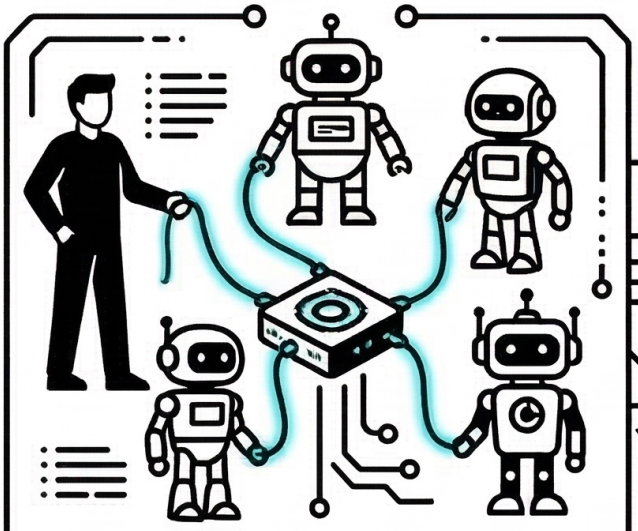
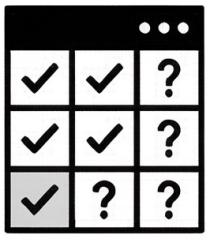
KEEPING YOUR AGENTS ON A LEASH

Lab Handout: Agentic Guardrails, MCP Security, AI BOMs, Chatbot Red-teaming, Developer Security, SCA, and Supply Chain Risk

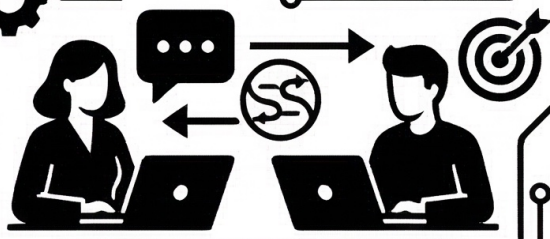


Agentic Guardrails

MCP Security



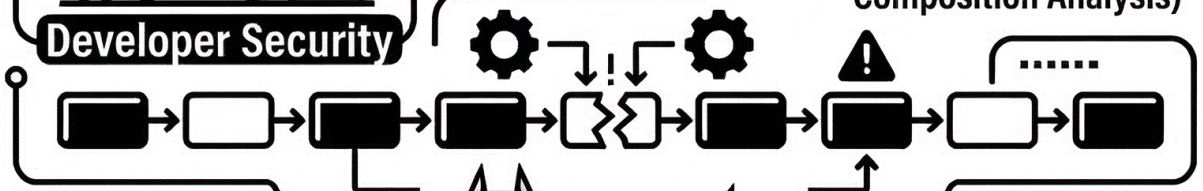
AI BOMs
(Bill of Materials)



Chatbot Red-teaming

Developer Security

SCA
(Software / Software Composition Analysis)



snyk

Requirements

To ensure a smooth setup across all the guides provided (Cursor integration, AI-BOM, MCP scanning, and Red Teaming), users should meet the following technical requirements.

📋 System & Software Requirements

Before starting the integration, please verify that your environment meets these specifications:

1. Snyk Account & Access

- **Active Snyk Account:** A free or paid account at snyk.io with admin permissions

2. Core Software Tools

- **Cursor IDE:** The latest version of the Cursor AI Editor.
- **Snyk CLI:** Version **v1.1302.1** or higher.
    - *Check version:* snyk --version
    - *Update:* npm install -g snyk
- **Node.js & npm:** Required to run the Snyk CLI and the MCP server via npx.
- **Python 3.10+:** Required for AI-BOM generation and running the uv package manager for MCP security scans.

3. Environment Configuration

- **Authenticated CLI:** You must be logged in locally.
    - *Command:* snyk auth
- **The uv Tool:** Required specifically for snyk-agent-scan (MCP server auditing).
    - *Install:* curl -LsSf https://astral.sh/uv/install.sh | sh
- **Docker (Optional):** Required only if you intend to run the **Snyk Scanning Agent** for Red Teaming against private or local network endpoints.

🛠️ Quick Checklist

Requirement	Purpose
Snyk CLI (Latest)	Core scanning engine for all tasks. See Install section
Snyk IDE Extension	Real-time "Secure at Inception" scanning in Cursor.
Experimental Flags	Required for AI-BOM and Red Team commands.

