THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
KOREA

서울

From CVEs to Compliance: Automating Embedded Linux Kernel Security

Kyungsik Lee
LG Electronics

Contents

Why Security Response Now Matters

🇪🇺 EU CRA
- Secure development and vulnerability handling throughout the product lifecycle

🇬🇧 UK PSTI
- Security requirements for consumer connected products

🇺🇸 🇯🇵 Cybersecurity Labels
- Security labels help buyers identify more secure products

Security response is now an ongoing engineering process.

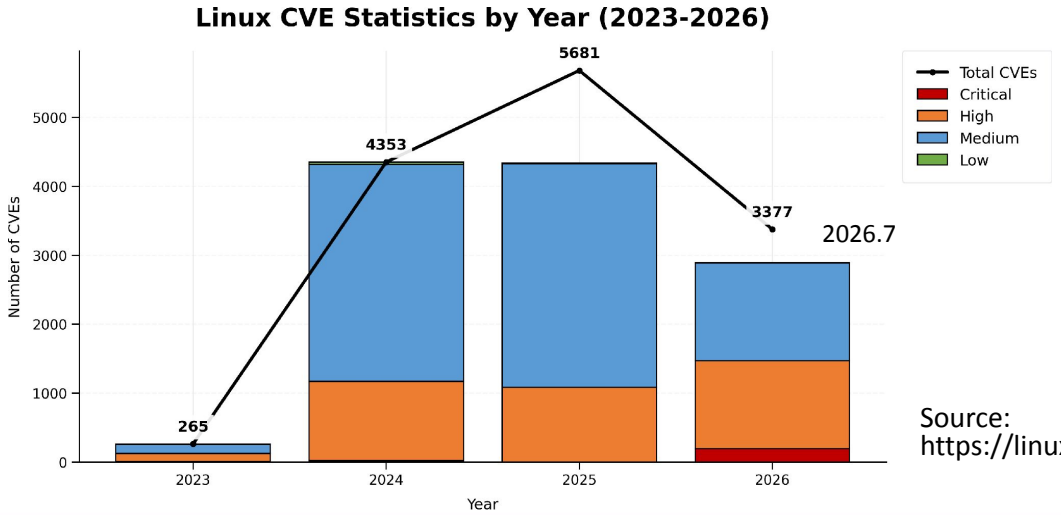
EU CRA: Timeline Pressure

Prepare the response process before the reporting clock starts.

CVE Volume Growth

AI is accelerating vulnerability discovery

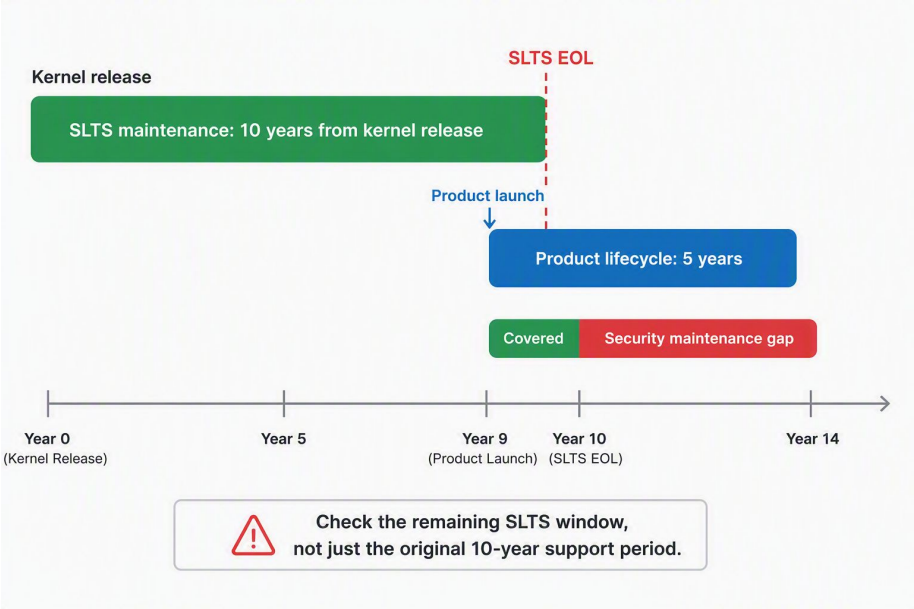
- AI-assisted fuzzing and code analysis find more vulnerabilities
- Security researchers now use AI agents to discover bugs at scale



Source: https://linuxcvetracker.com/

Embedded Linux Maintenance Reality

📺 Products often ship for 5-10+ years
🔧 Vendor customizations make kernel upgrades expensive
📉 Upstream support eventually ends
⚠️ Security maintenance continues anyway



