

Hacking the Truth

Forged Lean Proofs and the Risk of Verifier Bugs

Jonathan Brossard, Moabi
JP Aumasson, Cloudflare

[un]prompted II



Math proofs created by machines [AI]

Home → Science → News

OpenAI released 722 math papers at once and mathematicians are freaking out

The "romantic" age of mathematics might be over.

FINITE TIME BLOWUP FOR NAVIER-STOKES

OPENAI

ABSTRACT. For every positive viscosity, we construct a solution of the three-dimensional incompressible Navier-Stokes equations that starts from rest and develops unbounded velocity in finite time while maintaining uniformly bounded kinetic energy.

Verified by machines

Navier-Stoke: 707 files, 412,645 lines of **Lean**

```
2975 theorem temporalIncrementState_supportedGauge (h : ℝ) (index : ℕ → ℕ)
2976   (axial : PressureStream.Plane × PressureStream.Plane) (n : ℕ)
2977   (hf : ContDiffOn ℝ ∞ (u.axialResidual c n) (PhysicalMeanDomain.slowDomain U.carrier))
2978   (hp : PhysicalMeanDomain.PeriodicOn U.carrier (u.axialResidual c n))
2979   (hsz : SupportedGauge g.radial.inner g.radial.outer (qLength coord) U.carrier (u.axialResidual c n))
2980   (hsθ : SupportedGauge g.radial.inner g.radial.outer (qLength coord) U.carrier (u.thetaResidual c n)) :
2981   SupportedGauge g.radial.inner g.radial.outer (qLength coord) U.carrier
2982     ((temporalIncrementState g h index axial c u).radial n) ∧
2983   SupportedGauge g.radial.inner g.radial.outer (qLength coord) U.carrier
2984     ((temporalIncrementState g h index axial c u).angular n) ∧
2985   SupportedGauge g.radial.inner g.radial.outer (qLength coord) U.carrier
2986     ((temporalIncrementState g h index axial c u).axial n) := by
2987   have hL : ContinuousOn (qLength coord) U.carrier :=
2988     ((qLength_contDiffOn U.coord_pos U.coord_lt_one).mono (fun s hs => U.time_pos s hs)).continuousOn
2989   have hpot := streamPotential_supportedGauge (M := g.radial.frequency n) ha g.radial.inner_lt_outer hd
2990     (qLength coord) g.radial.radialDirection U.isOpen
2991     (fun s hs => qLength_pos U.coord_pos U.coord_lt_one (U.time_pos s hs))
2992     (temporalAtIndex_contDiffOn h n (index n) U.isOpen hf hp)
2993     (temporalAtIndex_supportedGauge h n (index n) hsz)
2994   have hB := streamBeta_supportedGauge U.isOpen hL hpot (c.operators.epsilon n • axial)
Type : qa and press <Enter> to exit Vim
[1] 0:unprompted- 1:navier-stoke* "vi VariableGaugeMean."
```

Theorems verified by humans?

Nobody Will Read the Lean: A Case for High-Level Cryptographic Languages

Nadim Kobeissi
Symbolic Software
nadim@symbolic.software

October 2026

Abstract

In September 2026, an AI produced a Lean proof of Fermat's Last Theorem in eleven days, and AI provers have started on cryptography. Within a few years, an AI may hand you a machine-checked proof that Signal is secure. You won't need to read it, because Lean will have checked every step. What Lean can't check is whether the theorem says what you think it says. That depends on its definitions, which the AI will also have written at length for that one proof. This already happens. A Lean library's security theorem for Schnorr signatures passed the checker, yet as written it held for every adversary and so promised nothing.

When proofs are cheap, the scarce resource is a person who can read what was proved. I

<https://eprint.iacr.org/2026/2441.pdf>

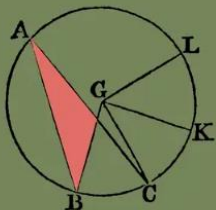
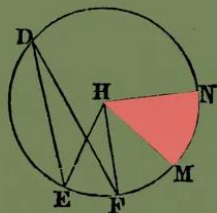
Proofs verified by humans?

Point 1 is sort of addressed by the presence of Lean and other aspects of formal mathematics. As for verification, automated formalisers are getting better and better, e.g. having almost autonomously formalised the proofs of sphere packing bounds due to Viazovska and collaborators. This is also discussed here on MO, but I think the point of the discussion on MO is that **one should not stop trying to understand a proof just because there is a Lean proof**. Again, as Thurston already said, the point is in the human understanding of the proof, not just the formal correctness.

