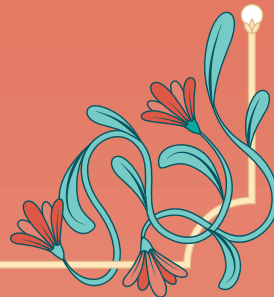


An Approach To Managing CVEs With Vendor Kernels

Jan Pfiffner, Grossenbacher Systeme AG

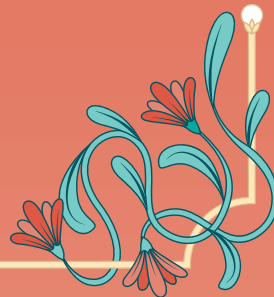
Slides: CC BY 4.0



An Approach To Managing CVEs With Vendor Kernels

Jan Pfiffner, Grossenbacher Systeme AG

...with Yocto



About me & the code

Facts

- Head of Software Engineering @ Gesys
- Focusing on embedded security: secure boot, bootloaders, secure updates, CRA
- When not developing I spend time with my family or do stuff for OpenAir St.Gallen
- Opinions are my own and do not reflect those of my employer and nothing is meant as legal advice

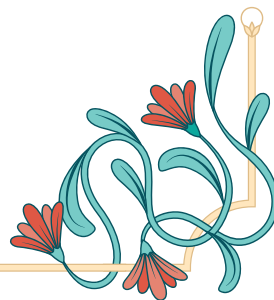
Contact

- jan@pfiffner.io
- <https://pfiffner.dev>

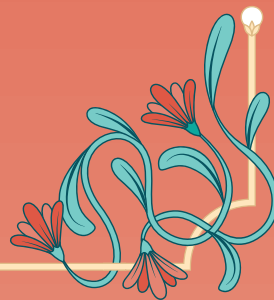


<https://gitlab.pfiffner.dev/yocto/>

- [kernelcve-elc-setup](#)
- [meta-kernelcve-test](#)
- [buildcontainer_aarch64](#)



THE PROBLEM

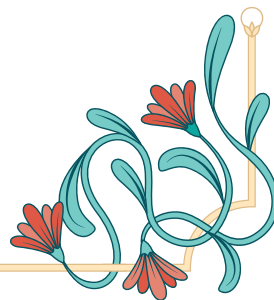


The good news (for some)

- Kernel is its own CNA since 2024 -> every fix can become a CVE
- Thousands of kernel CVEs per year
- On linux-yocto it got "easy": oe-core 596ad6c (May 2026) generated CVE exclusion list + generate-cve-exclusions.py removed -- sbom-cve-check reads the kernel CNA data from cvelistV5 itself

But:

- On a vendor kernel: do_cve_check says ... 725 unpatched
- Vendor BSPs are on LTS (scarthgap), not master -> this doesn't reach you



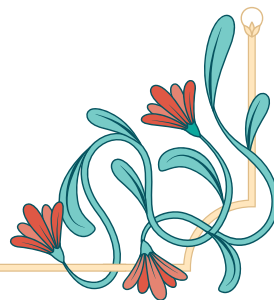
Option 1, 2, 3: don't.

How to handle CVEs on a vendor kernel:

1. Don't -- if you can avoid a vendor kernel (vendors are moving closer to upstream, good)
2. Don't -- use upstream / linux-yocto
3. Don't -- buy from a vendor that already does it for you, or hope yours is upstream very soon
4. You have to. -> rest of this talk



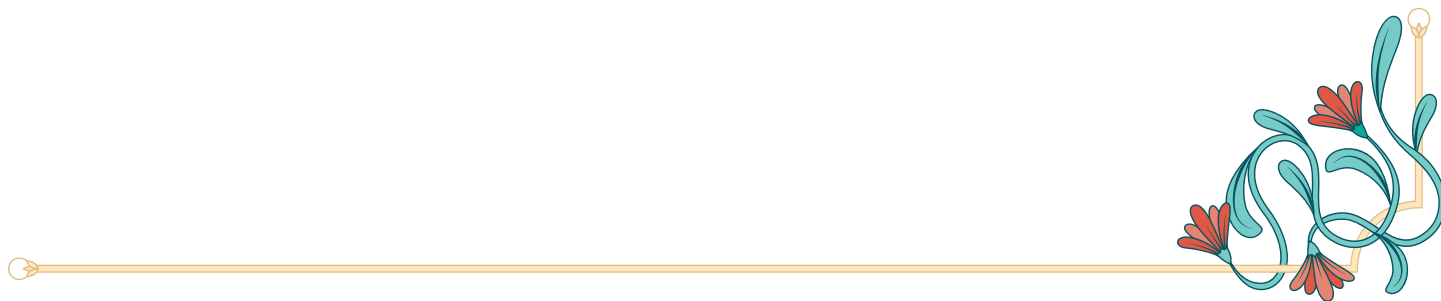
"Seinfeld" © Castle Rock Entertainment / Sony Pictures Television



