Trusted OSS

AI-Powered Open Source Risk Management

An ISO self-certification kit,
and a 5-level model for governing AI coding

Haksung Jang

Open Source Program Manager, SK Telecom
Ambassador, OpenChain Project

Open Source Summit Korea 2026
August 12, 2026 · Rose

THE PROBLEM

What changed

Three shifts, and the gap each one opened

Supply chain attacks keep happening

2024

XZ Utils

A backdoor planted in the upstream project, after two years of earning trust

2021

Log4Shell

One library. Hundreds of millions of systems. No malice at all

Different causes, **the same failure** — nobody knew which product had the library inside.

Working out the blast radius took days on its own

The bar went up. The means did not.

REPORTING SEPT 11, 2026

EU CRA

Full application follows on December 11, 2027

PUBLISHED JULY 29, 2026

CISA 2026

Replaces NTIA 2021. Every transitive dependency, not just the top level. Hashes and licenses required. Scope reaches AI software and SaaS

The problem is not that suppliers cannot produce an SBOM. **They produce one, and it still falls short.**

AI coding changed three conditions

INTAKE

The agent installs what it suggests. A human sees the name **after it is in the code**.

REVIEW DENSITY

Output grows several times over. **Reviewer headcount stays exactly where it was.**

TOOL CALLS

An MCP tool description enters the context. **Nobody reads it, and it is not in the repo.**

Existing controls assume a human wrote the code and a human reviewed it.

THE MAP

What we built

Two areas, and the tools that run what they produce

What Trusted OSS covers

An initiative of the OpenChain Korea Work Group, free to use and adapt.

PART 1

Build the program

ISO/IEC 5230 and 18974 self-certification

`9 agents → 24 artifacts → declaration`

PART 2

Govern AI coding

A 5-level maturity model

`ad-hoc → rules → gates → AI defense → monitoring`

both built in the DevSecOps guide

A different kind of risk. They meet on the roadmap, slide 18

PART 3

Run it

TRUSCA when the whole program has to keep running

BomLens when all you need is an accurate SBOM

PART 1

Build the program

Policies, processes, artifacts — from nothing to a declaration

What the two standards actually ask for

ISO/IEC 5230 ONLY
license compliance

- License obligations
- Notices
- Contribution policy

SHARED BASE
build it once, it counts for both

- Policy · Organization
- Process · Training
- SBOM

ISO/IEC 18974 ONLY
security assurance

- CVE tracking
- Response
- Records

The shared base is the larger half. Build one and most of the other is already done.

The OpenChain Korea Work Group publishes an enterprise guide and policy and process templates for both.

An agent writes each part of the program

Nine of them, 24 artifacts, run in order. Every one is a prompt you can read. We will open the policy agent live: steps, then output.

Organization	roles, RACI, appointment letter	steps · output
Policy	OSS policy, allowed license list	steps · output
Process	approval, pre-release check, vulnerability response	steps · output
SBOM	generation scripts, license report	steps · output
Vulnerability	CVE report, remediation plan	steps · output
Training	curriculum, completion tracking	steps · output
Conformance	gap analysis, declaration draft	steps · output

How you declare conformance

1 Download the checklist

`OpenChain-Project/Reference-Material` · 25 items each

2 Self-assess — yes or no on every item

your gap analysis already answers most of them

3 Register

`openchainproject.org/get-started`

No external audit. No cost. The declaration is valid for 18 months.

PART 2

Govern AI coding

Five kinds of control, and which ones you already have

Five layers, one slide each

What to add, a sample to copy, and where it runs today.

LEVEL			SEE IT
L1	**Prompt-dependent**	policy in one person's memory, or nowhere	—
L2	**Rules in the repo**	CLAUDE.md, AGENTS.md	rules-template
L3	**Blocked in CI/CD** DEVSECOPS	gitleaks, semgrep, grype	cicd-quick
L4	**AI defense layer**	4a review · 4b fuzzing · 4c agent and MCP tools	ai-security-review
L5	**Continuous monitoring** DEVSECOPS	dependabot, DAST	dependabot.yml

Levels 3, 4 and 5 all run in TRUSCA's own CI. All five sit in one repository you can fork: `ai-coding-best-practice`.