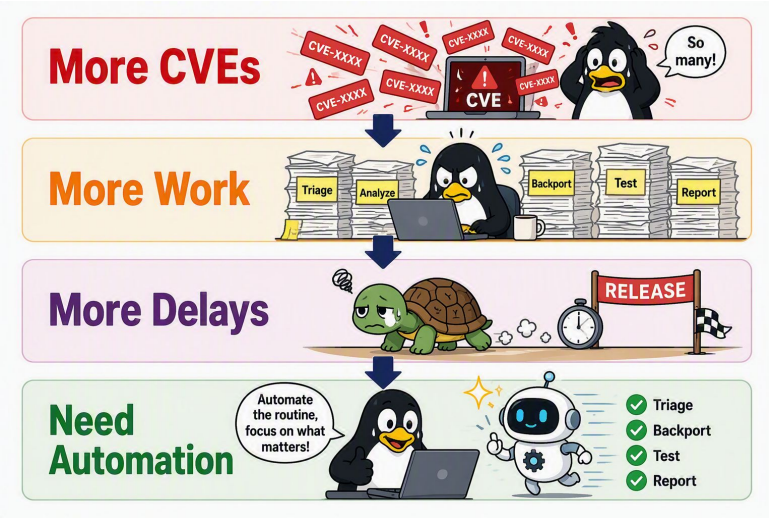
Why Automate Vulnerability Response?

✅ Triage every CVE

✅ Find the right fixes

✅ Validate every change

✅ Keep compliance evidence



Security response must be repeatable, scalable, and auditable.

Scope: Known Kernel Vulnerabilities

Out of Scope: Zero-Day Response

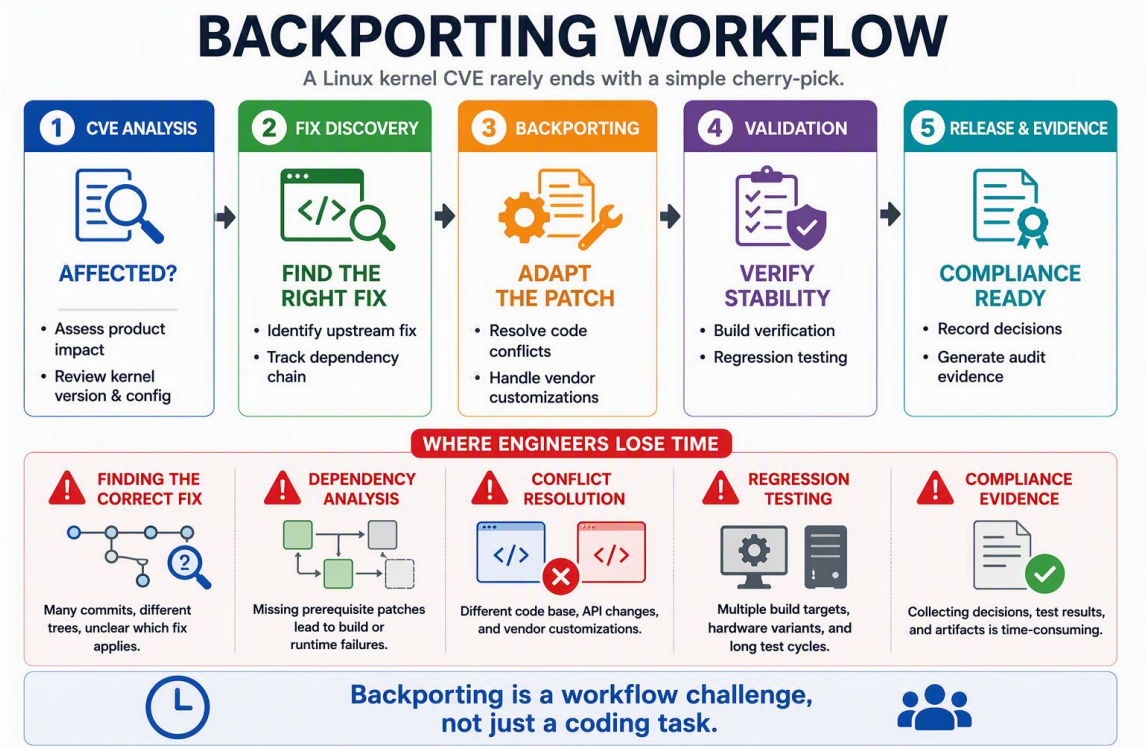
- Unknown or newly discovered vulnerabilities
- Requires incident-driven investigation and emergency mitigation

In Scope: 1-Day and N-Day Response

- Public vulnerability information and fixes are usually available
- Triage, backporting, testing, and evidence can be automated

The pipeline automates repeatable response to known vulnerabilities, not the discovery or containment of zero-day attacks.

