— GUÍA PASO A PASO

Escanea tu código con *Claude* Security.

→ El plugin oficial de Anthropic para Claude Code, en tres comandos.

→ Del primer escaneo al parche aplicado, paso a paso.

Un equipo de agentes lee tu código como lo leería un investigador de seguridad: mapea la arquitectura, caza vulnerabilidades y descarta las falsas alarmas por votación. Corre en tu máquina. Tu código no sale de ahí.

AUTOR

Jorge Riquelme

Fundador · Otherway Labs

HERRAMIENTA

Claude Code

Plugin oficial · Seguridad · Beta

PASOS

06

— INTRODUCCIÓN

Nunca revisaste si *lo que subiste* era seguro.

Llevas meses construyendo con Claude Code. Publicaste features, conectaste APIs, guardaste claves en algún archivo. Y probablemente nunca te sentaste a revisar si eso era seguro. No es flojera: revisar seguridad a mano es lento, aburrido y hay que saber qué buscar.

Claude Security es el plugin oficial de Anthropic para eso. No busca patrones como un linter. **Sigue cómo se conecta tu código entre archivos** y va tras lo que de verdad importa: claves expuestas, logins que se pueden saltar, permisos mal puestos, errores de lógica.

Y antes de mostrarte algo, sus agentes lo discuten entre ellos y votan para descartar las falsas alarmas. Por eso los informes son cortos y se leen.

→ LEE ESTO

Todos los comandos de esta guía están **probados** sobre un proyecto real en producción. Cópialos tal cual.

— CONTENIDO

01 Requisitos previos 03
02 Instalación en tres comandos 04
03 Los tres trabajos del menú 05
04 Los cuatro niveles de esfuerzo 06
05 Cómo leer el informe 07
06 Convertir hallazgos en parches 08
07 Lo que NO hace, y sus límites 09

Pasos	Lectura	Nivel
06	12min	Medio

PASO 01 · ANTES DE EMPEZAR

Cuatro cosas *que necesitas.*

- **Claude Code v2.1.154** o superior, en plan de pago.
- **Python 3.9.6** o superior, disponible como `python3`.
- **Git**, solo si vas a escanear cambios o generar parches.
- macOS, Linux o Windows. Da lo mismo.

Verifica las dos versiones antes de instalar nada:

```
claude --version
python3 --version
```

→ PLAN PRO Si estás en Pro, activa **Dynamic workflows** en `/config` antes de instalar. El escaneo los usa para coordinar sus agentes y sin eso no arranca.

CONTEXTO

Cómo trabaja por dentro

No es un linter con reglas fijas. Cada escaneo es un equipo de agentes que recorre seis etapas, y saber cuáles son te ayuda a leer el informe después:

01 **Inventario.** Parte tu repositorio en componentes. Cada carpeta de primer nivel queda escaneada o descartada con una razón escrita.

02 **Modelo de amenazas.** Un agente por componente arma qué se podría atacar ahí.

03 **Investigación.** Un agente por cada cruce de componente y categoría de vulnerabilidad.

04 **Barrido.** Una pasada extra sobre lo que la matriz anterior no alcanzó a cubrir.

05 **Panel.** Tres verificadores independientes cuya pega es *desmentir* cada hallazgo. Solo pasa lo que dos de tres confirman.

06 **Adversarial.** Solo en el nivel más alto: un refutador de red team contra cada sobreviviente.

> *La etapa 5 es la que hace la diferencia. Los verificadores trabajan desde el código, no desde el reporte, y tienen orden de declarar falsa alarma salvo que puedan trazar una ruta real de explotación.*

PASO 02 · INSTALACIÓN

Tres comandos *y listo.*

2.1 Instala el plugin

Dentro de una sesión de Claude Code, escribe:

```
/plugin install claude-security@claude-plugins-official
```

Si te dice que no encuentra el marketplace, agrégalo primero y vuelve a intentar:

```
/plugin marketplace add anthropics/claude-plugins-official
```

2.2 Actívalo sin reiniciar

```
/reload-plugins
```

2.3 Confirma que quedó bien

Desde tu terminal normal, fuera de Claude Code:

```
claude plugin list
```

Tiene que aparecer `claude-security` con `Scope: user` y estado `enabled`. Ese *scope user* es la parte importante: significa que el plugin queda disponible en **todos tus proyectos**, no solo en el que tenías abierto.

2.4 La versión de terminal

Lo mismo sin entrar a una sesión. El `--scope user` es lo que lo deja global, y `uninstall` lo saca si algún día quieres:

```
claude plugin install claude-security@claude-plugins-official --scope user
claude plugin uninstall claude-security
```

→ TIP ¿Ya tenías el marketplace agregado hace tiempo? Córrele `claude plugin marketplace update` antes de instalar, para traer la **versión más reciente**.

