Safe and Secure Field Updates of Embedded Linux Systems

Enrico Jörns – e.joerns@pengutronix.de

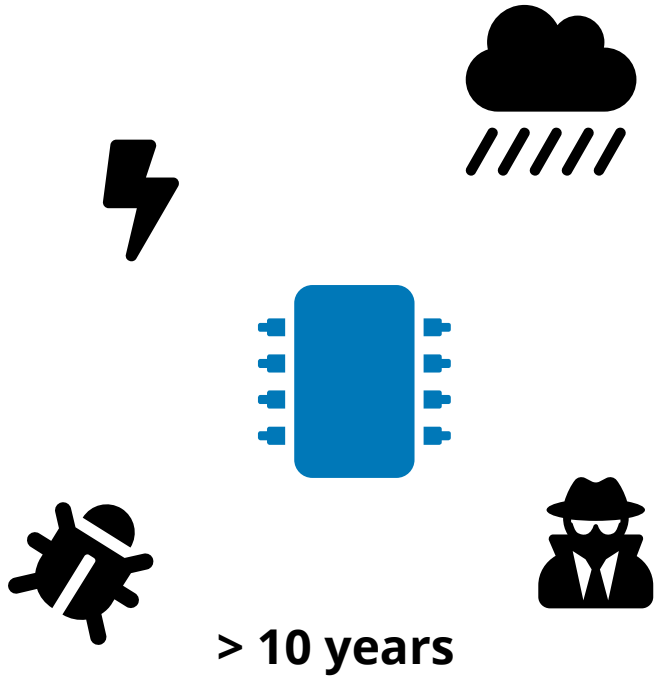
About Me

- Embedded software developer
- Co-maintainer of FOSS update framework RAUC
- At Pengutronix since 2014

- Embedded Linux consulting & support since 2001
- > 4500 patches in Linux kernel

Updating Embedded Systems

- Unattended / remote
- Long life time
- Insecure physical environment
- CVEs, bugs
- Technology changes