Option 4 is edgy

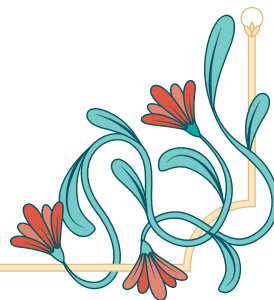
"Don't cherry-pick CVE fixes, take the whole stable release." -- the kernel community's position (paraphrased, Greg K-H)

- Vendor delta makes rebasing onto stable impractical
- So we end up close to exactly that
- Close -- but not the same



Where it actually hurts

- Volume: detecting is not fixing -- each CVE needs a decision
- Vendors: updates late, partial, or never
- Tooling: built around mainline/stable -- "truth" is the upstream kernel, not your vendor tree



THE IDEA: RISK, NOT COUNT

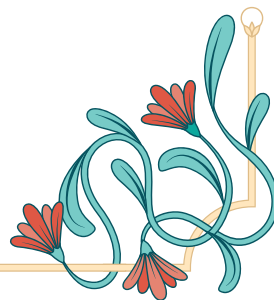


Most reported CVEs aren't in your device

- Embedded kernels are minimal configs
- CVE in SMB client? How many of your devices mount SMB shares?
- The question isn't "is this kernel version vulnerable" but "is THIS build vulnerable"

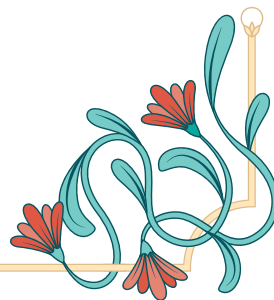


"Star Wars" © Lucasfilm Ltd.



CRA: risk-based, not patch-everything

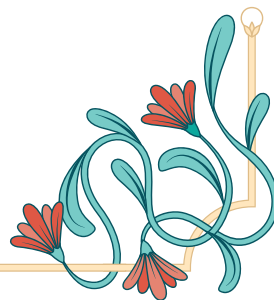
- CRA asks for risk assessment, defense in depth, due diligence
- "Without known exploitable vulnerabilities" != "zero open CVEs (or EUVD IDs)"
- With a proper risk analysis there is no need to patch every CVE
... especially now that nearly every kernel bug fix gets a CVE
- But: every "not affected" needs a reason you can show



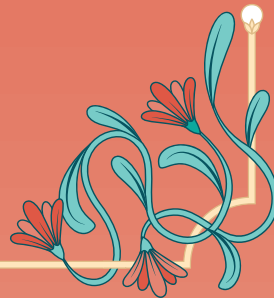
Three levers

1. Severity & attack vector -> what matters for THIS device
2. Find the real fix -> on the matching stable branch
3. Automate + human review -> output you can commit to Yocto, and keep

Prior art on 2nd: Takuma Kawai, OSS Japan 2022
(ossjapan2022.sched.com/event/1D14m)



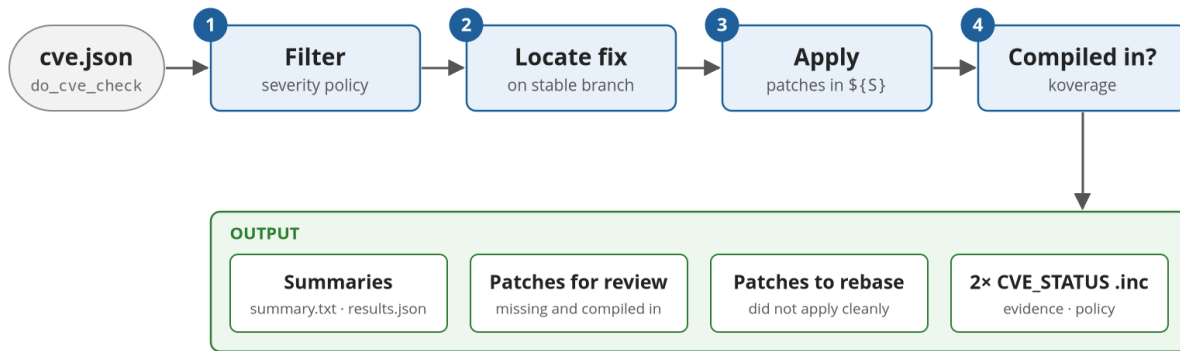
WHAT THE LAYER DOES



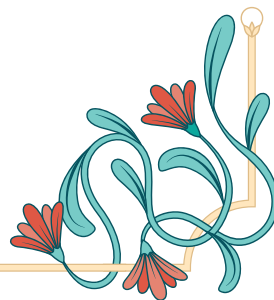
Four steps, one task

meta-kernelcve-test (scarthgap, attaches to virtual/kernel)

validated with: Variscite i.MX8M Plus (6.6) and Jetson Orin Nano / meta-tegra (5.15)



```
$ bitbake -c process_kernel_cves virtual/kernel
```



