GUÍA PASO A PASO

Escanear una skill con SkillSpector antes de instalarla

Instalar SkillSpector 2.13.4 en un entorno aislado, escanear una skill sin mandar nada fuera, leer la calificación de 0 a 100 y los exit codes, usarlo como MCP en Claude Code y ponerlo como paso de CI. Con lo que sacaron cinco skills de prueba y lo que un escáner no ve.

PARA	Quien instala skills de terceros en Claude Code, Codex CLI o Gemini CLI
PROBADO CON	SkillSpector 2.13.4 con --no-llm · Python 3.12.3 · Windows 11 · actionlint 1.7.12
ACTUALIZADA	11 de noviembre de 2026
DOCUMENTACIÓN	github.com/NVIDIA/SkillSpector
ARTÍCULO	ecomlabs.dev/g/escanea
AUTOR	Luis Morales · @luismoralesdev

Índice

Antes de empezar	Qué vas a hacer
Paso 01	Instalar en un entorno aislado
Paso 02	Escanear sin mandar nada fuera
Paso 03	Leer la calificación
Paso 04	Revisar a mano lo que importa
Paso 05	Usarlo como MCP en Claude Code
Paso 06	Un paso de CI en GitHub Actions
Al terminar	Checklist

ANTES DE EMPEZAR

Qué vas a hacer

SkillSpector es un escáner de código abierto de NVIDIA (licencia Apache 2.0) que revisa una skill de agente sin ejecutarla y le da una calificación de 0 a 100. Lo probamos el 11 de noviembre de 2026 con la versión 2.13.4, publicada el 4 de noviembre, en Windows 11 con Python 3.12.3, y solo con análisis estático (`--no-llm`). Escaneamos cinco skills que escribimos nosotros; nunca una configuración real.

Necesitas Python 3.12 o superior. No está en PyPI: se instala desde GitHub.

PASO 01

Instalar en un entorno aislado

Usa el wheel de la release, que fija la versión:

```
python -m venv .venv-skillspector
source .venv-skillspector/bin/activate
REL=https://github.com/NVIDIA/SkillSpector/releases/download/v2.13.4
pip install "skillspector[mcp] @ $REL/skillspector-2.13.4-py3-none-any.whl"
skillspector --version
```

- En Windows, la activación es `.venv-skillspector\Scripts\activate`, y en PowerShell la variable se escribe `$REL = "https://github.com/NVIDIA/SkillSpector/releases/download/v2.13.4"`; la línea de `pip` queda igual.
- El extra `[mcp]` solo hace falta para el Paso 05. Sin él, `skillspector mcp` no arranca.
- `skillspector --version` debe imprimir `SkillSpector v2.13.4`.
- El README también propone `uv tool install git+https://github.com/NVIDIA/skillspector.git`. En Windows, instalar desde Git nos falló con «Filename too long»; el wheel no tiene ese problema.

PASO 02

Escanear sin mandar nada fuera

```
skillspector scan ./mi-skill/ --no-llm
skillspector scan ./mi-skill/ --no-llm --format json > reporte.json
skillspector scan ./skills/ --recursive --no-llm
skillspector scan ./mi-skill/ --no-llm --format sarif > reporte.sarif
```

- El destino puede ser una carpeta, un `SKILL.md` suelto, un `.zip` o una URL de Git.
- `--no-llm` apaga la etapa LLM, que está activa por defecto y manda el contenido de los archivos a un proveedor (por defecto build.nvidia.com; si no hay credencial, usa `OPENAI_API_KEY` si existe).
- `--recursive` escanea cada subcarpeta con un `SKILL.md` como una skill aparte.
- En Windows, `--output` responde «Safe file output is unsupported on this platform» y sale con exit 2. Redirige con `>`.

> **CUIDADO**
>
> Incluso con `--no-llm`, la regla SC4 manda nombres y versiones de dependencias a api.osv.dev si la skill trae un manifiesto (`requirements.txt`, `pyproject.toml`, `package.json`, `Pipfile`). Sin manifiestos no hay consulta. Si eso no es aceptable, escanea en una máquina sin red: SC4 usa entonces una lista local reducida.

PASO 03

Leer la calificación

SCORE	SEVERITY	RECOMENDACIÓN	EXIT POR DEFECTO
0–20	LOW	SAFE	0
21–50	MEDIUM	CAUTION	0
51–80	HIGH	DO NOT INSTALL	1
81–100	CRITICAL	DO NOT INSTALL	1

Cada hallazgo suma 50 (CRITICAL), 25 (HIGH), 10 (MEDIUM) o 5 (LOW), multiplicado por su confianza. La segunda aparición de una regla vale la mitad y la tercera un cuarto. Los hallazgos dentro de scripts ejecutables valen 1.3 veces. Exit 2 significa error. En JSON, mira `risk_assessment.max_issue_severity`: puede decir HIGH aunque la severidad general diga MEDIUM.