Anthropic on the trust stack

ANTHROPIC

LLMs are making formal software verification feasible at a much larger scale



While this is an enormous volume of proof code, virtually none of it needs to be manually reviewed. The Rocq proof checker ensures that the spec is correctly satisfied by the proof. As a result, the only artifacts that a developer needs to trust are:

- The formal models of RISC-V and of Linux processes;
- The exact theorem statements that were proved;
- The Rocq proof checker itself. ⁵

When the machine fails

Postmortem for Kernel Soundness Bug #14576

2026-08-01

A soundness bug in the Lean kernel ([#14576](#)) was reported and fixed during the week of July 27. It has had visibility on Zulip and social media (e.g., X, LinkedIn, and Mastodon).

What happened

On July 25, Ramana Kumar published a repository containing a `sorry`-free "disproof" of the Collatz conjecture, produced with AI assistance. It is not a valid proof because it exploits a bug in the kernel's handling of nested inductive types. On July 28, Kiran Gopinathan reduced it to a small proof of `False` and opened issue [#14576](#). We pushed a fix one hour after the report ([#14577](#)). [Joachim Breitner](#) reviewed it and suggested improvements, and it was merged. New patch releases are out.

The bug: when the kernel eliminates a nested occurrence under an inductive type `T` with parameters `Ds`, and these parameters are phantom (not mentioned in constructor fields), they disappear from the generated auxiliary type and thus escape type checking. An ill-typed argument in that position could be used to make the kernel accept a proof of `False`. The bug is only reachable through metaprogramming, by sending the inductive declaration to the kernel directly. The frontend checks the arguments and catches the ill-typed term. **This is an implementation bug, not a hole in Lean's meta-theory.**

<https://leodemoura.github.io/blog/2026-8-1-postmortem-for-kernel-soundness-bug-14576/>

When two machines fail

Why nanoda did not catch it

The original Collatz repository also passed a week-old version of [nanoda](#), the main external checker. nanoda is an independent kernel (aka proof/type checker) for Lean implemented in Rust by [Chris Bailey](#). The surprising part is that there are two unrelated bugs involved. The official kernel had a missing check in the nested inductive type support, as explained above. nanoda did check that spot, but did not verify the type name in a projection node. The nanoda bug was reported by Jeremy Chen and fixed a week before the Lean bug was reported. The proof was built so that the expression the kernel never inspects is one that the old nanoda accepted.

Ramana believes the timing was coincidental, but cannot rule out that the model had seen the nanoda report. Joachim proposed the hypothesis that the timing coincidence is due to the availability of strong models able to find this bug.

<https://leodemoura.github.io/blog/2026-8-1-postmortem-for-kernel-soundness-bug-14576/>

Complex software and trust...

