zkSync

zkEVM Security

Porter Adams
Security Engineer
Matter Labs

Porter Adams

@portport255

Background

- Porter Adams
- Security Engineer at Matter Labs
- I like math

What is zkSync's purpose?

Our purpose is to break financial barriers and enhance the world's freedom — by accelerating the mass adoption of public blockchains.

There are many security topics:

- Cryptography
- Upgrades, Governance, and Security Council
- Bridging
- Trusted Execution
- Infrastructure
    - Secure SDLC
    - DevOps

zkEVM

zkEVM transaction lifecycle

1. Sequencer accepts transactions
2. VM executes transaction correctly
3. Prover proves a batch of transactions
4. L1 verifies the proof and updates the state root

L2s inherit the security of Ethereum

- How do we make it so users don’t need to trust the L2?
- Forced include transactions from the L1 (Sequencer Failure)
  - Adds censorship resistance
- Prove correct execution of the L2 (State Validation)
  - Either ZK proofs or Fraud proofs
- Permissionless withdrawals (Proposer Failure)
- See current status at: https://l2beat.com/scaling/risk

Forced transaction inclusion

- If the sequencer fails:
    - L2 nodes censor transactions
    - L2 nodes stop processing all transactions
- Solution:
    - Force inclusion through the L1
    - Allow anyone to be the sequencer

