

```
// smart contract vulnerabilities × large language models
```

Efficiently using LLMs to find and fix vulnerabilities in live protocols

Ivan Bogatyy

benchmarks · pipelines · real findings · what comes next

```
$ forge test --match exploit
```

```
✓ attacker balance +0.1 ETH [PASS]
```

Four questions

1

Can LLMs actually find bugs?

The new exploit benchmarks — and a 4-year Zcash bug found by Claude.

2

Where do we point them?

The bottleneck is budget, not IQ.

3

What can you find on \$2k?

\$2M ZK circuit exploit

4

How do we make models better?

RL the exploitation skill back into the weights.

// part 1 — can LLMs find bugs?

Benchmarks moved from “spot the bug” to “write the exploit”

How the new benchmarks work

Seed corpus

DeFiHackLabs — hundreds of real DeFi hacks reproduced as runnable Foundry PoCs.

Task

Fork the contract at a historical block; the agent must find the bug and write a working exploit.

Grading

Binary: did it extract real profit (≥ 0.1 native token)? Sandbox restarts before scoring — no cheats.

SCONE-bench Anthropic FRT · after the knowledge cutoff

65%

Opus 4.5 alone on post-cutoff hacks (13 / 20)

100%

Claude Mythos Preview, 2026 — all 12 fresh exploits

- **Frontier trio** 56% of 34 post-cutoff hacks (19/34) → \$4.6M
- **Scale** Mythos drained ~\$35M, ≈75% ahead of any rival.
- **Context** Full 405-set incl. pre-cutoff: 51% / \$550M — treat as contaminated. Doubling ≈ 0.7 mo.

// part 1 — everyone is building these now

Two more benchmarks, two more angles

EVMbench OpenAI × Paradigm · Feb 2026

117 vulns from 40 audits (Code4rena, Tempo). Three modes: detect · patch · exploit.

71% GPT-5.3-Codex

exploit mode — vs 33% for GPT-5, one generation earlier

⚠️ Caveat OpenZeppelin's review found training-data contamination and several “high-severity” bugs that aren't actually exploitable.

Crypto Security Bench V12

289 vulns across 31 projects in Solidity, Rust, C++ and Go. Metric is recall — how many known bugs you catch.

V12	73.6%
Claude Code · Opus 4.7	57.1%
Codex · GPT-5.5	46.2%

Solidity-only recall

⚠️ Caveat vendor-run — V12 tops its own leaderboard against self-curated ground truth.

Takeaway the capability is real and rising fast — but ground truth is contested. Trust post-cutoff, reproducible sets.

Zcash: a 4-year counterfeiting bug, found by Opus 4.8

The bug — Orchard shielded pool

The halo2 scalar-mult circuit never constrained the base point to the protocol's `g_d` — `assign_advice` where it needed `copy_advice`.

A prover could swap in an arbitrary base point → a different nullifier for the same note → spend it again and again. Undetectable counterfeiting of ZEC.

Latent since NUS (May 2022) — over four years.

Found by an AI-assisted audit

- **May 29, 2026** Taylor Hornby (Shielded Labs) found it with Opus 4.8.
- **Older models missed it** prior audits of the same circuit came up empty.
- **Fallout** emergency hard fork; ZEC fell ~30–40%.

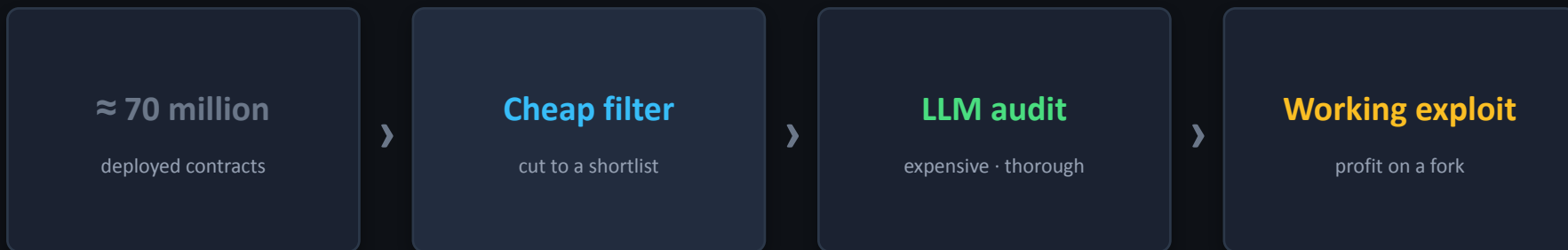
*And my vanilla Claude Code rediscovered it in ~1 hour!
(with some good pushing!)*

Why it matters the capability floor is high — an off-the-shelf model reproduced a bug four years of expert review had missed.

