Google, Microsoft and Redhat: [https://foundation.kernelci.org/](https://foundation.kernelci.org/)

- CIP are collaborating with KernelCI to improve the range of tests supported by KernelCI, starting with LTP

- Further collaboration is being discussed between CIP and KernelCI
Summary
Summary

- CIP Kernel Team follows “Upstream First” principle, and contributes to upstream.
- CIP open source tools are developed to facilitate the contribution activities.
- 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 Repository

• 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
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