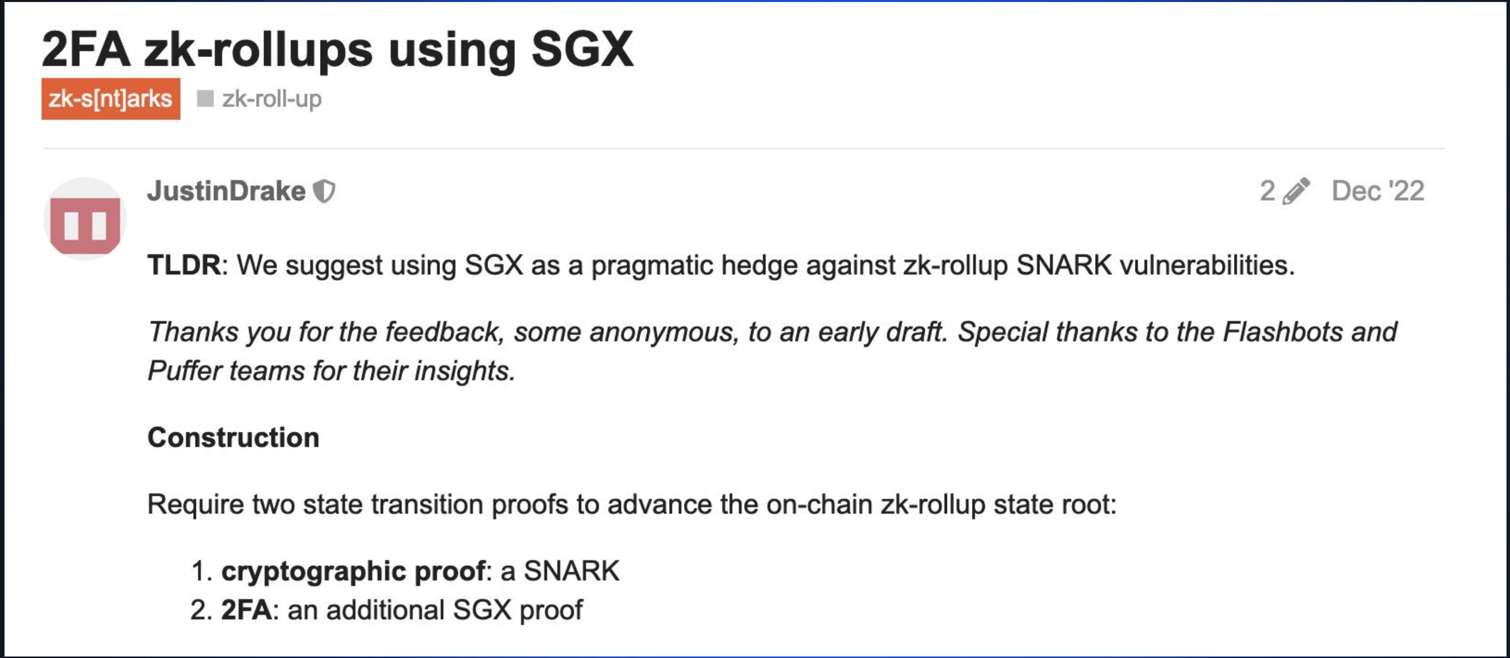
Proposer (state update) Failure

- If the proposer fails:
  - No more state updates on the L1 *from the L2*
- Solution
  - Allow anyone to propose state updates
  - Escape hatch
    - Users can exit directly via the L1

State Transition Failure

- If there is a bug in the ZK proof or fraud proof:
  - An invalid state transition can occur
- Solution
  - Do lot’s of security for your proof system
  - Have 2nd factor checks on the proofs
    - Trusted Execution Environments
    - Out-of-Circuit code
    - Multiple proof implementations in different languages

Trusted Execution Environments



2FA zk-rollups using SGX

zk-s[nt]arks zk-roll-up

JustinDrake 2 Dec '22

TLDR: We suggest using SGX as a pragmatic hedge against zk-rollup SNARK vulnerabilities.

Thanks you for the feedback, some anonymous, to an early draft. Special thanks to the Flashbots and Puffer teams for their insights.

Construction

Require two state transition proofs to advance the on-chain zk-rollup state root:

1. **cryptographic proof**: a SNARK
2. **2FA**: an additional SGX proof

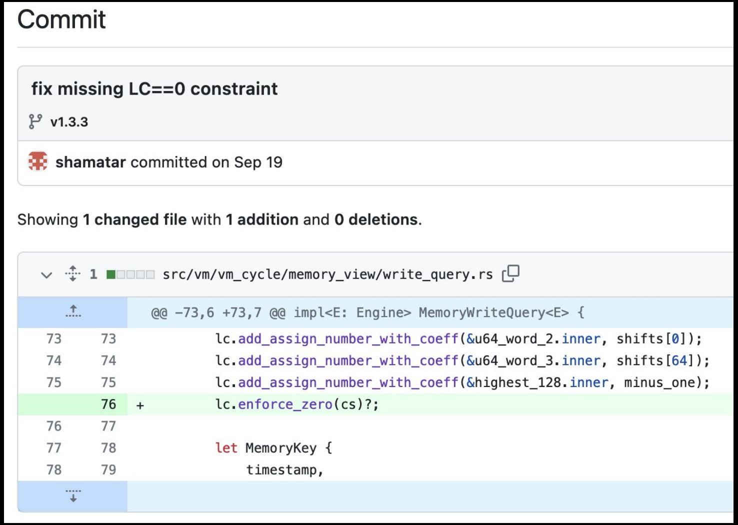
What has zkSync done to secure our ZK code?

- Traditional audits
  - Trail of Bits
  - Spearbit
  - Chainlight
- Audit competitions
  - Code4rena
- Bug Bounty

Why study ZK security?