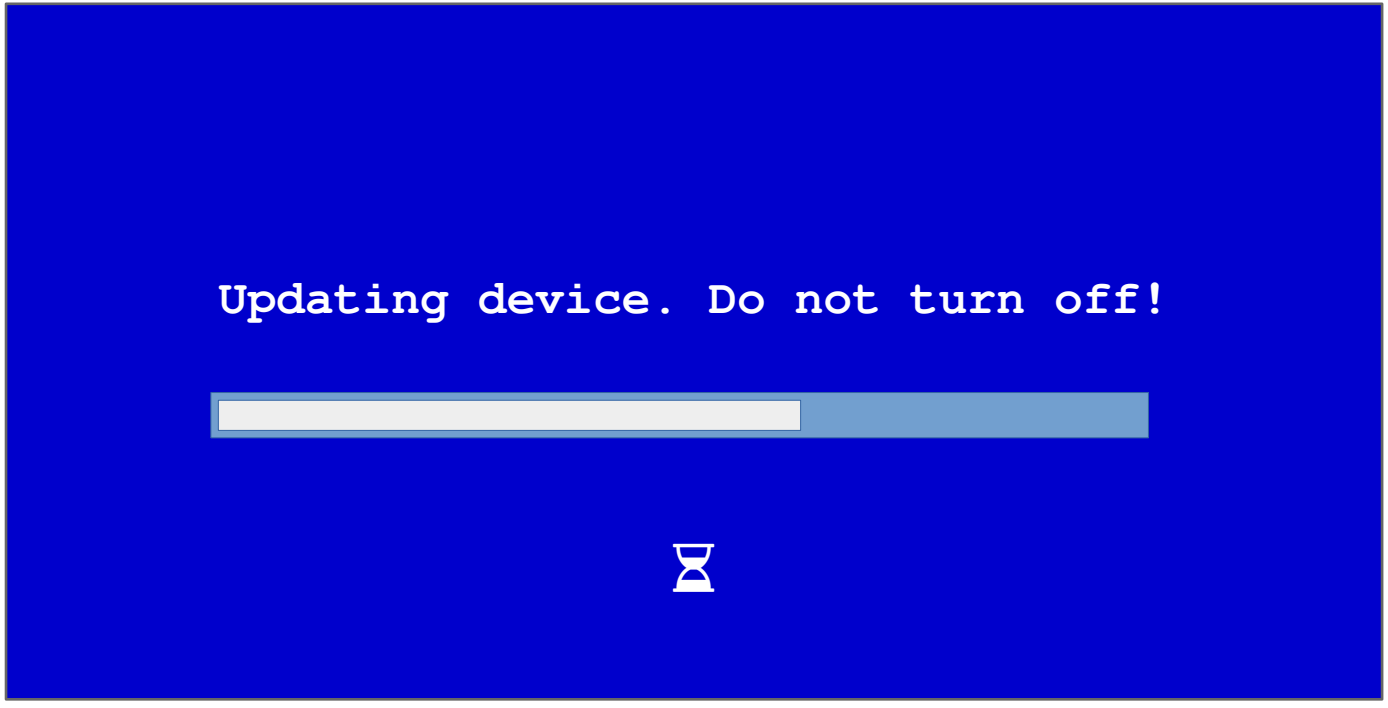


→ Requires robust and fail-safe update & system design

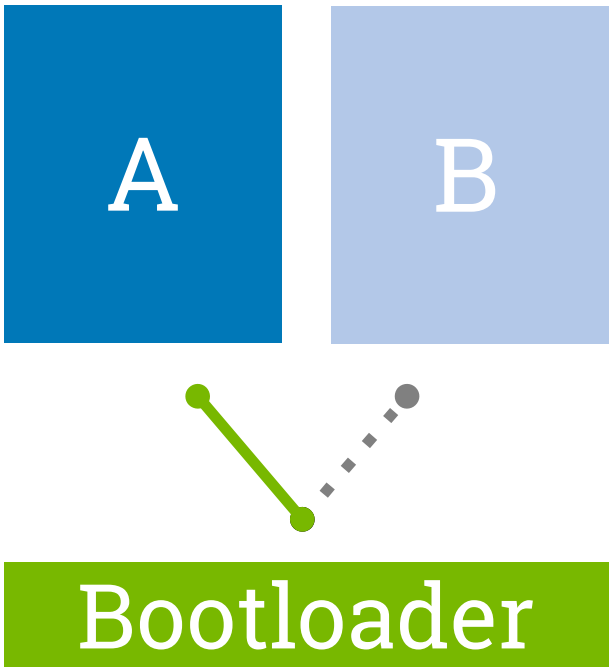
Image-Based Updating

- ✖ Package-based updates inappropriate
  - ✖ Require interactive administration
  - ✖ Conflicts
  - ✖ Untested combinations
  - ✖ Affected by file system corruption
- ✔ Full image-based system updates
  - ✔ Well-defined state (reproducibility!)
  - ✔ Well-tested state of application + software