Step 1: Filter by severity policy

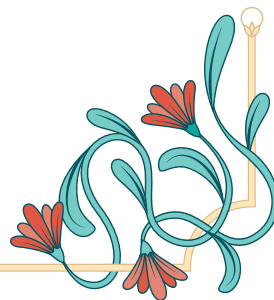
KERNEL_CVE_SEVERITY_POLICY ?= "N:7.0 A:7.0 L:9.7 P:9.7"

KERNEL_CVE_SEVERITY_UNSCORED ?= "check"

- Per CVSS attack vector: minimum base score, or "check" / "ignore"
- Unscored -> checked (fail safe, nothing dropped silently)
- Set-aside CVEs are recorded with their reason



"The Lord of the Rings" © New Line Cinema



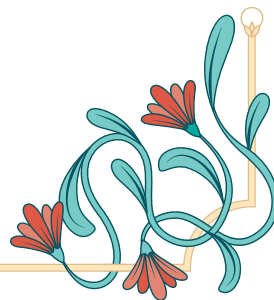
Step 1: Filter by severity policy

KERNEL_CVE_SEVERITY_POLICY ?= "N:7.0 A:7.0 L:9.7 P:9.7" *Notice what I did here?*
KERNEL_CVE_SEVERITY_UNSCORED ?= "check"

- Per CVSS attack vector: minimum base score, or "check" / "ignore"
- Unscored -> checked (fail safe, nothing dropped silently)
- Set-aside CVEs are recorded with their reason



"The Lord of the Rings" © New Line Cinema



Step 1: Filter by severity policy

KERNEL_CVE_SEVERITY_POLICY ?= "N:7.0 A:7.0 L:9.7 P:9.7" *Notice what I did here?*
KERNEL_CVE_SEVERITY_UNSCORED ?= "check"

- Per CVSS attack vector: minimum base score, or "check" / "ignore"
- Unscored -> checked (fail safe, nothing dropped silently)
- Set-aside CVEs are recorded with their reason

Max. possible CVSS 3.1 base score:

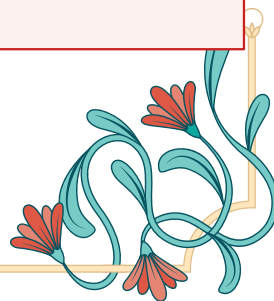
AV:L = 9.3, AV:P = 7.6

-> thresholds of 9.7 effectively ignore local and physical attack vectors.

Deliberate choice for this demo, not general advice.



"The Lord of the Rings" © New Line Cinema

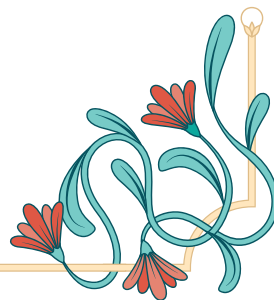


Step 2: Locate the fix on the stable branch

Ubuntu CVE tracker: break-fix: <introduced> <fixed>

Against linux-<x.y>.y:

- `git merge-base --is-ancestor <break> HEAD` -> code_not_present?
- `git merge-base --is-ancestor <fix> HEAD` -> already_fixed_in_base?
- `git log --grep=<fix>` -> backport found -> patch
else no_fix_on_branch

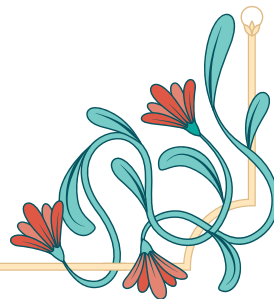


Step 3: Apply against the real source

In `${S}` -- the tree you actually build:

- `git apply <patch>` -> applied (fix was missing)
- `git apply --check --reverse <patch>` -> already there (vendor has it)
- Neither -> `can_not_apply` (human)