$530m

TVL on zkSync Era
Dec 12, 2023


Commit

fix missing LC==0 constraint

v1.3.3

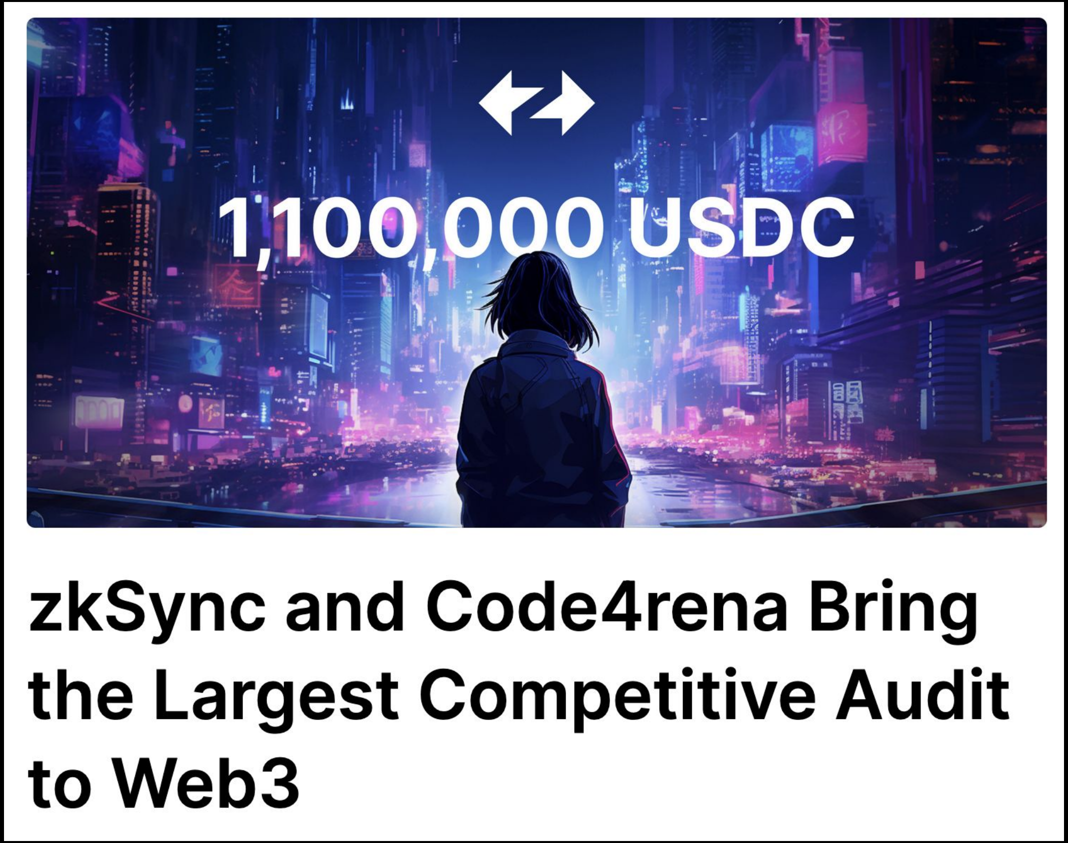
shamatar committed on Sep 19

Showing **1 changed file** with **1 addition** and **0 deletions**.

`src/vm/vm_cycle/memory_view/write_query.rs`

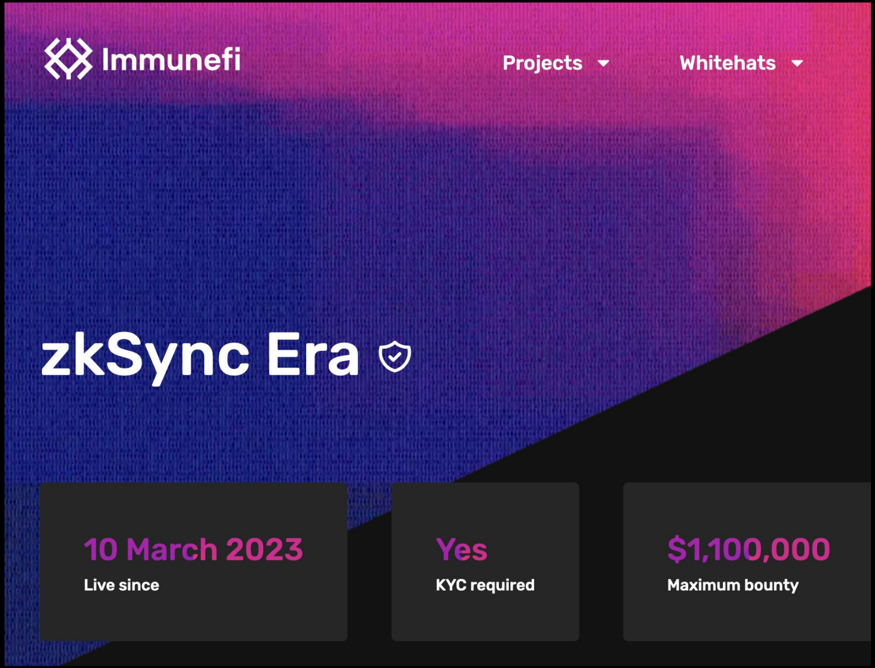
```
@@ -73,6 +73,7 @@ impl<E: Engine> MemoryWriteQuery<E> {
        lc.add_assign_number_with_coeff(&u64_word_2.inner, shifts[0]);
        lc.add_assign_number_with_coeff(&u64_word_3.inner, shifts[64]);
        lc.add_assign_number_with_coeff(&highest_128.inner, minus_one);
+       lc.enforce_zero(cs)?;

        let MemoryKey {
            timestamp,
```