Fail-Safe Updating



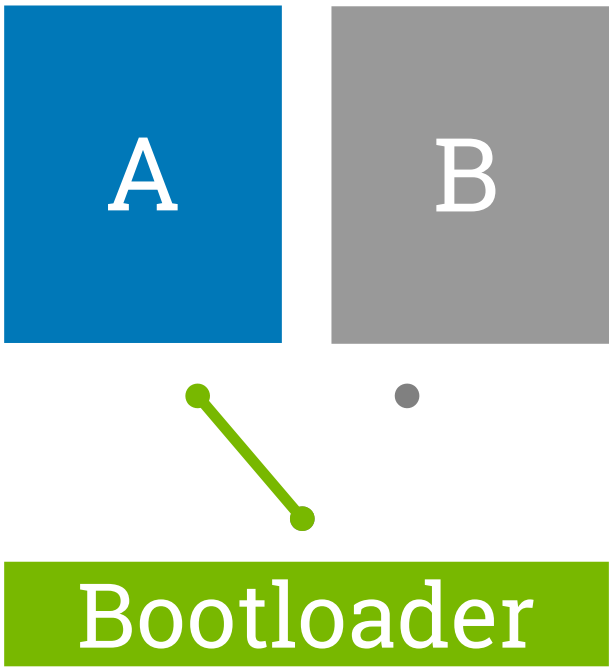
Fail-Safe Updating



A: Active (running) system

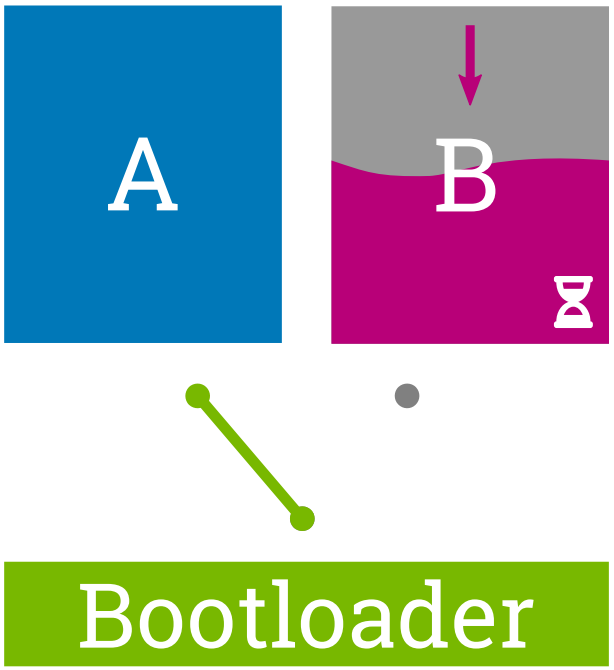
B: Non-running system

Fail-Safe Updating



- Deactivate partition to update

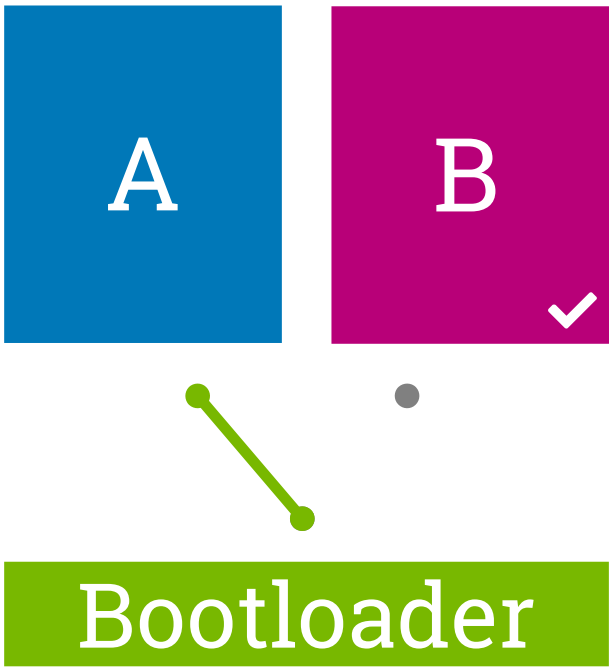
Fail-Safe Updating



- Deactivate partition to update
- Write update(s) to disk

  Critical Operation!

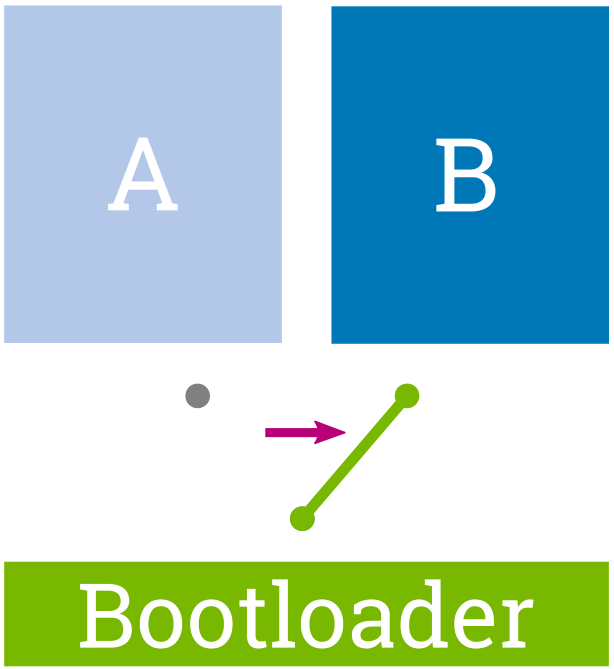
Fail-Safe Updating



- Deactivate partition to update
- Write update(s) to disk

  Critical Operation!
- Update fully completed + verified, etc.