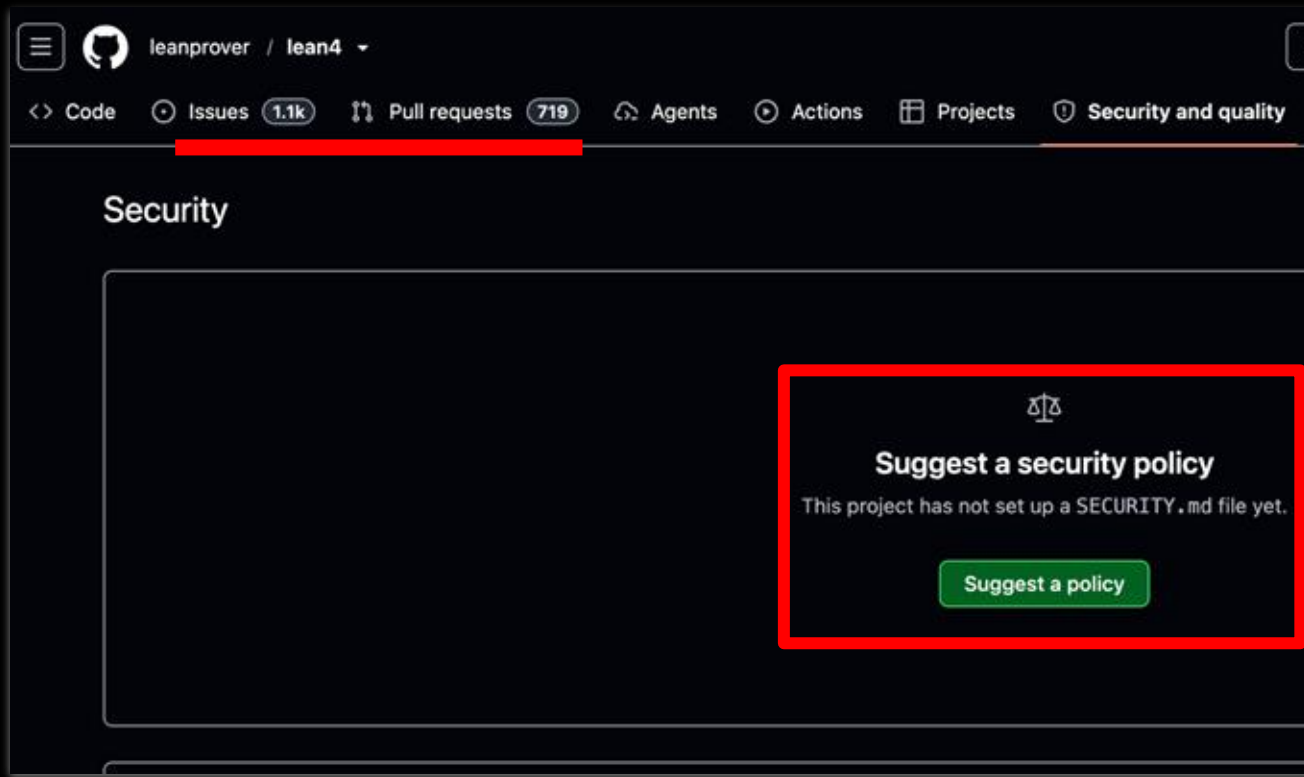
A critical look at Lean4

by Milo Moses 9th Oct 2026

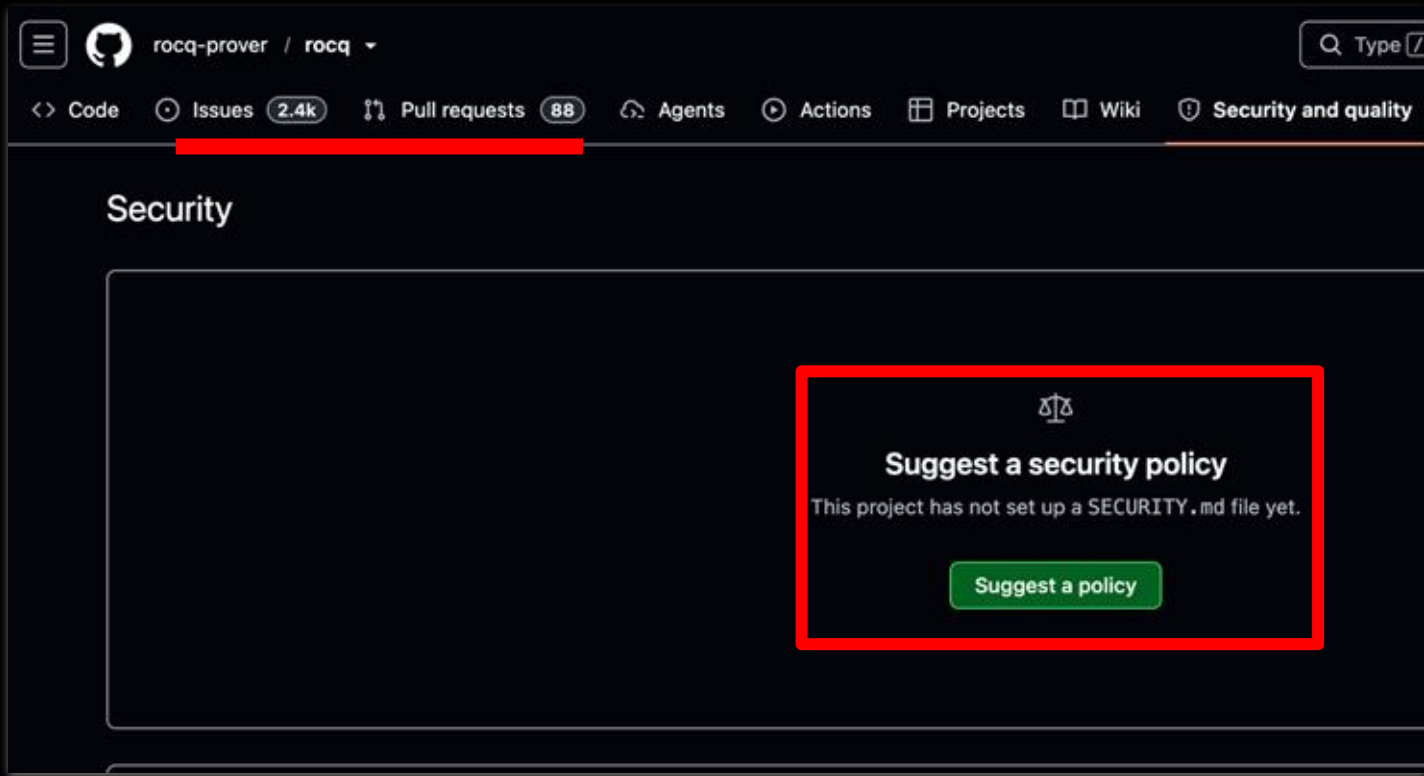
Lots of the discussion around Lean certificates and trust centers around general philosophical ideas of how much humans should trust formal certificates. It's important to remember, though, that Lean4 is not an ideal formal certificate machine: Lean4 is a software in its infancy and has issues. There are reasonable reasons to specifically not trust Lean4.