Manual Response Bottlenecks



Linux Kernel CVE Response Workflow

From CVE to Release

🔍 Triage: Is this CVE our problem?

↓

🩹 Find the Fix: Patch + Dependencies

↓

⚙️ Backport: Apply or Adapt

↓

✅ Validate: Build + LTP + Selftests

↓

📋 Release & Evidence: Decision + Traceability

Where AI Can Help in the Workflow

AI supports each step, but engineers remain in control.

🔍 Triage: Summarize CVEs and analyze product impact

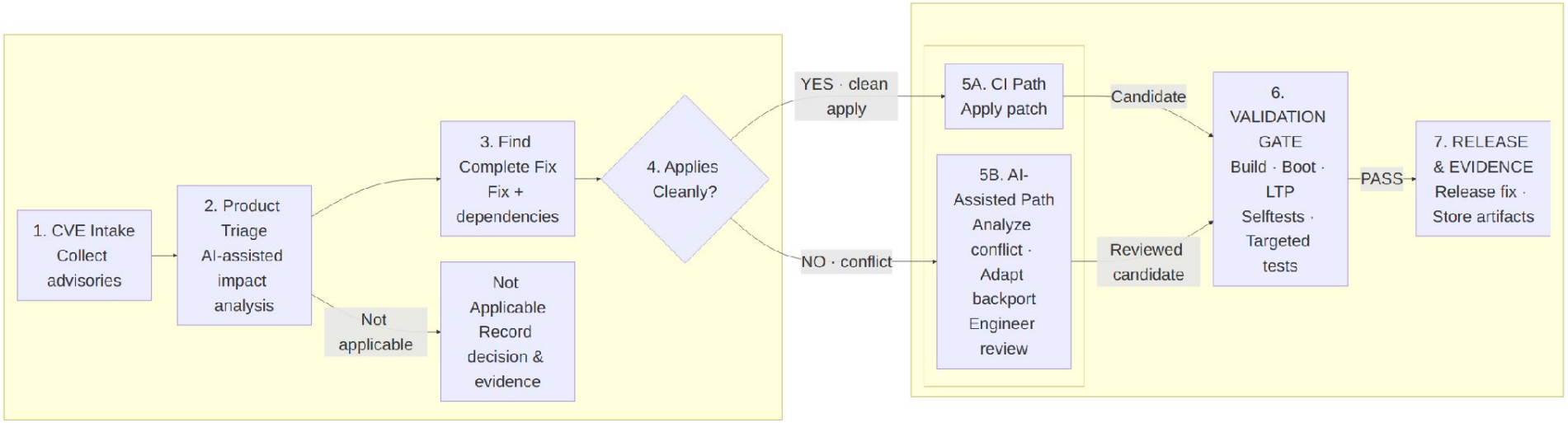
🩹 Find the Fix: Find upstream fixes and patch dependencies

⚙️ Backport: Explain conflicts and suggest adaptations

✅ Validate: Suggest targeted tests and analyze failures

📋 Release & Evidence: Summarize decisions, test results, and traceability

CI Pipeline with an AI-Assisted Path



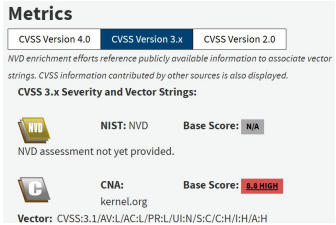
Stage 1 & 2: CVE Intake, Triage & Prioritization

Is this CVE really our problem?

- ✅ Gather CVEs from trusted sources
- ✅ Match against our kernel version & config
- ✅ Check whether the vulnerable path is reachable
- ✅ Prioritize by actual product impact

Result: Applicability decision + Priority + Evidence

- 2026/08/04 #3: CVE-2026-64562: KVM: nVMX: Hide shadow VMCS right after VMCLEAR (Greg Kr
- 2026/08/04 #2: CVE-2026-64563: rhashtable: clear stale iter->p on table restart (Greg Kroah-Hartman <
- 2026/08/04 #1: CVE-2026-64564: sctp: don't free the ASCONF's own transport in DEL-IP processing (G
- 2026/07/30 #1: CVE-2022-4994: KVM: x86: wean fast IN from emulator_pio_in (Greg Kroah-Hartman
- 2026/07/29 #5: CVE-2026-64560: posix-cpu-timers: Prevent UAF caused by non-leader exec() race (Gre
- 2026/07/29 #4: CVE-2026-64558: s390/pkey: Check length in pkey_pckmo handler implementation (Gre
- 2026/07/29 #3: CVE-2026-64559: s390/pkey: Check length in PKEY_VERIFYPROTK ioctl (Greg Kroa
- 2026/07/29 #2: CVE-2026-64556: perf/core: Detach event groups during remove_on_exec (Greg Kroah-
- 2026/07/29 #1: CVE-2026-64557: Bluetooth: L2CAP: Fix use-after-free in l2cap_sock_new_connection