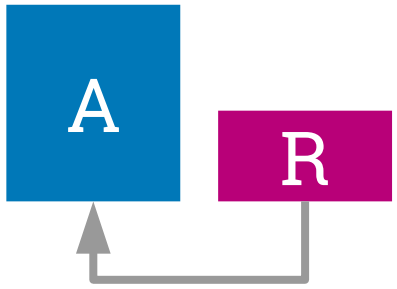
Fail-Safe Updating



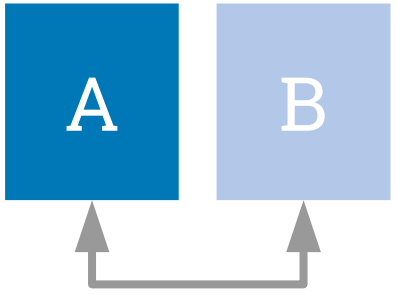
- Deactivate partition to update
- Write update(s) to disk

  Critical Operation!
- Update fully completed + verified, etc.
- Activate updated slot

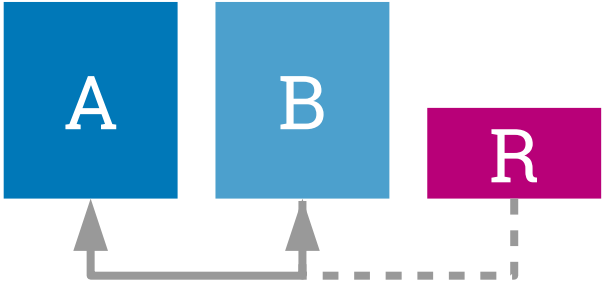
Redundancy – Variants



- ✖ Downtime
- ■ Fallback
- ✔ Disk space

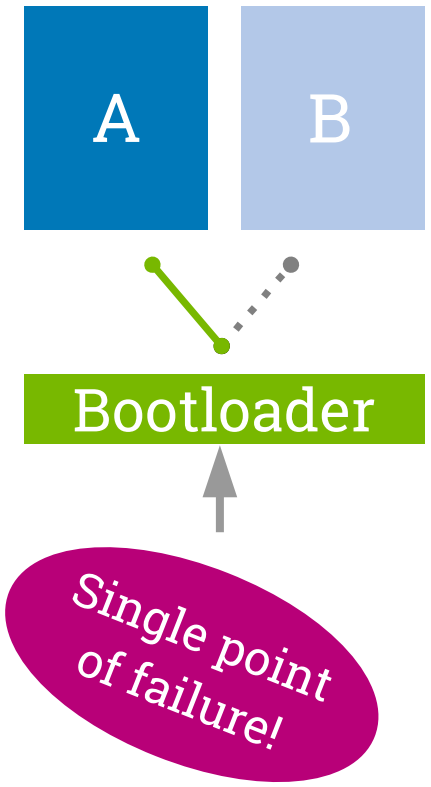


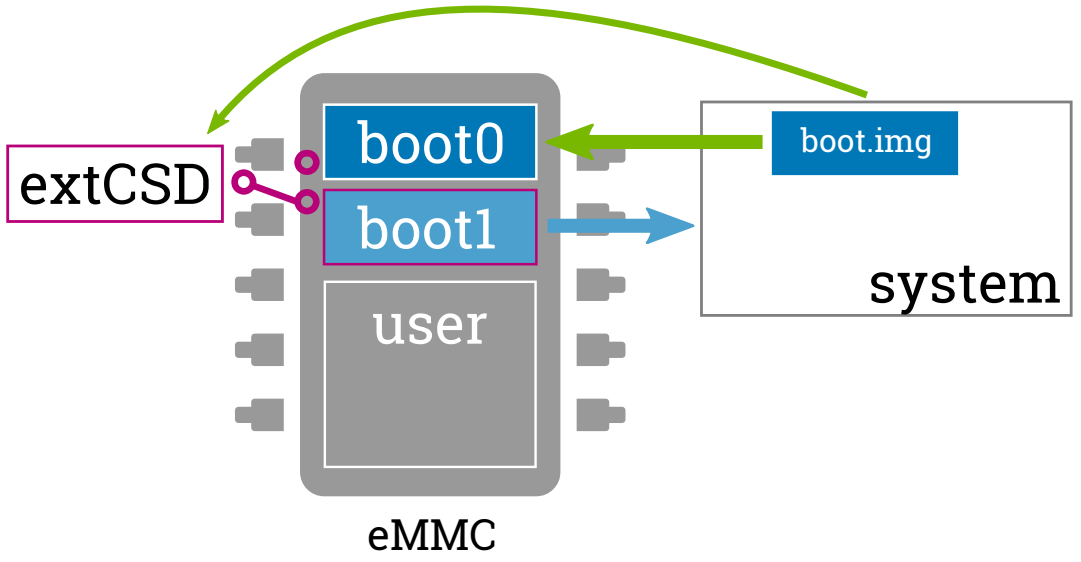
- ✔ No downtime
- ✔ Fallback
- ✖ Disk space



