PLAIN-ENGLISH GUIDE

OCTOBER 2026 · ANTHROPIC + THEOREM LABS

PROOF-CHECKED CODE, EXPLAINED

Claude wrote machine-checked proofs that 239 entry points across 123 open-source libraries won't crash on any input up to 64 MB, under clear assumptions. This guide explains what that means, what it doesn't, and what you can do with it, without any math.

01
What happened
The project, the numbers and the bugs it found, in plain words.

02
The limits
What "proved" covers, what it doesn't, and why tests still matter.

03
Use it
Who this helps, how the Claude skill works, and questions to ask.

01 CLAUDE WROTE THE PROOFS

PUBLISHED OCT 22, 2026

Anthropic and its collaborator Theorem asked Claude to prove that real open-source programs never crash on a whole class of inputs. Then a separate program, the Rocq proof checker, checked every proof.

123
open-source libraries (from Google's OSS-Fuzz)

239
entry points proved crash-free

~180M
lines of proof in total

A few weeks
for the whole OSS-Fuzz effort

HOW DEEP IT GOES

- The proofs are about the compiled program: the RISC-V instructions the processor actually runs, not just the source code.
- A typical proof runs to about 600,000 lines, for a program of about 18,000 instructions.

WHAT IT FOUND

- Hundreds of bugs surfaced while trying to prove the code was safe.
- One logic bug in a critical application had been hiding for more than a decade.
- In one week on a widely used PDF reader, as many security bugs as are usually reported in a year.

Why it's news: formal proofs used to be rare and slow. Anthropic's example: the seL4 kernel took 2 person-years to build and 20 person-years to verify.

02 THE ONE IDEA TO REMEMBER
TESTED VS PROVED

A test tries some inputs. A proof covers every input in its range, including the ones nobody thought to try.

SIDE BY SIDE

TESTING (FUZZING)	PROOF
Throws lots of random inputs at the code	Shows mathematically that no input in range can crash it
Finds bugs it happens to hit	Can't pass while a crash is still possible
"We didn't see a crash"	"There is no crash, under these assumptions"
Cheap and fast to run	Expensive to make (hundreds of millions of tokens), cheap to re-check

1. **They turned a test into a promise**
   Each fuzz test became a written rule (a "spec"): this program won't crash on any input under 64 MB.

2. **Claude wrote the proof**
   When a proof failed, that often pointed at a real bug. Claude patched it and tried again.

3. **A checker, not a person, reviews it**
   The Rocq proof checker confirms each step, so nobody has to read 180 million lines or trust Claude's explanation.

03 WHAT "PROVED" COVERS

READ BEFORE YOU REPEAT THE HEADLINE

Anthropic says it plainly: "formally verified" does not mean "bug-free". A proof is only as good as the rule it proves and the model of the machine it assumes.

THE GUARANTEE ASSUMES

- Inputs under 64 MB
- A single-threaded, static program built with musl libc (not the usual glibc builds most people run)
- A fixed memory budget
- A modeled RISC-V processor and Linux system

IT DOESN'T COVER

- Every entry point of every library (not all were verified)
- Every kind of memory bug (the crash detector, ASan, doesn't catch all of them)
- Logic that's wrong but doesn't crash
- Bugs in the hardware, OS or the checker's own tools

WHAT WAS RELEASED

- A subset of the proofs, plus the RISC-V and Linux process models, so anyone can re-check them.
- Proofs that could reveal still-unpatched bugs are being held back.
- These are the project authors' own research results, not an independent audit.

! **Keep your tests.** A proof makes one strong promise in a defined setting. Tests, reviews and fuzzing still catch everything outside it.

04 THEOREM'S CLAUDE SKILL
ADD YOUR OWN PROJECT

Theorem published a Claude skill (oss-verify, "add-project") that walks a maintainer through enrolling an open-source project. Theorem also offers to run the same pipeline on other open-source projects.

WHO THIS HELPS

- A solo maintainer of a small library that reads untrusted files, who can't pay for a security audit
- Teams whose apps depend on parsers, image or PDF libraries
- Anyone choosing between two open-source libraries: a published proof is a strong signal

WHAT YOU NEED

- You maintain the project (only the maintainer opens the pull request)
- C, C++, Rust or Python code
- A copy of the oss-verify repo and a working build setup

1. **Say what "correct" means**
   The agent asks you about entry points, untrusted inputs, size limits and rules that must always hold. It is told not to guess.

2. **Write the fuzz targets**
   Each rule becomes a check, shown to run and shown to be able to fail.

3. **Port the build and check it**
   From Theorem's templates, then run its validation tools until the build repeats exactly.

4. **Open the pull request**
   Only your project's folder changes. The skill notes that AI findings and patches can be wrong or incomplete.

05 NO CODE NEEDED
CHECK ANY "VERIFIED" CLAIM

"AI-verified" will show up in a lot of marketing. Paste this into Claude, ChatGPT or Gemini with the claim you want to understand.

VERIFICATION CLAIM CHECKER · COPY

```
I want to understand a claim that some software is "verified" or "proved safe".
The claim and its source: [paste the text or link]
Explain in plain English:
1. What exactly was proved (the rule, or "spec"), in one sentence.
2. Which inputs, sizes and settings the proof covers, and which it doesn't.
3. What has to be trusted: the checker, the machine model, the compiler, the hardware.
4. Whether the proofs are published so others can re-check them.
5. Who made the claim, and whether anyone independent has checked it.
6. What I should still test myself before relying on this software.
Keep it short and flag anything the source doesn't say.
```

GREEN FLAGS

- The exact rule and limits are written down
- Proofs are published and machine-checkable
- Limits are stated up front, like Anthropic's post does

RED FLAGS

- "Bug-free" or "100% secure" with no limits
- No published proof, just an AI's explanation
- No word on which version or build was checked

AI STACK SIGNAL

FOLLOW AI STACK SIGNAL
CUT THROUGH THE AI NOISE

Short, fact-checked reels on the AI tools worth your time, with the catch included.

news.aistacksignal.com/claude-proofs

SOURCES · CHECKED OCT 23, 2026

Anthropic: LLMs are making formal software verification feasible at a much larger scale · anthropic.com

Theorem Labs: published proofs · github.com/theorem-labs/oss-verify-proofs

Theorem Labs: add-project Claude skill · github.com/theorem-labs/oss-verify

Independent guide. Not affiliated with or endorsed by Anthropic, Theorem or any project named. Figures are as published by Anthropic on Oct 22, 2026 and describe the authors' own research results under stated assumptions; they are not an independent audit. A formal proof covers only its written specification and modeled environment. Keep testing your software.