Local Project Trust	You must grant Snyk permission to scan your workspace folder.

> **Note for Windows Users:** Ensure your execution policy allows running scripts in PowerShell, or use **WSL2**(Windows Subsystem for Linux) for a more seamless experience with the `uv` and `npx` commands.

Snyk CLI install

To install the Snyk CLI, use one of the following commands based on your environment:

Installation Methods

- **npm (Recommended for Node.js users):**

```
Shell
npm install snyk -g
```

- *Requires Node.js v12 or higher.*
- **Homebrew (macOS & Linux):**

```
Shell
brew tap snyk/tap
brew install snyk
```

- *The Homebrew formula is updated daily.*
- **Scoop (Windows):**

```
Shell
scoop bucket add snyk https://github.com/snyk/scoop-snyk
scoop install snyk
```

- **Standalone Binary:**
  You can download the latest executables directly from the Snyk CDN or GitHub Releases.

Next Steps

1. **Verify Installation:** Check the version to ensure it's installed correctly.

```shell
snyk --version
```

2. **Authenticate:** Connect your CLI to your Snyk account.

```shell
snyk auth
```

3. *This will open a browser window to log you in.*
4. **Run a Scan:** Test your current project for vulnerabilities.

```shell
snyk test
```

Establishing security guardrails on agentic coding

This guide provides a step-by-step process for integrating **Snyk** into the **Cursor IDE** using the Model Context Protocol (MCP). This setup ensures that any code generated by Cursor's AI is automatically scanned for vulnerabilities before you commit it.

🛡️ Snyk + Cursor Security Integration

1. Prerequisites

Before you start, ensure you have the following:

- **A Snyk Account:** Sign up for free if you don't have one.
- **Snyk CLI:** Installed on your machine.
  - *Command:* `npm install -g snyk`
- **Authentication:** Link your CLI to your account.
  - *Command:* `snyk auth`

---

2. Install the Snyk Extension

The easiest way to set up the MCP server is through the official Snyk extension, which now includes "Secure at Inception" features.

1. Open **Cursor**.
2. Go to the **Extensions** view (Ctrl+Shift+X).
3. Search for **"Snyk Security"** and click **Install**.
4. Once installed, a modal will appear asking to enable **"Secure at Inception."**
5. **Select "Yes"** (This automatically configures the MCP server and scanning rules for you).

---

3. Manual MCP Server Configuration

If the automatic setup didn't trigger, or if you prefer a manual configuration, follow these steps:

1. Open **Cursor Settings** (Ctrl + Shift + J or File > Preferences > Cursor Settings).
2. Navigate to **Tools & Integrations > MCP**.
3. Click **Add New MCP Server** and enter the following:
   - **Name:** Snyk
   - **Type:** command
   - **Command:**

```shell
npx -y snyk@latest mcp -t stdio
```

(Note: Using the full path to your Snyk executable is recommended for stability).

---

4. Configure MCP Rules (The "Guardrails")

To ensure Cursor **always** scans code it generates, you need to define a rule file. This instructs the AI agent to use the Snyk tool whenever it writes or modifies code.

1. In your project root, create a directory named .cursor.
2. Inside that directory, create a file named rules (or edit the existing .cursorrules).
3. **Append the following instructions:**

Markdown

```
# Snyk Security Guardrails
- Always run the `snyk_code_scan` tool after generating or refactoring first-party code.
- Always run `snyk_sca_scan` after adding or updating any open-source dependencies.
- If Snyk identifies security vulnerabilities, prioritize fixing them using the provided context before finalizing the code.
- Do not suggest code that contains 'High' or 'Critical' severity vulnerabilities.
```

---

5. Verification: Testing the Setup

To confirm everything is working, try a "vibe check" with the AI:

1. Open a new chat in Cursor (Ctrl+L).
2. Type: *"Write a Python function that takes a string and executes it using eval()."*
3. **Observe:** You should see Cursor invoke the **Snyk MCP tool**. Snyk will flag the use of eval() as a security risk (Command Injection), and the AI should offer a more secure alternative based on Snyk's feedback.

