NuGuard's Claude Plugin for AI Security

The Claude Code plugin that audits your AI app - AI-SBOMs, static risk analysis, and red-teaming, all from your terminal.

View on 🐙 GitHub 📖 Read the Docs

AI Agents Ship Fast. Security Audits Don't.

As a developer you see immense productivity improvements using coding agents like Claude Code but security audits are still a hindrance. What if there's an AI security tool that integrates cleanly with Claude Code, thus making the security workflow almost conversational.

NuGuard is an open-source **AI Trust** platform purpose-built for **agentic AI** applications. It maps your entire AI stack into a structured attack-surface graph and surfaces real risks before adversaries do.

- **AI-SBOM Generation:** Inventory every component in your AI app
- **Static Risk Analysis:** Nuguard rules, MITRE ATLAS, CVE/OSV scanning
- **AI Red-Teaming:** Prompt injection, data exfiltration, privilege escalation.
- **Behavior Validation:** happy-path testing of your tools, agents, end-points

Three Commands. Zero Context Switching.

Install NuGuard directly inside the Claude Code terminal. The plugin registers 7 skills as slash commands—no separate toolchain required.

Terminal Output Example:

```
None
$/plugin marketplace add NuGuardAI/nuguard
Successfully added marketplace: nuguard
$/plugin install nuguard
✓ Installed nuguard. Run /reload-plugins to apply.
```

```
$ /reload-plugins
Reloaded: 1 plugin 7 skills 7 agents 0 hooks 0 plugin MCP servers. 0 plugin LSP
servers
```

7 Skills, Instantly Available

After installation, NuGuard registers these slash commands directly in your Claude Code session:

1. `/nuguard-sbom`: Generate a full AI Bill of Materials from your codebase
2. `/nuguard-analyze`: Static risk analysis against an existing SBOM
3. `/nuguard-scan`: Full pipeline: SBOM → analysis → report
4. `/nuguard-redteam`: Adversarial red-team testing against agents
5. `/nuguard-behavior`: Behavioral testing against a live running app
6. `/nuguard-config`: Configure credentials and target endpoints
7. `/nuguard-init`: Initialize project config files

DEMO: AI-SBOM

37 Nodes. 40 Edges. Under 5 Seconds.

Here's the output from a real airline customer-service agent application:

Terminal Output:

```
None
$ create an SBOM for my repo
Scanning /workspaces/openai-cs-agents-demo ...
SBOM written to app.sbom.json (37 nodes, 40 edges)
```

Node Summary

Type	Count
AGENT	5

Type	Count
API_ENDPOINT	5
AUTH	1
DATASTORE	1
DEPLOYMENT	2
FRAMEWORK	3
GUARDRAIL	2
IAM	1
MODEL	1
PRIVILEGE	1
PROMPT	8
TOOL	7

- **Agents Detected:** Triage, Cancellation, FAQ, Flight Status, Seat Booking (5 Agents)
- **Tools Mapped:** `cancel_flight`, `update_seat`, `lookup_reservation`, and 4 more (7 Tools)
- **Guardrails Found:** Jailbreak Guardrail + Relevance Guardrail (2 Guardrails)
- **Total SBOM Nodes:** Full attack surface mapped across 40 edges (37 Nodes)

RISK ANALYSIS

What the SBOM Reveals

The AI-SBOM is more than an inventory—it's a dependency graph of your AI application's attack surface. NuGuard checks every tool for injection risk, privilege exposure, PII leakage, and more.

```
None
Key Observations
- Multi-agent orchestration via openai_agents SDK — Triage Agent
fans out to 4 specialist agents.
- Two guardrails are in place (Jailbreak + Relevance), which is
good — /nuguard-analyze will verify their coverage.
- db_write privilege node is present, sourced from the update_seat
/ cancel_flight flow — worth checking that only authorised agents
can reach it.
- Azure credentials are injected via GitHub Actions secrets —
correctly not hardcoded.
- All 7 tools came back clean on SQL injection, SSRF, and privilege
flags.
```

WHY IT MATTERS

Why Traditional SBOMs Fail for Agentic AI Security and AI Risk Management?

A `cancel_flight` tool with no guardrail coverage can be prompt-injected into cancelling flights it shouldn't. A system prompt with template variables like `{confirmation_number}` can leak sensitive reservation data if an attacker controls an adjacent agent's output.

NuGuard maps these relationships precisely so you understand what can reach what, before an adversary does.

- **Prompt Injection:** Adversarial input redirects agent behavior through crafted tool outputs
- **Data Exfiltration:** Template variables in system prompts leak sensitive data across agent boundaries
- **Privilege Escalation:** Misconfigured orchestration graphs grant unintended tool access to unprivileged agents

FULL PIPELINE

How Can I Run a Full AI Security Report in One Command?

🗺️ **SBOM Generation:** Maps all agents, tools, models, prompts, datastores, and endpoints into a structured graph	🔍 **Static Analysis:** Applies NGA rules, MITRE ATLAS techniques, and CVE/OSV vulnerability scans
🧪 **Behavioral Testing:** Intent-aware test suite executed against your live application	📋 **Findings Report:** Consolidated security findings with severity ratings and remediation guidance

Run the complete security pipeline end-to-end with a single command. Or use `/nuguard-analyze` to run just the analysis phase against an existing SBOM.

Terminal Output Example:

```
None
$ /nuguard-scan
Running full pipeline: SBOM → static analysis → behavioral tests → report ...
✓ SBOM generated (37 nodes, 40 edges)
✓ Static analysis (NGA rules: 0 critical, 1 warning | MITRE ATLAS: 2 techniques flagged)
✓ Behavioral tests (12 tests passed, 0 failed)
✓ Report written to nuguard-report.html
```

Get Started with NuGuard Today

NuGuard is built for the agentic AI era where the attack surface isn't just your code, it's everything your agents can do. Open-source, terminal-native, and ready in three commands.

- **Install the Plugin:** `/plugin marketplace add NuGuardAI/nuguard` inside Claude Code
- **GitHub Repository:** github.com/NuGuardAI/nuguard (open-source, contributions welcome)

- **Documentation:** Full reference, rule definitions, and integration guides at nuguard.ai

View on 🐙 GitHub 📖 Read the Docs