- ✚ Extra robust

Bootloader Updates – eMMC

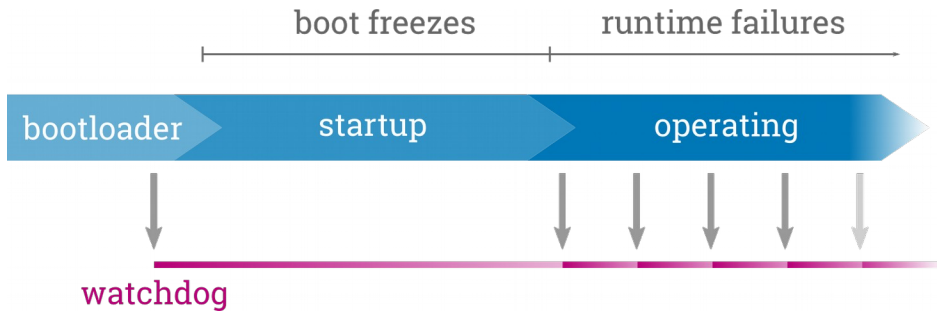




- Provides 2 built-in boot partitions
- Selectable by extCSD register
- → Allows atomic bootloader update

Failure Detection & Fallback

- Watchdog to detect fatal crashes + hangs
- Optional: Fallback after n failed attempts



- When is a system considered booted successfully?
- How to detect faulty runtime behaviour?

systemd – System and Service Manager

example.service:

```
[unit]
Description=Critical application

[Service]
ExecStart=/usr/bin/myapp
WatchdogSec=30s
Restart=on-failure
StartLimitInterval=5min
StartLimitBurst=3
StartLimitAction=reboot-force
...
LimitNPROC=32
...
ProtectSystem=strict
```

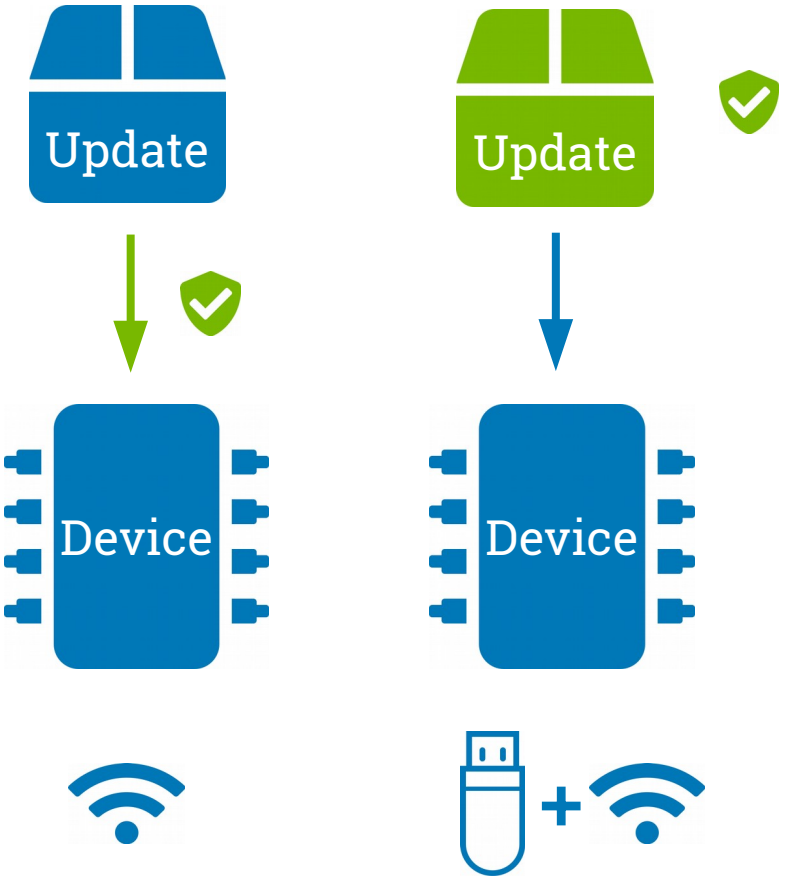
- Watchdog multiplexer & application control
- Detect dying + hanging services
- Resource limiting
- Encapsulation + hardening features