PASO 03 · EL MENÚ

Un comando, *tres caminos.*

Todo el plugin vive detrás de un solo comando. Escríbelo y te abre un menú con tres opciones:

```
/claude-security
```

---

01 Scan codebase

Escanea el repositorio completo o un área acotada. Antes de preguntarte nada, el plugin **lee tu proyecto**: cuántos archivos tiene, qué carpetas guardan código real, en qué rama estás. Después te ofrece opciones concretas con el número de archivos de cada una. Si no sabes cuál elegir, responde *"I don't know"* y elige por ti.

02 Scan changes

Escanea solo lo que cambió: el diff de tu rama contra la base, el de un pull request abierto, o un commit suelto. Esto es lo que corres **antes de subir algo**. Ojo con una cosa: solo lee cambios ya commiteados. Si tienes trabajo a medias, haz commit o stash primero.

03 Suggest patches

Toma los hallazgos de un informe y los convierte en archivos de parche listos para aplicar. Puede correr sobre un informe que ya tienes, o hacer un escaneo fresco primero si el que tienes quedó viejo.

No estás obligado a pasar por el menú. Puedes pedirlo directo en español: *"escanea mi rama"*, *"revisa el commit abc1234"*, *"arregla el hallazgo F3"*.

→ TIP Activa el **modo auto** antes de escanear: aprieta `Shift+Tab` hasta que la barra de estado diga "auto mode". Sin eso, cada agente del escaneo te pide permiso y el proceso queda esperándote.

PASO 04 · ESFUERZO

Cuánto quieres *que busque.*

Cada escaneo corre a uno de cuatro niveles. Puedes pedirlo por nombre o dejar que el plugin lo elija según el tamaño de lo que vas a revisar.

`low`	Un solo investigador sobre todo el repositorio, más el panel de verificación. Triage rápido, igual de verificado.
`medium`	El pipeline completo: inventario del repo, modelo de amenazas, un investigador por cada componente × categoría, un barrido de relleno y el panel. **Es el default** y el que deberías usar casi siempre.
`high`	Igual que medium pero con inventario más amplio, **dos** investigadores por celda y dos barridos.
`max`	Igual que high más una fase adversarial: los hallazgos dudosos vuelven al panel y cada sobreviviente enfrenta a un refutador de red team.

> *El panel de verificación son siempre tres votantes, en los cuatro niveles. Bajar de nivel reduce cuánto busca — nunca cuánto verifica.*

QUÉ CAZA

Cuatro categorías

Inyección y entrada

SQL, comandos, código, XSS, XXE, deserialización, plantillas, path traversal y prompt injection.

Auth y accesos

Login que se salta, permisos faltantes o mal puestos, IDOR, escalada de privilegios, CSRF, SSRF, open redirect.

Cripto y secretos

Cripto mal usada, aleatoriedad débil, reuso de claves, canales laterales de tiempo y **claves hardcodeadas**.

Memoria y operaciones inseguras

Desbordes de buffer, use-after-free, overflow de enteros, confusión de tipos, FFI sin proteger.

→ TIP Parte con **medium sobre un área acotada**, no con el repo entero. Si sale limpio, sube a high sobre esa misma área antes de escanear todo.

PASO 05 · EL INFORME

Qué te deja *en disco.*

Cada escaneo escribe una carpeta con la fecha y hora en tu repositorio. Adentro hay tres archivos y nada más:

```
CLAUDE-SECURITY-20260727-143022/
  CLAUDE-SECURITY-RESULTS.md
  CLAUDE-SECURITY-RESULTS.jsonl
  CLAUDE-SECURITY-REVISION-a1b2c3d4e5f6.json
```

5.1 El informe que lees

El archivo `.md`. Cada hallazgo trae un identificador corto (`F1` , `F2` ...), el impacto, un escenario concreto de explotación, las precondiciones que necesita, la severidad, la confianza y una recomendación. Ese identificador es el que usas después para pedir el parche.

5.2 Los mismos datos en JSON

El archivo `.jsonl` : un objeto JSON por línea. Sirve si quieres meter los hallazgos a otra herramienta o a un tablero.

5.3 El sello de revisión

Dice qué commit exacto se escaneó, a qué nivel, cuántos hallazgos hubo por severidad y qué tan verificado quedó el escaneo. Si el nombre del archivo termina en `-dirty` , significa que había cambios sin commitear en el árbol que se revisó.