PATCHES FOR THIS CVE	VERDICT	NEXT
all already in <code>\${S}</code>	→ <code>already_patched</code>	done, vendor has it
some already there, rest applied	→ <code>partially_patched</code>	→ step 4 (koverage)
all applied	→ <code>continue</code>	→ step 4 (koverage)
one fits neither way	→ <code>can_not_apply</code>	→ human rebase



Step 4: Is the code even compiled? (koverage)

koverage (kmax, Paul Gazzillo et al.) with `${B}/.config`

For every line the fix touches:

`FILE_EXCLUDED`

`LINE_EXCLUDED_FILE_INCLUDED`

otherwise

file not built

behind a disabled `#ifdef`

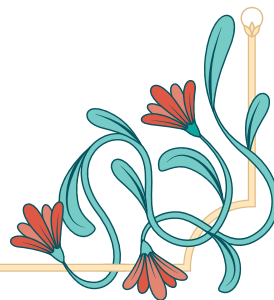
reachable

`>= 1` reachable line

no reachable lines

-> `affected_and_compiled`

-> `not_compiled` (evidence for `CVE_STATUS`)



What you get

- summary.txt
- system_affected_cve_patches/
- needs_manual_rebase/
- cve-exclusion-detailed_<ver>.inc
- cve-policy-severity_<ver>.inc
- results.json

the overview, one page

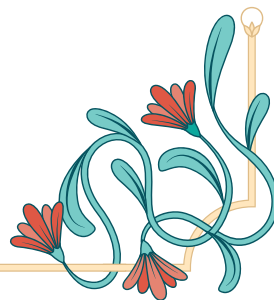
missing AND matter -> review & carry

did not apply, with git error

CVE_STATUS backed by evidence

CVE_STATUS by policy decision

everything, queryable (jq)

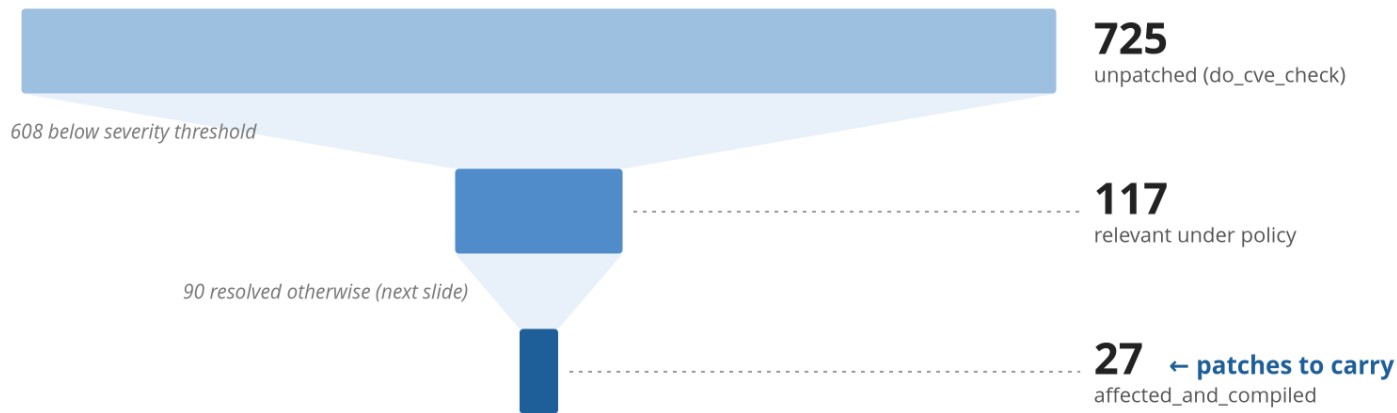


RESULTS & LIMITATIONS



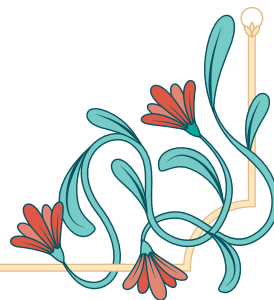
Real run: i.MX8M Plus, vendor 6.6.144

Variscite DART-MX8M-PLUS, linux-variscite-fslc-imx 6.6.144
vs linux-6.6.y @ v6.6.157



