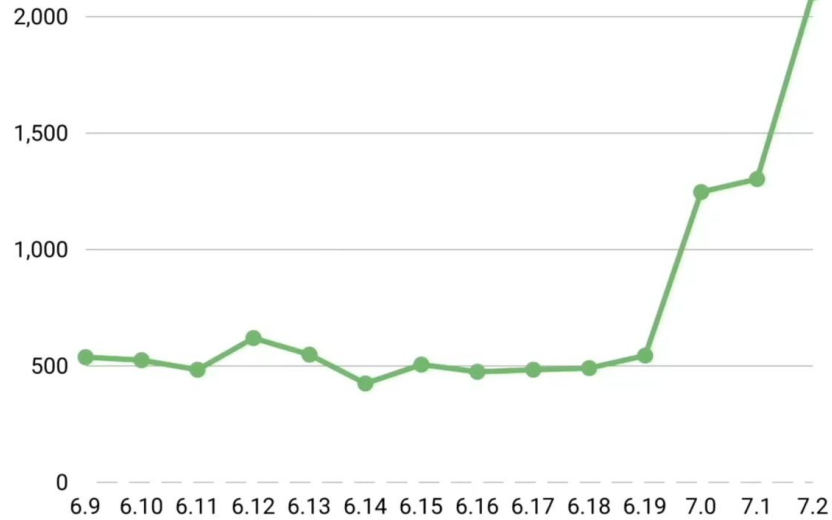


Linux CVEs, vulnerability management and agentic security workflows

Context

- What a year it has been, huh?
- In the past few years, we discussed how to collaborate on CVE triaging
 - <https://github.com/cloud-lts/linux-cve-analysis/>
 - CVSS vs no CVSS, what fields do we need in the analysis, etc...
- This [morning's talk](#) by AWS provides great context too
- ... Not much of an interesting problem anymore?

CVE fixes per release



From [Greg's talk at Kernel Recipes](#)

Defenders vs Attackers

- CVEs are assigned after a fix gets to stable and the CNA votes
- $O(\text{weeks})$ after it was public on mainline
- In parallel, the bad guys could:
 - Track mainline
 - Agentically scan every single commit
 - Try to exploit what seems to be an interesting fix
 - ... in $O(\text{hours})$
 - *(or maybe they are the ones who found the bug in the first place and want to publish a blog post about it ;))*

Some examples

- A few LPEs:
 - "Copy Fail" (CVE-2026-31431)
 - "Dirty Frag" (CVE-2026-43284 / 43500)
 - "Fragnesia" (CVE-2026-46300)
 - Bug also...
 - "ITScape" (CVE-2026-46316)
 - "Januscape" (CVE-2026-53359)
 - "Zapscape" (CVE-2026-64561)
- First publicly exploited KVM escapes
→ doomsday for a cloud provider

Focus shift

- CVEs should no longer be the sole focus of security teams
- Companies all started building the *same* agentic pipelines:
 - Scan code with \$llm + \$harness
 - Triage potential vulnerabilities
 - Attempt to reproduce them according to a threat model
 - Generate fixes
 - ... ?

A few open questions

- Does anyone have a different perspective on the problem?
- Is there an opportunity to collaborate on agentic workflows?
 - How to validate that they do what you expect?
- How to avoid overloading maintainers with slop reports?
 - Report deduplication
 - Database deduplication? :)
- How can products be secured in a timely manner against public exploits?
- What is this BoF going to look like next year?

A possible answer: Akrites

- Initiative backed by the Linux Foundation and a number of organizations
- Shared Security Incident Response Team (SIRT) for critical OSS
- Goals:
 - Validate & deduplicate LLM-found bugs *before* reporting
 - "Contribute-to-participate": members provide engineering & compute to reproduce, fix, and test
 - Confidential coordination so critical fixes are tested and ready to deploy at disclosure
- But: what works for generic OSS won't work for Linux, we need kernel-specific tooling