LO MÁS IMPORTANTE

La sección Coverage

Es la parte que casi nadie lee y la que más vale. Dice qué se revisó y **qué no**, con la razón de cada exclusión: código generado, dependencias de terceros, documentación. Un informe limpio sin mirar esa sección no significa "está todo bien" — puede significar "no lo miré".

→ OJO La carpeta trae su propio `.gitignore` , así que un `git add .` distraído nunca la sube. Si quieres el informe en el historial, borra ese archivo y commitea la carpeta como cualquier otra.

PASO 06 · LOS PARCHES

De hallazgo *a arreglo.*

6.1 Pídelos

Corre `/claude-security` y elige *"Suggest patches"*, o pídelo directo: *"arregla el hallazgo F3"*. Después eliges cuáles de los hallazgos quieres atacar.

Si el código cambió desde el escaneo, esos hallazgos se saltan con una nota. El plugin **no escribe un arreglo contra código que nunca vio**: te ofrece un escaneo fresco en vez de parchar desde un informe viejo.

6.2 Cómo se construye cada uno

El parche se escribe en una copia aparte de tu repositorio. **Tu carpeta de trabajo no se toca.** Después, un agente distinto al que lo escribió lo revisa: corre los tests de tu proyecto contra el cambio y lee el diff por su cuenta.

El parche solo se emite si esa revisión puede responder que sí a las tres:

01 ¿Arregla ese hallazgo, y solo ese?

02 ¿No introduce una vulnerabilidad nueva?

03 ¿Deja el comportamiento igual? Y cambiar qué datos acepta el código **cuenta como cambio de comportamiento**.

Si no puede responder que sí a las tres, recibes una nota corta explicando por qué, en vez del parche.

6.3 Aplícalos tú

Los parches quedan en la carpeta `patches/` del informe, uno por hallazgo. Se aplican desde tu terminal:

```
git apply CLAUDE-SECURITY-20260727-143022/patches/F1.patch
```

→ TIP Aplica **un parche por pull request**, para que se revise y se pruebe solo. Si el código parchado no tiene tests, el parche lo dice: significa que la garantía descansa en revisión, no en una corrida de pruebas.

— ANTES DE CORRERLO

Lo que *no* hace por ti.

Nada se aplica solo. No hace commit, no hace push, no abre pull requests y no aplica ningún parche por ti. Deja archivos en disco y tú decides. Tampoco hace llamadas de red durante el escaneo: nada de tu código sale de tu máquina.

No corre aislado. Se ejecuta en tu sesión, con tus permisos y tu configuración. Está pensado para tu propio código. Si vas a revisar un repositorio de terceros que no controlas, corre toda la sesión dentro de un entorno aislado.

Una cosa más: los escaneos no son deterministas. Dos pasadas sobre el mismo código pueden encontrar cosas distintas, y encuentran más a medida que los modelos mejoran. Esto no reemplaza tus escáneres de siempre — los complementa. Córrelo seguido.

— CHECKLIST DE ACTIVACIÓN · 10 MINUTOS

01 Verifica `claude --version` y `python3 --version`.

02 Instala el plugin y corre `/reload-plugins`.

03 Activa el modo auto con Shift+Tab.

04 Escanea un área acotada a nivel medium, no el repo entero.

05 Lee la sección Coverage antes que los hallazgos.

06 Pide parches solo de lo que entendiste. Aplícalos de a uno.



— SOBRE EL AUTOR

Jorge *Riquelme.* Fundador · Otherway Labs

Construye **plataformas internas y automatizaciones con IA** para empresas en Chile y LATAM. Publica guías, MCPs y skills de Claude en `otherwaylabs.cl` e `@jorgeriquelme.ia`.

— PRÓXIMOS PASOS

Tres formas *de seguir avanzando.*

— PARA EMPRESAS
Sesión gratuita

¿Hay procesos en tu empresa que *deberían correr solos*?

Agenda una sesión y revisamos juntos qué parte de tu operación puede automatizarse con IA este mes.

Agenda en
otherwaylabs.cl →

— PARA EQUIPOS
Capacitación

Forma a tu equipo para que *use IA todos los días*.

Capacitaciones prácticas en Claude para equipos técnicos y no técnicos, con casos de tu propia operación.

Revisa en
otherwaylabs.cl/capacitaciones →

— ACCESO ANTICIPADO
Grupo WhatsApp · Gratis

Recibe estos recursos *antes que nadie*.

Grupo exclusivo donde publico cada guía, MCP y skill antes de subirla a YouTube o Instagram. Sin esperar a que salgan los videos.

Únete en
chat.whatsapp.com →