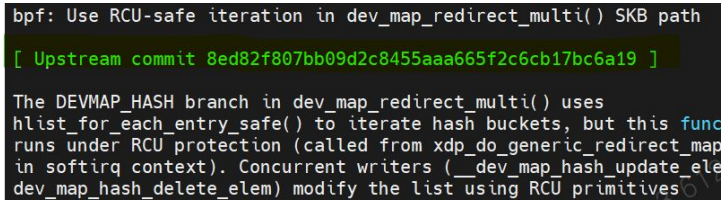
https://copy.fail	CVE	Exploit
https://git.kernel.org/stable/c/19d43105a97be0810edbda875f2cd03f30dc130c	kernel.org	Patch
https://git.kernel.org/stable/c/3115af9644c342b356f3f07a4dd1c8905cd9a6fc	kernel.org	Patch
https://git.kernel.org/stable/c/893d22e0135fa394db81df88697fba6032747667	kernel.org	Patch

Source: linux-cve-announce, vulns.git, stable repo, NVD

Stage 3: Finding the Complete Fix

One CVE may require more than one patch

✅ Find the original upstream fix
✅ Locate the best stable backport
✅ Identify prerequisites and follow-up fixes
✅ Track patch origin and relationships



Result: Complete patch set ready for backporting

Stage 4: Backport Complexity Assessment

Choose the fastest safe path

✅ Check whether the fix already exists
✅ Attempt automated patch application
✅ Analyze conflicts, APIs, and dependencies
✅ Assess backport complexity

Decision: CI or AI-Assisted Path

Simple → CI Path
Complex → AI-Assisted Path

Stage 5A: CI-Based Backporting

For clean and repeatable fixes

✅ Apply the patch automatically
✅ Detect patch conflicts
✅ Preserve patch metadata and traceability
✅ Send the candidate to validation

Output: Backport candidate ready for validation

Stage 5B: AI-Assisted Backporting

For conflicts and older kernels

- ✅ Explain the original fix intent
- ✅ Analyze conflicts and missing dependencies
- ✅ Suggest adaptations for the target kernel
- ✅ Suggest targeted validation tests
- ✅ Require engineer review

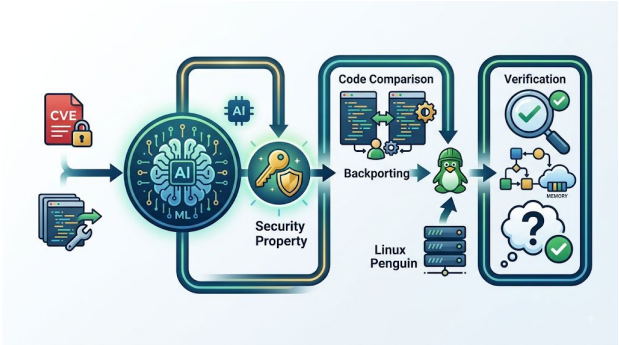
```
# 🤖 [Project/Task Name] Agent Instructions

## 1. Agent Role & Domain Context
## 2. Environment & Prerequisites
## 3. Workflow & Pipeline
   ### Step 1: Pre-check & Context Gathering
   ### Step 2: Code Modification / Backporting
   ### Step 3: Conflict Resolution (If applicable)
   ### Step 4: Verification & Static Analysis
## 4. Coding & System Conventions
## 5. Allowed Tools & Commands
## 6. Safety Guardrails & Forbidden Actions
## 7. Output & Commit Message Format
## 8. Escalation & Fallback Rules
```

Output: Engineer-reviewed candidate ready for validation

AI-Driven Threat Modeling for CVE Backporting

- AI identifies the security invariant.
- The invariant guides conflict resolution and backporting.