// part 2 — where do we point them?

The bottleneck isn't the auditor — it's what reaches it

Zero-day discovery is two stages. A cheap filter narrows millions of deployed contracts down to a shortlist; only then does the expensive LLM audit run. The filter decides which contracts the model ever sees.



Reframe: The "harness" is a list of contracts, not a list of tools

Better rules, then a learned model

Anthropic SCONE filter

14.9%

recall · 26/174

~382k candidates

`verified ∧ ERC-20 ∧ DEX-traded
∧ liq ≥ $1k`

≈4x
recall

Select better clauses

59.2%

recall · 103/174

~411k candidates

`verified ∧ tx ≥ 5 ∧
bytecode ≥ 2KB ∧ callers ≥ 5`

≈5x
denser

Train GBDT with cross-validation

38%

recall · range 34–42%

just 50k candidates

`GBDT ranker over on-chain
activity features`

Two moves swapping the rule predicate quadrupled recall at the same cost; a learned GBDT ranker then cut the audit set ~8x — trade a little recall for an order-of-magnitude cheaper audit.

// part 3 — a live mainnet sweep

3,200 contracts audited → 2 provable extractions

fork @ block 25.48M · ~\$1,169 total spend · every hit verified by cheatcode-free fork-profit

DELTA Deep Farming Vault

compound(address) is public and routes 1% of the target account's recycled DELTA + ETH to msg.sender

Loop it over ~110 decayed farmer accounts, skim 1% of each, swap DELTA → WETH → ETH.

+0.227 ETH

Notional V1 fixed-rate lending

Fork time-warped to 2026 → 113 fCash positions past maturity and underwater. Liquidating them via Escrow.liquidateBatch pays the 6% governance discount.

+0.118 ETH

Aztec Connect escape hatch: an unconstrained proof_id

The circuit

```
public_witness_ct(&composer, 0);
```

publishes proof_id — but never constrains it to 0.

Should have been

```
auto proof_id = public_witness_ct(&composer, 0);  
assert_equal_constant(proof_id.witness_index, 0);
```

Aztec Connect RollupProcessorV2 — barretenberg escape-hatch circuit.

What it lets you do

- **proof_id == 0** deposit / withdraw — the contract moves funds.
- **proof_id == 1** ops like key rotation — the contract skips moving funds.

The exploit: prove a real 150,000-DAI deposit but tag it proof_id = 1. Solidity skips the debit — you mint a 150k-DAI private note for free, then withdraw it.

Worst-case impact ≈ \$2M.

Whitehat reproduction Aztec Connect was already sunset; no funds at risk. Now a DeFiHackLabs PoC (AztecEscapeHatch_exp2.sol).

Today it's scaffolding. The frontier is RL into the weights.

The status quo — frozen models

Almost everything shipping is an agent loop wrapped around a frozen frontier model:

- **XBOW** autonomous pentest — #1 on HackerOne US
- **OpenAI Aardvark** GPT-5 agent that finds & patches flaws
- **PoCo · Prompt-to-Pwn** agentic PoC exploit generation for contracts

Capability comes from general RL + better scaffolding — not a specialist model.

Proven nearby — RL into the model

- **CTF-Dojo** 32B open model → 31.9% Pass@1, execution-verified flags
- **Crypto-CTF GRPO** procedurally-generated challenges, flag = reward
- **PrivEsc-LLM** Qwen3-4B → 95.8%, matches Opus at ~100× less cost
- **SmartCoder-R1** S-GRPO for secure Solidity — reward is a scanner

techniques RLVR · GRPO · verifiable rewards ·
off-policy (Prime-RL)

The exploit oracle already exists

No one has published RL-training an LLM to write DeFi exploits with a **Foundry “did-funds-move” reward**.

Security is one of the few domains outside math and code with a clean, cheap verifier — a fork test that checks whether the attacker's balance went up. The pieces are all here; nobody has combined them.

Reward

A Foundry fork-test — “attacker balance increased” — is a perfect, objective oracle.

Data

DeFiHackLabs as a seed corpus; procedural generation to scale verifiable tasks.

Hard part

Sparse, long-horizon reward over expensive rollouts → off-policy recipes.

Dual-use *the reward that trains a finder trains an attacker. Worth naming out loud.*

// takeaways

Three things to leave with

01

LLMs already write working exploits.

Post-cutoff, no memorization — and the curve doubles in months.

02

The leverage is where you point them.

Better rules beat the baseline 4×; a learned ranker is another ~5×.

03

It's all about lowering the cost of intelligence.

The margin between frontier closed models and everything else is the opportunity.