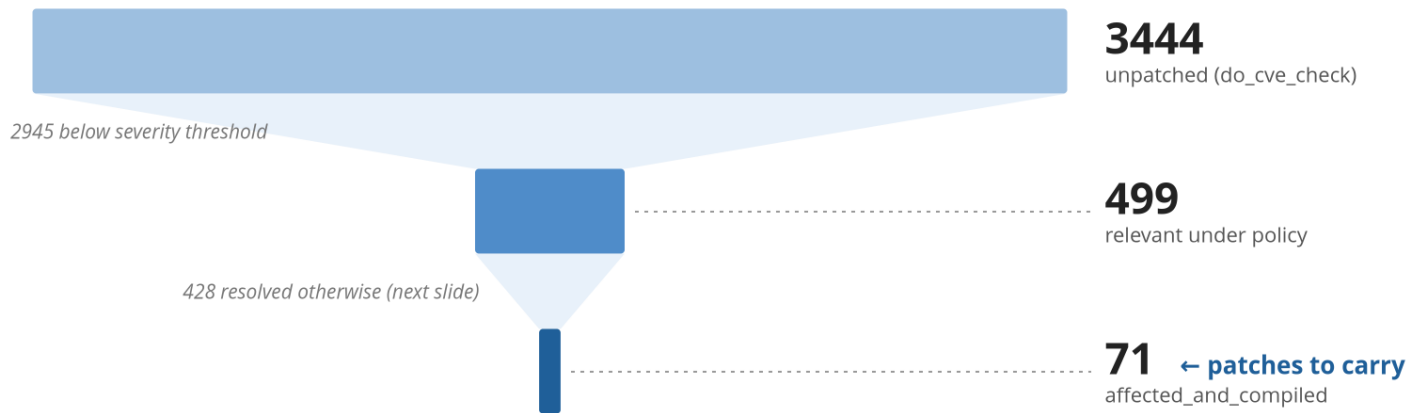
The other 90 (i.MX8M Plus)

no_fix_on_branch	46	fix upstream, never backported to 6.6.y
code_not_present	22	bug introduced after branch point
can_not_apply	7	-> needs_manual_rebase/
not_compiled	5	koverage: not built here
already_fixed_in_base	3	
no_breakfix_data	3	(old CVEs)
deferred	2	
koverage_impossible	2	



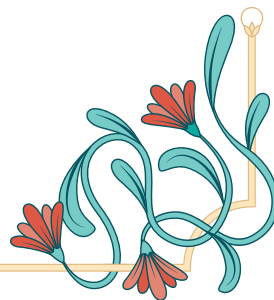
Real run: Jetson Orin Nano, vendor 5.15.185

jetson-orin-nano-devkit, linux-jammy-nvidia-tegra 5.15.185
vs linux-5.15.y @ v5.15.221



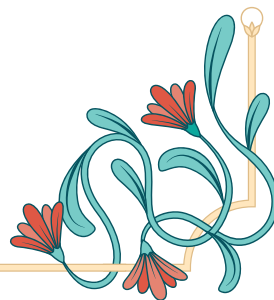
The other 428 (Jetson)

no_fix_on_branch	152	fix upstream, never backported to 5.15.y
not_compiled	145	coverage: not built here
code_not_present	57	bug introduced after branch point
can_not_apply	38	-> needs_manual_rebase/
already_patched	18	vendor already has the fix
no_breakfix_data	6	(old CVEs)
partially_patched	5	
coverage_impossible	4	
already_fixed_in_base	2	
deferred	1	



Limitations

- Depends on Ubuntu CVE tracker break-fix data
- CVSS for kernel CVEs: NVD scores generically, kernel CNA scores none
- "Compiled" != "reachable" -- koverage answers the first only
- Patches still need human review (and testing!)
- scarthgap only so far
- Not a replacement for tracking stable



Takeaways & discussion

- If you can avoid a vendor kernel: do.
- If you can't: judge the build, not the version number
- Verified, evidence-backed patching -- not a cherry-picked list

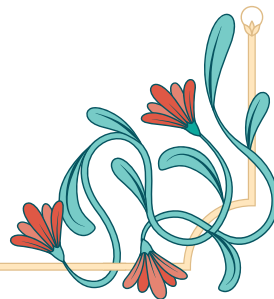
Let's discuss:

- How do YOU handle CVEs on vendor kernels?
- Should something like this live in oe-core / meta-security?



`gitlab.pfiffner.dev/yocto/
meta-kernelcve-test`

Questions?



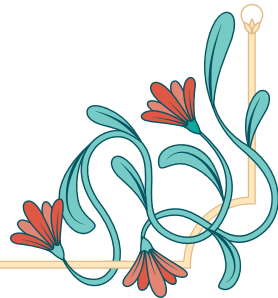
Thank you



"School of Rock" © Paramount Pictures

Contact: jan@pfiffner.io

Slides licensed under CC BY 4.0
creativecommons.org/licenses/by/4.0
Third-party images and logos remain the
property of their respective owners.



THE LINUX FOUNDATION



**EMBEDDED
LINUX
CONFERENCE**

EUROPE