Updating – Authentication

With great power

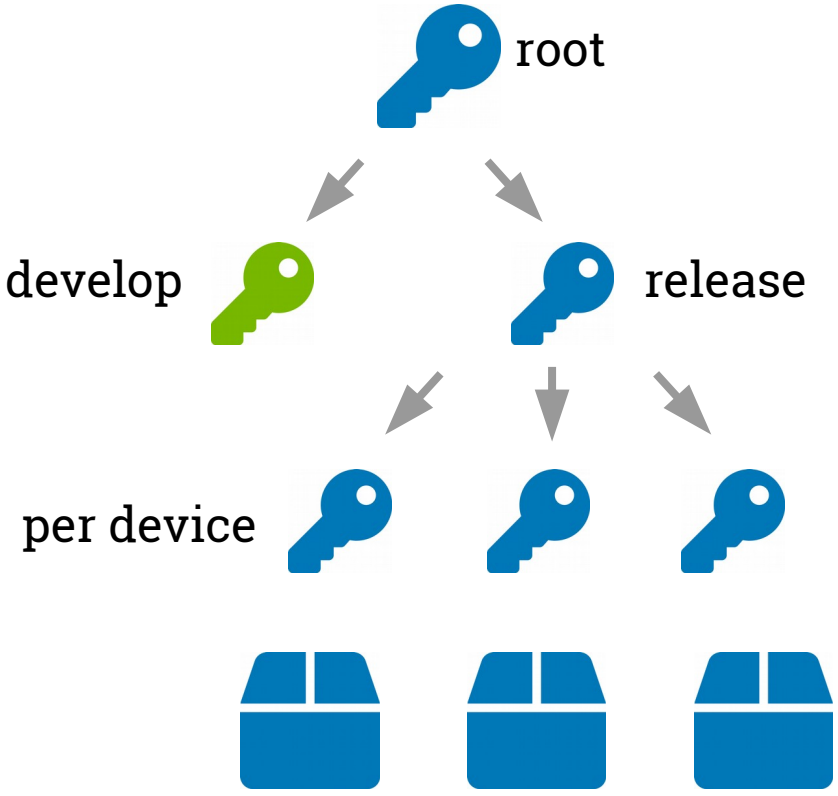
comes great responsibility!

Updating – Authentication



- Prevent unauthorized access!
- Signing the update artifact
- Verification on target
- Use well-proven & open source crypto (OpenSSL)

Authentication – X.509 PKI

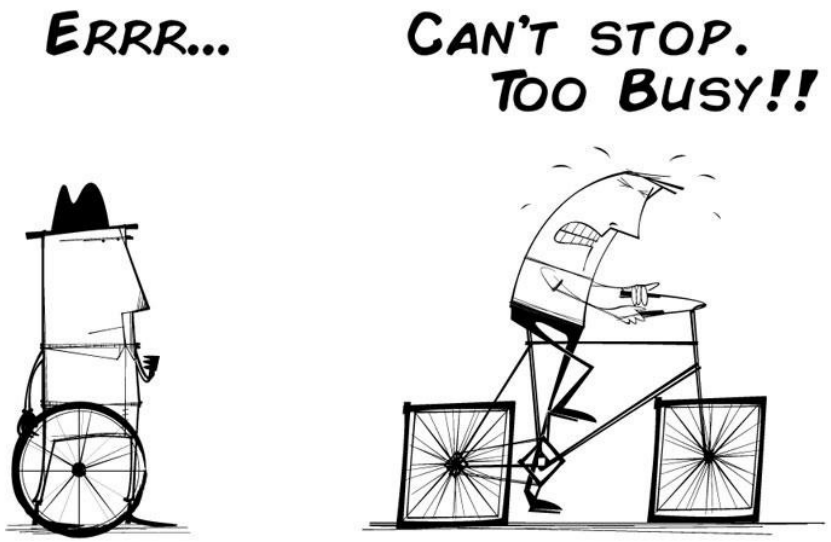


- Self-signed (for testing!)
- Development vs. release key
- Per-device keys
- Multiple signer
- Replace and revoke keys

A New CVE For Our Platform Came Up!

When can we deploy the fix?