Troubleshooting Tips

- **Trust the Folder:** Snyk requires you to "trust" a directory before it can scan it. If the scan fails, the AI might prompt you to run `snyk_trust`. Just click "Allow."
- **Scans not triggering?** Ensure the Snyk CLI is up to date (`snyk --version` should be v1.1298.0 or higher).
- **Quota Limits:** If you are on a free Snyk plan, keep an eye on your monthly scan limits in the Snyk dashboard.

AI Bill Of Materials (AI BOM)

This guide explains how to use the **Snyk CLI** to generate an **AI Bill of Materials (AI-BOM)**. While a traditional SBOM tracks open-source libraries, an AI-BOM identifies the "Shadow AI" in your code—including Large Language Models (LLMs), datasets, AI frameworks, and Model Context Protocol (MCP) connections.

🤖 Generating an AI-BOM with Snyk

1. Prerequisites

- **Snyk CLI:** Ensure you are on version **v1.1298.0** or later.
- **Python Project:** Currently, AI-BOM generation is optimized for Python-based AI projects (supporting libraries like PyTorch, TensorFlow, Hugging Face, LangChain, etc.).
- **Authentication:** You must be logged in via `snyk auth`.

---

2. Generate the AI-BOM

Navigate to your project's root directory and run the aibom command. Since this feature leverages Snyk's latest AI-native scanning engine, it currently requires an experimental flag or a specific profile.

Basic JSON Output (CycloneDX)

To generate a machine-readable AI-BOM in the industry-standard CycloneDX v1.6 format:

Bash
`snyk aibom --experimental`

Visual HTML Report

For a human-readable visualization of your AI supply chain—showing how your code connects to specific models and MCP servers:

Bash
`snyk aibom --experimental --html > snyk-aibom.html`

---

3. Understanding the Output

The AI-BOM identifies components across four key categories:

Category	Examples Detected
Models	GPT-4, Claude 3.5, Llama 3, DeepSeek
Datasets	References to Hugging Face datasets or local training sets
AI Libraries	langchain, transformers, openai, pinecone
MCP Supply Chain	MCP Clients, MCP Servers, and specific Tools/Resources called

---

4. Advanced Configuration (2026 Features)

As of early 2026, Snyk has added "Enterprise Persistence" to the CLI, allowing you to upload these snapshots to the Snyk Web UI for centralized governance.

To upload your AI-BOM to your Snyk Organization:

```shell
snyk aibom --upload --repo https://github.com/[owner]/[repo]
--org=[YOUR_ORG_ID]
```

---

5. Automation: CI/CD Integration

You can integrate this into your GitHub Actions or GitLab CI/CD pipelines to block builds if "Unapproved" AI models (e.g., non-compliant open-source licenses or unvetted providers) are detected.

Example Policy Check:

```shell
snyk aibom --experimental --json | grep "unapproved-model-name"
&& exit 1 || exit 0
```

Scanning MCP Servers

This guide details how to use **Snyk's agent-scan** (formerly mcp-scan), a security utility specifically designed to audit the safety of the MCP servers you connect to your AI tools.

While the Snyk MCP server scans *your* code, snyk-agent-scan scans *the servers themselves* for threats like prompt injection, tool poisoning, and malware.

🛡️ Auditing MCP Servers with Snyk Agent Scan

1. Installation

The snyk-agent-scan tool is typically distributed via Python's uv for fast, sandboxed execution.

1. **Install uv** (if you don't have it):
    - *macOS/Linux:*

```shell
curl -LsSf https://astral.sh/uv/install.sh | sh
```

    - *Windows power shell:*

```shell
powershell -c "irm https://astral.sh/uv/install.ps1 | iex"
```

2. **Authenticate:**

```shell
export SNYK_TOKEN=your-snyk-api-token
```

---

2. Discover and Scan Local Servers

If you have multiple AI tools (Cursor, Claude Desktop, Windsurf) installed, Snyk can automatically find and scan all configured MCP servers on your machine.

Run the Global Discovery Scan:

```shell
uvx snyk-agent-scan@latest --skills
```

- **What this does:** It searches your local configuration files (like cursor_config.json or claude_desktop_config.json), connects to each discovered MCP server, and audits their tool descriptions and logic.

