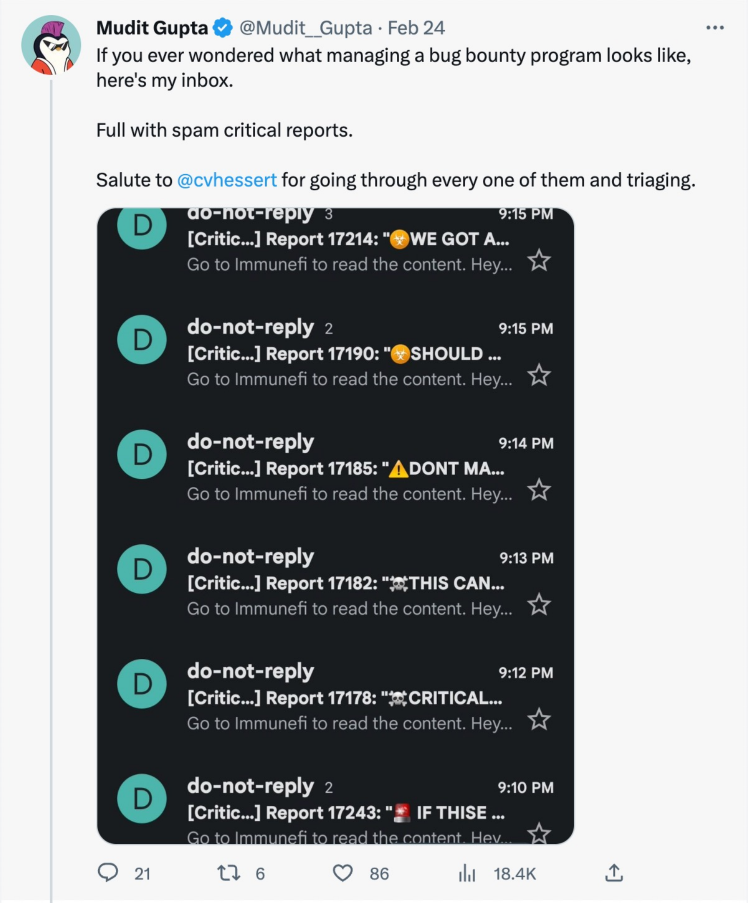
zkPoEX

Zero Knowledge Proof of Exploit

Bug Bounties and the Challenge of DeFi Security

- Despite the growth of DeFi, **security remains a critical concern**, with frequent hacks and vulnerabilities in smart contracts.
- **The lack of incentives** for security experts and the **difficulty of running effective bug bounty programs** exacerbate this problem.



Mudit Gupta @Mudit_Gupta · Feb 24

If you ever wondered what managing a bug bounty program looks like, here's my inbox.

Full with spam critical reports.

Salute to @cvhessert for going through every one of them and triaging.

do-not-reply 3 — 9:15 PM
[Critic...] Report 17214: "☣WE GOT A...
Go to Immunefi to read the content. Hey... ☆

do-not-reply 2 — 9:15 PM
[Critic...] Report 17190: "☣SHOULD ...
Go to Immunefi to read the content. Hey... ☆

do-not-reply — 9:14 PM
[Critic...] Report 17185: "⚠DONT MA...
Go to Immunefi to read the content. Hey... ☆

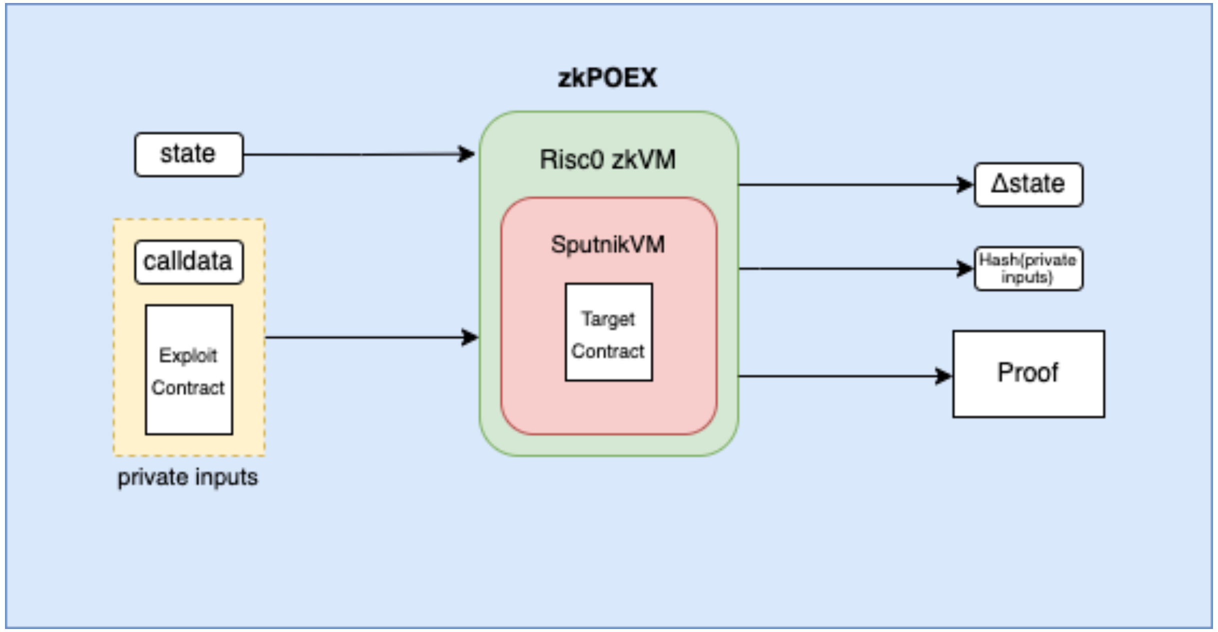
do-not-reply — 9:13 PM
[Critic...] Report 17182: "☠THIS CAN...
Go to Immunefi to read the content. Hey... ☆

do-not-reply — 9:12 PM
[Critic...] Report 17178: "☠CRITICAL...
Go to Immunefi to read the content. Hey... ☆

do-not-reply 2 — 9:10 PM
[Critic...] Report 17243: "🚨 IF THISE ...
Go to Immunefi to read the content. Hey... ☆

zkPoEX: Improving DeFi Vulnerability Reporting

- Security experts can use zkPoEX to **report vulnerabilities** anonymously and securely **in a provable manner**, and earn compensation for their efforts.
- **DeFi teams can identify vulnerabilities through confidential reporting and verification of proofs,** leading to an overall increase in the security of their projects



Scope

zkPoEX has been designed to be **exploit agnostic.** It can be configured to prove most common types of attacks. Some examples of this include:

- *Reentrancy*
- *Faulty logic*
- *Denial of service*
- *Contract ownership*
- *Etc.*

Vulnerability Submission Process

1. Auditor copies state of relevant contracts, configures zkPoEX to generate the proof and sends proof to the team
2. Team verifies proof, sends funds to the escrow/committee
3. Auditor reveals vulnerability
4. Team fixes the vulnerability
5. Escrow releases funds to auditor

Future Improvements

- **CLI-tool**: simplify the process for auditors
- **Scalability**: use bonsai network to verify the proof
- **Trustless bounty claim**:
  - *Early on: escrow with a comitee*
  - *Later: fully trustless on-chain approach*
- **Extend the approach** to be more modular across more difficult types of attacks: cross-chain, miner attacks, non-EVM, etc.
- **Any change of state is provable using this architecture**, exploits are only a tiny subset of what can be done.

Links

github.com/zkoranges/zkPoEX/

twitter.com/zkPoEX