Software Stack – Technical Debt



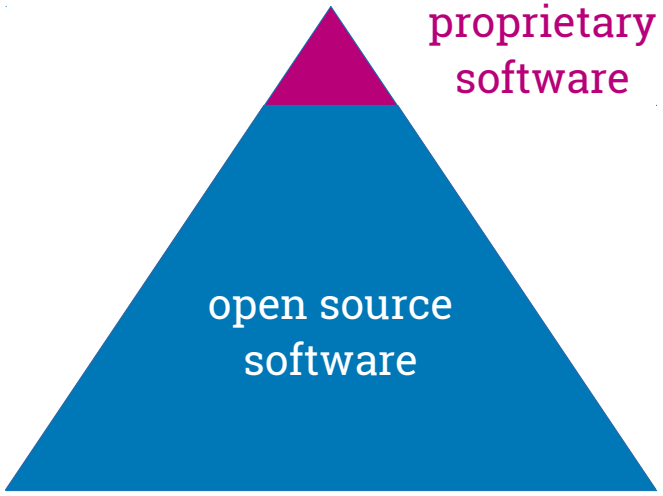
- Project-specific modifications
- Deprecated / unmaintained software

➔ unable to react!

👍 Resolve using open source tools & workflows

https://medium.com/@adamdonaghy/in-defence-of-tech-debt-dc9595e0d316

Software Maintenance



Embedded Linux platform

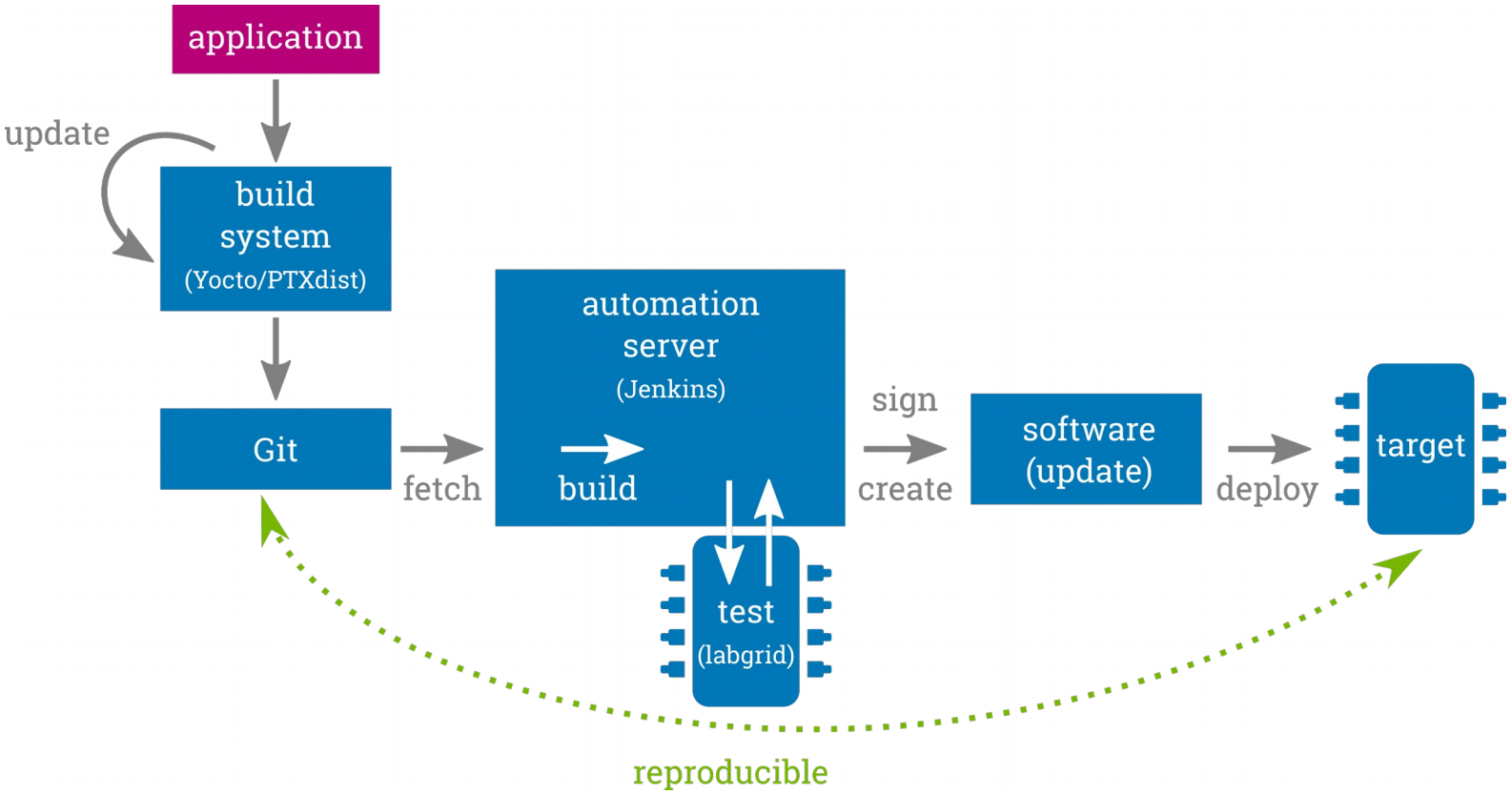
- Start with latest versions
- Push changes upstream
- Regular update cycles
- Use embedded Linux build systems

yocto PROJECT · PTX DIST

- Automated testing (on hardware)