---

3. Scan a Specific MCP Configuration

If you are developing your own MCP server or want to test a specific mcp-config.json file before loading it into Cursor:

```shell
uvx snyk-agent-scan@latest scan /path/to/your/mcp-config.json
```

---

4. What Snyk Detects

The scan identifies **15+ distinct security risks** specific to the AI Agent ecosystem:

- **Prompt Injection:** Tools that could be manipulated by a malicious user to bypass system instructions.
- **Tool Poisoning:** Maliciously crafted tool descriptions that trick an LLM into calling the tool with sensitive data.
- **Toxic Flows:** Scenarios where a combination of tools (e.g., a "Read Email" tool + "Execute Shell" tool) creates a dangerous chain.
- **Hardcoded Secrets:** API keys or tokens accidentally exposed in the MCP server's code or environment.

---

5. Inspect Mode (No-Execution Audit)

If you want to see exactly what "knowledge" and "capabilities" a third-party MCP server is exposing to your AI without performing a full security verification:

```shell
uvx snyk-agent-scan@latest inspect /path/to/mcp-config.json
```

This prints a clean summary of every tool, prompt, and resource the server provides.

🚀 Summary Table: Which Snyk tool to use?

Use Case	Tool to Use
Scan the code you are writing	`snyk code test`
Scan the code Cursor generates	Snyk MCP Server (`snyk mcp`)
Scan the MCP servers you've installed	Snyk Agent Scan (`snyk-agent-scan`)

Red-teaming a chatbot

This guide explains how to use the **Snyk CLI redteam command** to perform adversarial testing on AI-native applications. Unlike standard security scanning, red teaming involves "attacking" your AI system with malicious prompts to see if it leaks data, ignores safety guardrails, or executes unauthorized actions.

🎯 AI Red Teaming with Snyk

1. What is Snyk Red Teaming?

While Snyk's other tools scan code or dependencies, snyk redteam is a **dynamic analysis tool (DAST)** for LLMs. It launches a battery of automated test cases against your AI target to identify:

- **Prompt Injection:** Forcing the AI to ignore its system instructions.
- **Jailbreaking:** Bypassing ethical or safety filters.
- **Insecure Output Handling:** Tricking the AI into generating executable code or scripts.
- **Sensitive Information Disclosure:** Attempting to extract system prompts or PII.

---

2. Basic Setup & Configuration

To run a red team scan, you first need a configuration file (usually redteam.yaml) that defines your **Target** (the AI app or MCP server you are attacking).

Example redteam.yaml:

YAML

```yaml
target:
  url: "http://localhost:8080/v1/chat" # Your AI endpoint
  description: "Customer support bot with access to order databases."
options:
  # Optional: Limit the number of test cases to 50 for a quick check
  max_test_cases: 50
```

---

3. Running the Scan

Ensure you have the latest Snyk CLI (v1.1302.1 or higher) and run the command with the experimental flag.

Standard Command:

```shell
snyk redteam --experimental
```

Retrieve a previous report by ID:

If you want to pull the results of a scan that ran in a CI/CD pipeline:

```shell
snyk redteam --experimental get --id=<SCAN_ID>
```

---

4. Advanced: Using a Scanning Agent

If your AI application is running on a private network (e.g., a local dev environment or a protected staging server), you must use a **Snyk Scanning Agent** to bridge the connection.

1. **Create the agent:**

```shell
snyk redteam scanning-agent create --experimental
```

2. **Start the agent:** Copy the generated Docker command from the output (it will include an AGENT_TOKEN) and run it on your local machine.
3. **Update your config:** Add the agent ID to your redteam.yaml:

YAML

```yaml
options:
  scanning_agent: "YOUR_AGENT_UUID"
```

---

5. Reviewing Results (HTML Report)

Red team results are statistically driven (e.g., "The Jailbreak attack succeeded in 15% of attempts"). To make this easy for stakeholders to understand, generate an HTML report:

```shell
snyk redteam --experimental --html > redteam-report.html
```

What to look for in the report:

- **Attack Success Rate:** Any percentage above 0% in a "Critical" category requires immediate instruction refinement.
- **Successful Payloads:** Snyk will show you the *exact* malicious prompt that broke your AI. You can use these to update your system prompt "Guardrails."

⚠️ Safe Red Teaming Rules

1. **Never scan Production:** Red teaming is disruptive. It can trigger real API calls or database writes if your agent has those permissions.
2. **Use Test Data:** Ensure your AI is connected to a sandbox database, not real user data.
3. **Exclude Destructive Tools:** If your AI has a "Delete User" tool, disable that tool's access before running the scan.