LEVEL 1

Policy that lives in one person's head

"Use only MIT-licensed code" typed into the prompt, every time. **There is no file to show you at this level.**

Nothing written down is nothing to review, inherit, or audit.

LEVEL 2

Rules the agent reads before it writes

What you add	`CLAUDE.md`, `AGENTS.md`, `.cursor/rules` in the repository
Sample	A template to copy today, plus per-tool setup
In practice	trustedoss.github.io runs on one, and so does ai-coding-best-practice

The agent treats a rule as guidance, never as a gate. **That gap is the whole reason level 3 exists.**

LEVEL 3

The gate that does not negotiate

DEVSECOPS	WHAT IT STOPS	TOOLS	RUNS IN
Secrets	Keys and tokens left in the source	`gitleaks`	`secret-scan.yml`
SAST	Flaws in the code you wrote	`semgrep`, `CodeQL`	`sast.yml`
SCA	CVEs and licenses in what you pulled in	`syft` · `grype` · `cdxgen` · `Trivy`	`sca-self.yml`

Secrets first. A leaked key cannot be un-leaked, and rotating it is not the same as never leaking it.

Turn them on one at a time. cicd-quick walks the order. Container and IaC scanning join when you run either, and the reference repository has both.

LEVEL 4 · WHY IT EXISTS

Three places level 3 stops

WHAT IT FLAGS

Mixed with false positives. **A human re-judges each one.**

WHAT IT MISSES

Business logic, permission checks, state transitions. **No rule describes them.**

WHAT IT NEVER SEES

Tool descriptions and tool output the agent reads. **They never reach a diff.**

WHAT LEVEL 4 DOES ABOUT EACH

4a · findings-driven review

Raises precision on what was already flagged.

4b · AI fuzzing

Searches the area that was never flagged at all.

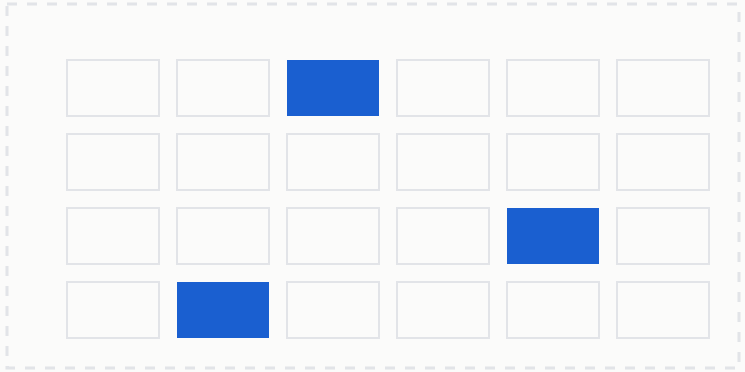
4c · agent and MCP tools

Guards what the agent calls, not what it writes.

LEVEL 4A

What actually reaches the model

YOUR REPOSITORY



semgrep.sarif, grype.json

8 findings, 5 lines each

model

true positive or false risk, exploit path

→ PR comment, not a gate

This runs on every pull request in TRUSCA — `ai-review.yml`. Advisory only, never a merge gate, and it skips entirely when no API key is set.