<https://www.lesswrong.com/posts/zHhGa3iQBQcdSifFL/a-critical-look-at-lean4-1>

Lean's security exposure



Rocq's security exposure



The point of this talk

LLMs + Lean = **superpower** = greater responsibility

The bug **tolerance threshold** is now lower

Soundness bugs must be **treated as security bugs**

Risks:

- Deliberate or accidental **proof forgery**
- Insecure formally **verified code**

From bugs to CVEs

7 known bugs → 7 CVEs (2 Lean, 5 Rocq)

At filing (Sept '26): 4 fixed, 3 unpatched

Other PoC'd bugs in verification tools Agda, F*, Dafny, Isabelle/HOL, Z3 and cvc5 (no CVEs yet)

Supply chain risk: Z3 bugs led code verifiers Verus and Frama-C to accept false claims

"Proof" of the Riemann Hypothesis

CVE-2026-72714 · Rocq #22287 (unpatched)

Correct statement of a known inequality
equivalent to RH (Lagarias's criterion)

$$\sigma(n) \leq H_n + \exp(H_n) \ln(H_n) \quad \text{for all integers } n \geq 1$$

$\sigma(n)$: sum of divisors of n $H_n = 1 + \frac{1}{2} + \dots + \frac{1}{n}$

Bug exploit, not RH-specific: the checker accepts a
contradiction: **ex falso quodlibet** $\rightarrow$ **RH**

Risky Take...



JP Aumasson ✓ @veorq · Sep 12



I conjecture that no LLM, even with an infinite amount of tokens, can find an efficient attack on AES, BLAKE3, ChaCha, or SHA3, for no such attack exists (also, $P=NP$)

[with infinite tokens/time it could theoretically break these algorithms, but not find a non-trivial attack,

[Show more](#)



10



4



93



7K



Arthur Lazzaretti ✓ @rthurknl · Sep 13



How confident are you? I would take the other side of this bet: "There will be a serious attack found for at least one of AES, BLAKE3, ChaCha, or SHA3 in the next five years." Serious would mean there is a set of inputs of non-negligible size for which the security is broken



1



104



JP Aumasson ✓
@veorq



serious as in "I get tattooed 'SHA3 > BLAKE3' if that happens"