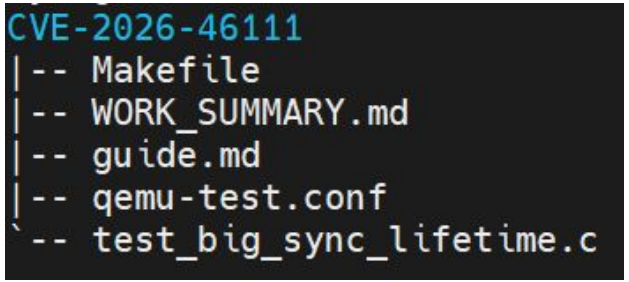
Key question:
“Does the backported patch preserve the original security property?”
Example:
“An unprivileged process must not access a kernel object after it is freed.”

Stage 6: Validation Gate

Every candidate must pass the same checks

✅ Build with product configurations
✅ Run boot and smoke tests
✅ Run LTP and kernel selftests
✅ Detect new regressions against the baseline
✅ Run vulnerability-specific tests

Decision: Accept, Rework, or Investigate



When Validation Gets Hard

Some fixes need more than a VM

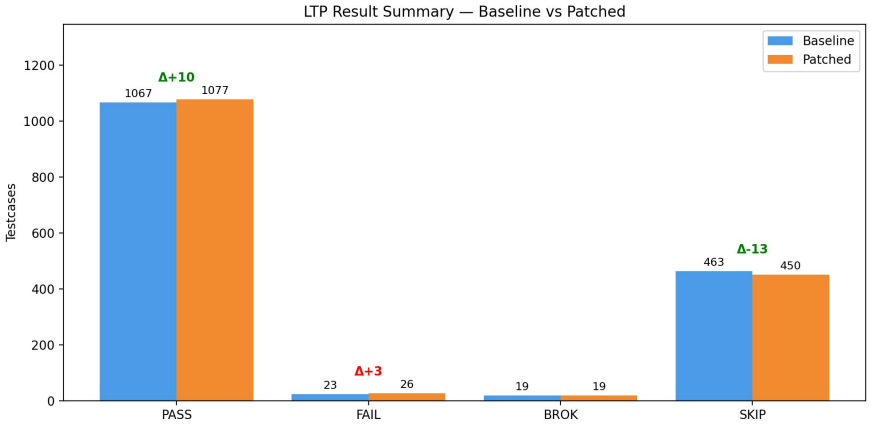
✅ VMs provide fast and scalable testing
✅ Some bugs depend on real hardware behavior
✅ Drivers, DMA, and peripherals may need device testing
✅ Hardware labs can also be automated

Use real hardware when the affected path depends on it.

Linux Kernel Validation Tools

- LTP: Broad system and kernel regression testing
- Kernel Selftests: Targeted testing for kernel features and interfaces
- Syzkaller: Coverage-guided fuzzing for finding unexpected bugs

Different tools catch different kinds of problems.



When Automation Meets Reality #1

Security Guardrails Can Block Valid Verification

⚠️ CVE test generation may look like exploit development
🛑 The AI agent may flag or stop the task
👨‍💻 An engineer reviews the intent and risk
✅ Approved testing continues in an isolated environment

> This content was flagged for possible cybersecurity risk. If this seems wrong, try rephrasing your request. To get authorized for security work, join the Trusted Access for Cyber program: https://chatgpt.com/cyber

When Automation Meets Reality #2

AI Agents Can Get Stuck

🔁 Repeats the same failed approach
🧩 Misses hidden kernel assumptions
📚 More context can add more noise
🛑 Stops after a defined retry limit
🧑‍💻 Hands the task off to an engineer

```
% Patch failed
```

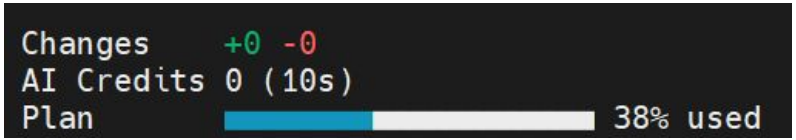
Good automation knows when to stop.

When Automation Meets Reality #3

AI Can Cost More Than Tokens

🔁 Retries: Conflicts and failures trigger repeated attempts
📚 Growing Context: More code, patches, and logs increase usage
🧪 Validation Loops: Every change needs another build and test
⏱️ Engineer Time: Incorrect results still need human review

Measure total engineering cost, not just token cost.



Humans on the Loop vs. in the Loop

Today · In the Loop

👀 Review AI-generated results

🧩 Resolve complex cases

✅ Approve critical decisions

🛑 Stop unsafe automation

Direction · On the Loop

📊 Monitor routine automation

⚠️ Handle exceptions

🛡️ Define limits and guardrails

🙋 Step in when needed

Key Takeaways

- Automate repeatable CVE response tasks
- Use AI as an engineering assistant
- Keep validation and critical decisions under human control

Q&A

Thank you!!