LEVEL 4B

Finding what no rule describes

Business logic and edge-case input handling, the area no rule describes

Running	ai-fuzzing.yml in the best-practice repository, on push and weekly

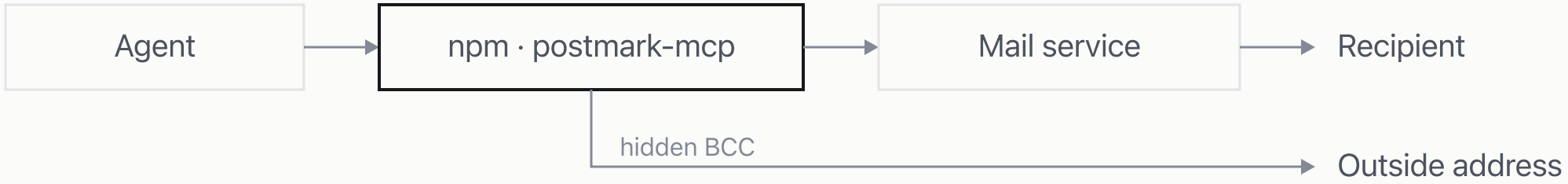
This one runs the app. Everything before it only reads.

Which is why it lands on a schedule rather than on every commit.

LEVEL 4C

An MCP server is supply chain input too

Every server the agent calls is a dependency you did not review.



`1.0.15`	clean
`1.0.16+`	A **hidden BCC** copied every outgoing mail to an outside address

A review at adoption would have passed it. **It was clean at the time.**

The starting version is the researcher's estimate. Source: Snyk.

LEVEL 4C

Six controls, and how to review

1 Server allowlist

2 Pin the version

3 Least privilege

4 Human approval, with an audit log

5 Review the tool description

6 **Judge the data egress path**

1 to 5 are Microsoft Incident Response and the MCP spec turned into working rules. 6 is from the case on the previous slide.

Scan first	Snyk agent-scan, Cisco mcp-scanner — check what they transmit first
Then govern	ToolHive, agentgateway — a person still judges scope and egress

LEVEL 5

What keeps running after the merge

LEVELS 1 TO 4 — LEVEL 5

Write · rules · gate — merge — Deploy, then every day after it

Updates	Patches raised for you, five ecosystems	Dependabot	trusca `dependabot.yml`
Rescan	SBOM rebuilt and rescanned daily	`cdxgen` · `Trivy`	trusca `sca-self.yml`
DAST	The deployed app probed, not the source	OWASP ZAP	best-practice `dast.yml`
Dogfooding	The scanner scans itself, advisory	TRUSCA itself	trusca `dogfood-scan.yml`

A CVE published tomorrow lands in code you merged last year.

PART 3

Run it

Documents do not execute themselves. These tools do

Writing the artifacts is not running the program

- A policy document **does not block** a forbidden license
- A process document **does not record** an approval
- One SBOM **does not track** the next CVE

18974 §4.3.2 asks for monitoring after release, a judgement on every finding, and a record of that judgement.

Including the findings you decide need no action (§4.3.2.2)

TRUSCA — Apache-2.0, self-hosted SCA

The answer to the slide before. Stand it up as it is, no licence to buy.

detect	cdxgen, more than 30 ecosystems
match	Trivy database — NVD, OSV, GHSA, EPSS, KEV
triage	VEX import and export, 7 triage states
enforce	3-tier license policy, CI gate, generated NOTICE
operate	RBAC, audit log, Compose and Helm

Runs inside your own network. Neither the code nor the SBOM leaves it.

Why this one

SELF-HOSTED

Runs inside your own network

Neither the code nor the SBOM leaves it. Apache-2.0

SIGNALS

EPSS, KEV and VEX, not just CVSS

Exploit probability and known exploitation, in the finding

EVIDENCE

Every judgement is recorded

Seven triage states, RBAC, an audit log the auditor can read

REGULATION

It drafts the documentation

G7 AI elements mapped to the EU AI Act and Korea's AI Act

No licence, no procurement, no data leaving the building.

Website `trustedoss.github.io/trusca` **Live demo** `trusca-demo.duckdns.org` **Source**

What actually arrives as an SBOM

Missing dependencies are only part of it.