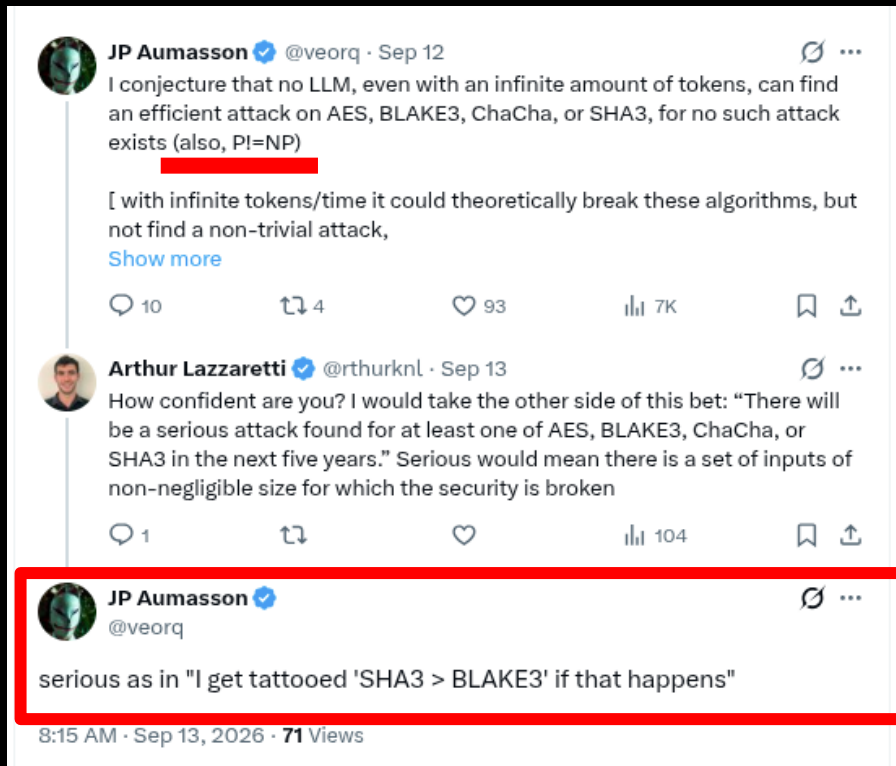
8:15 AM · Sep 13, 2026 · 71 Views

Game on...

I have a Proof of $P = NP$...

With a Lean 4 Certificate 🤖

I can recommend a few good tattoo artists... 😈



The screenshot shows a Twitter thread with three tweets. The first tweet by JP Aumasson (@veorq) from Sep 12 states a conjecture that no LLM can find an efficient attack on AES, BLAKE3, ChaCha, or SHA3, with a redacted portion. The second tweet by Arthur Lazzaretti (@rthurknl) from Sep 13 responds with a bet. The third tweet, highlighted with a red border, is a follow-up by JP Aumasson from Sep 13 at 8:15 AM, stating he is serious as in "I get tattooed 'SHA3 > BLAKE3' if that happens".

JP Aumasson @veorq · Sep 12

I conjecture that no LLM, even with an infinite amount of tokens, can find an efficient attack on AES, BLAKE3, ChaCha, or SHA3, for no such attack exists (also, $P=NP$)

[with infinite tokens/time it could theoretically break these algorithms, but not find a non-trivial attack, [Show more](#)

10 4 93 7K

Arthur Lazzaretti @rthurknl · Sep 13

How confident are you? I would take the other side of this bet: "There will be a serious attack found for at least one of AES, BLAKE3, ChaCha, or SHA3 in the next five years." Serious would mean there is a set of inputs of non-negligible size for which the security is broken

1 104

JP Aumasson @veorq

serious as in "I get tattooed 'SHA3 > BLAKE3' if that happens"

8:15 AM · Sep 13, 2026 · 71 Views

Lean Certificates of $P = NP$ and $P \neq NP$

CVE-2026-72844 · Lean #14576 (patched)

Type confusion bug leading to logical explosion

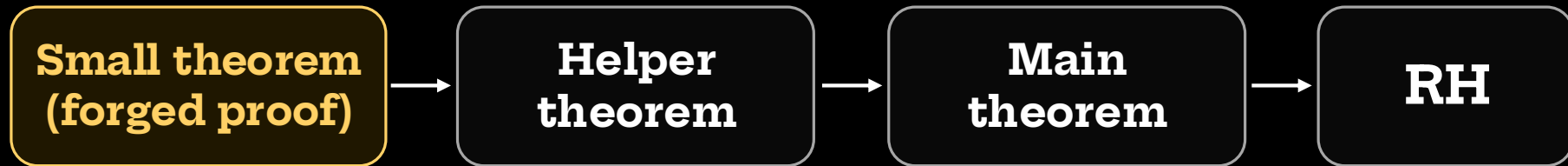
$P = NP$: Fake a polynomial-time 3-SAT solver

$P \neq NP$: Forge a proof that $3\text{-SAT} \notin P$

Lean certificates pass leanchecker

Towards more plausible "proofs"

A long proof can bury a bug exploit several layers down in an imported library:



A sound-looking proof attempt could fool agent/human reviewers for a few days/weeks/?