zkSync and Code4rena Bring the Largest Competitive Audit to Web3

Bug Bounty



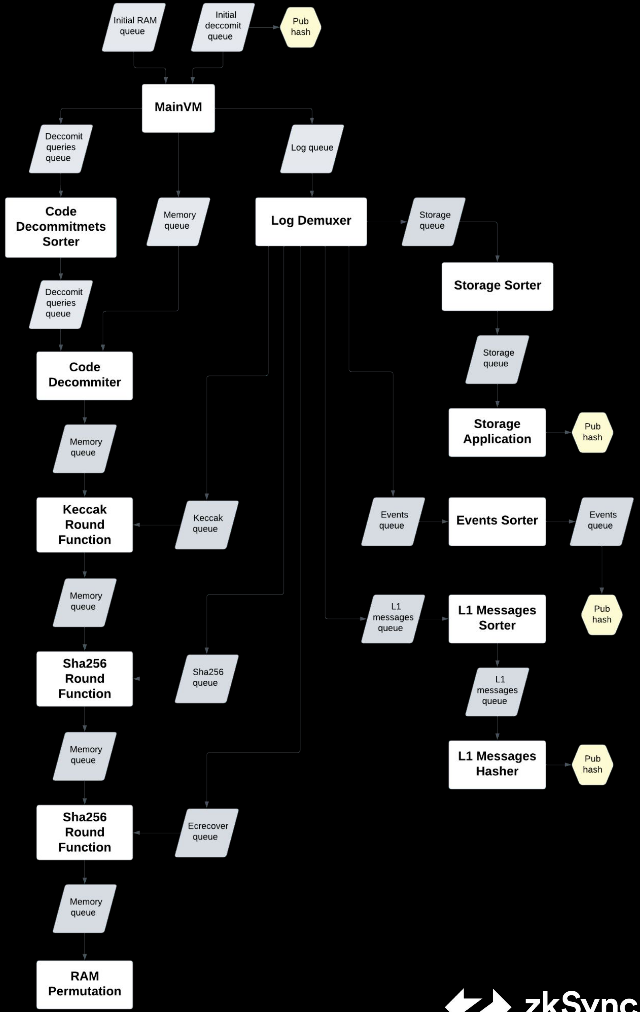
zkSync Era

10 March 2023	Yes	$1,100,000
Live since	KYC required	Maximum bounty

What do zkSync’s circuits look like?

Diagram of Boojum circuits

Intuition:
The proof shows that the Circuits have been computed correctly



Places to look for ZK bugs (1 of 3)

- Logic Bugs
  - Ex: Completely forgetting to prove something
- How the circuits fit together
  - Ex: the output of one circuit should be constrained to the input of the next circuit
- Individual circuits
  - Ex: Not checking memory read/writes correctly
- Proof system bugs
  - Ex: Breaking the soundness of FRI

Places to look for ZK bugs (2 of 3)

- Arithmetization bugs
  - Ex: Error in our PLONKish gates
- zkSNARK bugs
  - Ex: Mistake in the Fiat-Shamir implementation
- Cryptography bugs
  - Ex: Breaking the Poseidon Hash function
- Math bugs
  - Overflow

Places to look for ZK bugs (3 of 3)

- Governance bugs
    - Ex: If there is an emergency, can we upgrade the L1 verifier?
    - Ex: Who gets to upgrade the L1 verifier?
- SDLC
    - Ex: developer's github credentials are compromised
- Software Supply Chain
    - Ex: CVE found in a software dependency

Examples of ZK Bugs in practice

- https://github.com/0xPARC/zk-bug-tracker
- https://github.com/nullity00/zk-security-reviews

Walkthrough real ZK bugs

Future of ZK Security

Future of zkEVM Security

Thank you!

If you have more questions, message me on twitter: @portport255