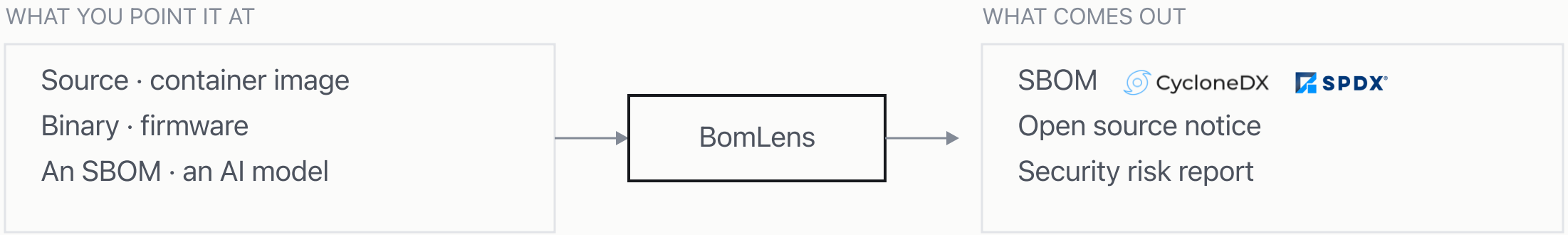
- A file with **no component information in it at all**
- A file listing **tens of thousands of file names**
- Dependencies resolved only at build time, so the transitive ones drop out

BomLens is what SK Telecom hands suppliers along with its guidance. Open source, so anyone else can use it too.

10 languages · Apache-2.0 · runs locally

One run, three documents

This is what you hand the supplier. No account, no upload — it runs on their machine.

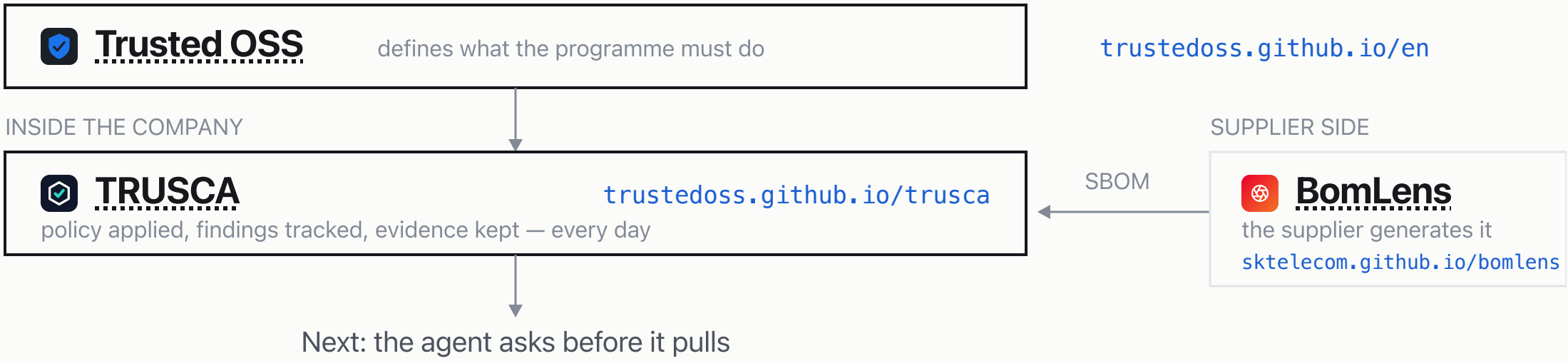


Suppliers who cannot hand over source can still hand over an SBOM.

Website `sktelecom.github.io/bomlens` **Live demo** `/demo` **Source**

Where the three are headed

One defines the programme, one runs it, one feeds it from outside.



All three are free and open. Two have a live demo you can open right now.

TRUSCA demo `trusca-demo.duckdns.org` **BomLens demo** `/bomlens/demo`

Questions

trustedoss.github.io/en

github.com/sktelecom/bomlens

github.com/trustedoss/trusca

Haksung Jang

Open Source Program Manager, SK Telecom

Ambassador, OpenChain Project

Lead, OpenChain Korea Work Group

Thank you