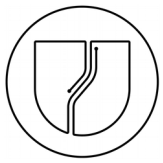
👍 > 90% of all critical bugs fixed by the open source communities

Automation & Reproducibility



Using Open Source Update Tools

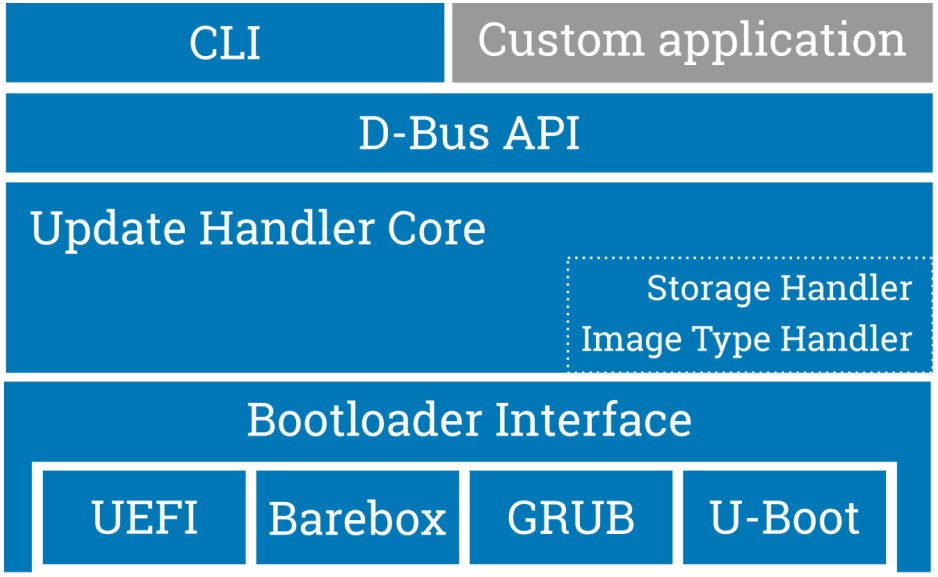
- Home-grown solutions error-prone!
- Less effort, less costs
- Stability through community review and testing
- New features ‘for free‘
- Use development time for new features instead of reinventing the wheel!



swupdate

FOSS Update Tools – RAUC

- LGPLv2
- C + glib + OpenSSL
- Flexible and extensible
- X.509 (CMS), PKCS#11
- Application-controlled update process
- Binary delta update streaming (casync)



RAUC

FOSS Update Tools – RAUC

```
[system]
compatible=MyProduct2000
bootloader=barebox

[keyring]
path=/etc/rauc/keyring.pem

[slot.rootfs.0]
device=/dev/sda0
type=ext4
bootname=system0

[slot.rootfs.1]
device=/dev/sda1
type=ext4
bootname=system1

[...]
```

Target device: RAUC system configuration

```
[update]
compatible=MyProduct2000
version=2019.02-4
build=20190228134503

[image.rootfs]
filename=rootfs.ext4
size=419430400
sha256=b14c1457dc1046...

[image.appfs]
filename=appfs.ext4
size=219430400
sha256=ecf4c031d01cb9...
hooks=post-install
```

Update Bundle: manifest

Conclusion

- Safe & secure updating is complex
- Use open source update frameworks!
- A good overall system design is the key to robustness
- systemd provides good tooling for fail-safe system designs
- Eliminate technical debt by using open source software and workflows!

Thank you!

Questions?

Pengutronix e.K.
H. 4-261

https://www.pengutronix.de

References

Yocto Project – System update:

https://wiki.yoctoproject.org/wiki/System_Update

RAUC system update documentation:

https://rauc.readthedocs.io/en/latest/

systemd watchdog handling

http://0pointer.de/blog/projects/watchdog.html

Labgrid – Embedded boards control for testing automation

https://labgrid.readthedocs.io/en/latest/

RAUC on GitHub:

https://github.com/rauc/rauc

Updating And Verified Boot