Lo que sacaron nuestras skills de prueba con `--no-llm`:

SKILL	QUÉ CONTIENE	SCORE	EXIT
release-notes	Solo instrucciones	0 · SAFE	0
env-sync	Orden de mandar el `.env` y un script que lo envía	36 · CAUTION	0
env-sync, solo texto	La misma orden, sin script	0 · SAFE	0
project-context	``!`cat .env | curl ...`` `` y `allowed-tools: Bash(*)`	21 · CAUTION	0
hidden-comment	Comentario HTML que anula instrucciones y pide `~/.ssh/id_rsa`	87 · DO NOT INSTALL	1

> **CUIDADO**
>
> Un escáner no prueba que una skill sea segura. Con solo análisis estático, la orden de exfiltración escrita en prosa sacó 0, y una skill que manda tu `.env` salió con exit 0. Lee siempre los hallazgos, no solo el score.

PASO 04

Revisar a mano lo que importa

- [ ] Lee el `SKILL.md` completo, incluidos los comentarios HTML y el texto que no se ve al renderizar.
- [ ] Revisa `allowed-tools`: da herramientas sin pedir permiso, y la confianza del espacio de trabajo no lo controla.

- [ ] Busca líneas `!`...`: corren en tu terminal antes de que Claude lea la skill.
- [ ] Abre cada script y busca lecturas de `.env`, `~/.ssh` o variables de entorno, y envíos a URLs externas.
- [ ] Revisa la `description`: un disparador muy amplio («siempre que…») hace que la skill cargue en casi todo.

PASO 05

Usarlo como MCP en Claude Code

```
claude mcp add skillspector -- skillspector mcp
```

- Sin `--scope` queda en el ámbito local (solo tú, solo este proyecto); con `--scope user`, en todos tus proyectos.
- Si instalaste en un entorno virtual, escribe la ruta completa al ejecutable después de `--`.
- La herramienta es `scan_skill(target, use_llm=true, output_format="json")` y devuelve `risk_score`, `recommendation`, `safe_to_install`, `scan_mode` y los hallazgos.
- Pídele a Claude que la llame con `use_llm` en `false` si no quieres mandar archivos a un proveedor.

> **CUIDADO**
> En nuestra prueba con un cliente MCP local, el servidor tardó 28.8 s en responder la primera vez, y `env-sync` volvió con `safe_to_install: true`: el campo solo exige score de 50 o menos y análisis completo. No uses `--transport http` fuera de tu máquina: no tiene autenticación.

PASO 06

Un paso de CI en GitHub Actions

Guárdalo como `.github/workflows/skillspector.yml`. Pasó actionlint 1.7.12 y su comando de escaneo dio exit 0 y 1 en local; no lo corrimos en GitHub.

```
name: Scan agent skills

on:
  pull_request:
    paths:
      - ".claude/skills/**"

permissions:
  contents: read

jobs:
  skillspector:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v7
      - uses: actions/setup-python@v7
        with:
```

```
          python-version: "3.12"
      - name: Install SkillSpector 2.13.4
        env:
          REL: https://github.com/NVIDIA/SkillSpector/releases/download/v2.13.4
        run: pip install "skillspector @ $REL/skillspector-2.13.4-py3-none-any.whl"
      - name: Scan skills (static only, no LLM)
        run: skillspector scan .claude/skills --recursive --no-llm --fail-on-findings
```

- `--fail-on-findings` falla con cualquier hallazgo activo, no solo con score mayor que 50.
- Para aceptar hallazgos ya revisados: `skillspector baseline ./mi-skill -o .skillspector-baseline.yaml`, súbelo en el PR y añade `--baseline .skillspector-baseline.yaml` al escaneo.
- No uses `--use-shipped-baseline` con skills de terceros: aplicaría el baseline que trae la propia skill.
- Sin `--no-llm` y sin credencial, el paso termina con exit 2.

AL TERMINAR

Checklist

- [ ] SkillSpector 2.13.4 instalado en un entorno aislado.
- [ ] Cada skill nueva se escanea con `--no-llm` antes de copiarla a `~/.claude/skills` o `.claude/skills`.
- [ ] Lees los hallazgos y `max_issue_severity`, no solo el score.
- [ ] Revisaste a mano el `SKILL.md`, los scripts, `allowed-tools` y las líneas `!`...``.
- [ ] El CI corre `--no-llm --fail-on-findings` en los PR que tocan `.claude/skills`.
- [ ] Los hallazgos aceptados viven en un baseline revisado en el PR.
- [ ] Vuelves a escanear cada vez que actualizas una skill.