Fiat-Crypto sabotage PoC

Rocq #21683 - CVE-2026-72705 (fixed)

Fiat-Crypto outputs verified field arithmetic
Used in BoringSSL and other projects

Sabotaged multiplication always returns zero
`carry_mulmod f g := zeros n`

Property `output = f × g (mod m)` still "verified"

Real exploitation tactics

Actual attacks must be **more subtle** and:

- Exploit 0-day bugs
- Fool agents doing code review
- Pass unit tests (like “bug attacks”)
- Remain plausibly deniable
- End up in critical systems

Code execution from malicious Lean

Ref count overflow to UAF to RCE, via heap spray
From public bug <https://github.com/leanprover/lean4/pull/15241>

No `unsafe`, FFI, or metaprogramming needed

Exploit scenarios (client or server):

- Malicious Lean dependency sabotage
- Hosted Lean service compromise
- Hosted CI compromise

Lean 4

Tool	Issue	CVE	Bug Summary	Level	Notes
Lean 4	#14576	CVE-2026-72844	Nested inductive projection type confusion	Kernel	1st CVE ever for PA kernel
Lean 4	#14484	CVE-2026-72711	Kernel accepts opaque decl with unbound free variable	Kernel	Fixed <4.32.2
Lean 4	#15241	—	Deletion cascade use-after-free (sticky refcount) — RCE via closure hijack	Runtime	Fixed v4.34.1



Tool	Issue	CVE	Bug Summary	Level	Notes
Rocq	#12155	CVE-2020-37268	Print Assumptions omits unsafe universe checking via Parameter Inline	Kernel	Unfixed since 2020
Rocq	#21682	CVE-2026-72703	Guard checker: non-terminating fixpoint via unchecked cross-calls	Kernel	Found by Opus
Rocq	#22024	CVE-2026-72704	Guard checker trusts corrupted recursive tree after transport	Kernel	0-day
Rocq	#21683	CVE-2026-72705	Guard checker: fixpoint as higher-order argument	Kernel	Found by Opus
Rocq	#22287	CVE-2026-72714	Universe checking desync after module close	Kernel	0-day; Millennium forgeries

SMT Solvers: Z3 & CVC5 🔥

Tool	Issue	CVE	Bug Summary	Level	Notes
Z3	#7842	—	FP + algebraic datatypes: sat with invalid model	Oracle	No CVE requested
Z3	NRA regr.	—	Refutation unsoundness on NRA (unit circle variant); regression across versions	Oracle	Exploited via Verus & Frama-C/WP
CVC5	#13012	—	Soundness bug	Oracle	PoC pinned v1.3.4

Isabelle, Agda, Dafny & F*

Tool	Issue	CVE	Bug Summary	Level	Notes
Isabelle/HOL	NBE bug	—	TYPE(T) args generalized to TYPE(?'a) by normalizer → proves False	Oracle	Fixed Isabelle2025-1
Agda	#7668	—	Positivity checker: identity type in negative position (Cubical) → ⊥	Kernel	Fixed 2.8.0
Dafny	#4053	—	Soundness bug	—	PoC v4.1.0
Dafny	#6158	—	Soundness bug	—	PoC v4.10.0
F*	#4480	—	Soundness bug	—	Docker PoC

Derived Exploits: Forgeries & CompCert

Exploit	Base Bug	Target	Description	Status
P=NP Forgery (Lean)	CVE-2026-72844	Lean 4 kernel	Lean 4 certificate proving P=NP; leancchecker passes	Confirmed
P=NP Forgery (Rocq)	CVE-2026-72714	Rocq kernel	Built by JP Aumasson (veorq)	Confirmed
Riemann Hypothesis	CVE-2026-72714	Rocq kernel	Lagarias criterion forgery; Print Assumptions clean	Confirmed
Navier–Stokes	CVE-2026-72714	Rocq kernel	Both regularity and its negation proven	Confirmed
BSD Conjecture	CVE-2026-72714	Rocq kernel	Rank equality forgery	Confirmed
Hodge Conjecture<	CVE-2026-72714	Rocq kernel	Print Assumptions whitelist clean	Confirmed
CompCert Backdoor	CVE-2026-72714	CompCert	Forged semantic-preservation proof for malicious optimization pass	Confirmed

Conclusion

FV tools need **higher-assurance SSDLC**, since they are:

- Complex software; bugs are inevitable
- Processing complex, **human-un(der)reviewed** code
- **No longer niche:**
 - The foundation of many math results
 - A target for attackers

Article in **Phrack**, PoCs code *soon* at

<https://github.com/endrazine/owning-the-truth>

Thank you! Q&A

endrazine@securityfocus.com

veorq@securityfocus.com

<https://moabi.com>

<https://bugtraq.io>

<https://aumasson.jp>

