# zkSync Insufficient Proof Verification Bugfix Review

Immunefi Editor

Follow

Immunefi

--

1

Listen

Share

## Summary

On October 5th 2023, security researcher LonelySloth responsibly disclosed a Critical vulnerability in zkSync Lite with multiple impacts, including unauthorized minting, freezing, and transferring of tokens, tampering with transactions, and preventing the recovery of the state tree by other users of the protocol.

Thanks to the responsible disclosure of the whitehat and fast response from the zkSync team, the vulnerability has been fixed. LonelySloth was awarded $200,000 for the finding. Let's dive into the technical details of the reported bug.

Note: This is not related to zkSync Era

## What is zkSync?

zkSync Lite is a scaling engine for Ethereum. Its current functionality scope includes low gas transfers of ETH and ERC20 tokens, atomic swaps, and limit orders while providing native L2 NFT support.

zkSync Lite is built on ZK Rollup architecture. ZK Rollup is an L2 scaling solution in which all funds are held by a smart contract on the mainchain, while computation and storage are performed off-chain. For every Rollup block, a state transition zero-knowledge proof (SNARK) is generated and verified by the mainchain contract. This SNARK includes the proof of the validity of every single transaction in the Rollup block. Additionally, the public data update for every block is published over the mainchain network as cheap calldata.

## Data packing

As stated in the project's description, zkSync utilizes a unique mechanism to optimize gas costs associated with public calldata. This involves a specialized format called the "packed floating point format." However, a critical aspect of this implementation has revealed a significant vulnerability, which could potentially be exploited for malicious actions, like token minting and transaction tampering.

- Tampering occurs with the transaction data during the packing and unpacking process, enabling a bypass of the proof generation system and resulting in the acceptance of altered transaction data.
- The primary issue concerns the parsing of floating point numbers during the allocation of bits computation to the mantissa variable. whitehat identified that it's possible to assign arbitrarily packed data to this variable, which can enable the generation of proof that the circuit's verification keys will accept. Consequently, a malicious user can generate a proof using arbitrary pairs of (packed_amount

, unpacked_amount

), potentially leading to transaction tampering.

## Packed Floating Point Format

In zkSync's architecture, amounts and fees for transactions such as transfers and swaps are serialized using the packed amount format. This format is integral for reducing gas costs. It necessitates an additional witness in the zkSync circuit — the unpacked amount corresponding to the packed amount. This setup is crucial for ensuring accurate calculations of balances and other operations without resorting to complex floating-point arithmetic.

## Implementation in Code:

The code for this system, particularly found in the franklin-crypto

library, reveals the process of decoding packed floating points into unpacked allocated numbers. A function, parse_with_exponent_le

, is central to this process. It takes a bit decomposition and repacks it into an AllocatedNum

, a type used in the circuit for representing numbers.

The code asserts that the length of bits equals the sum of exponent length and mantissa length. It then allocates numbers for various components, including the exponent and mantissa results, using the AllocatedNum::alloc

function. This function creates a new, private witness without imposing any constraints on the value of that witness. Because of that, anyone is able to run a prover with any arbitrary value as mantissa to generate a proof that can be verified using the same verification keys.

A malicious user can now generate a proof using arbitrary value, which leads to transaction manipulation.

The vulnerability fix was applied on the 7th of November, 2023 in thiscommit.

https://github.com/matter-labs/franklin-crypto/commit/0fd5c23fbce6ada6392599d3a4e8553fd4d70e99

- Where the mantissa is now being created with a new method into_allocated_num to enforce constraints and ensure that the value of the mantissa is derived from the expected calculations, thereby addressing the vulnerability.
- Removing unconstrained allocations and adding appropriate constraints are crucial steps in securing the circuit against the attacks described here, such as token minting, transaction tampering, and token theft.

# Acknowledgments

We extend our gratitude to the whitehat LonelySloth for responsibly disclosing such a crucial bug in a timely manner. Special recognition also goes to the zkSync team, who promptly responded to the report and patched the issue.

If you’re a developer or a whitehat considering a lucrative bug-hunting career in web3, this message is for you. With rewards 10–100x greater than those commonly found in web2, your efforts will pay off exponentially by transitioning to web3.

Check out the Web3 Security Library and start earning rewards on Immunefi — the leading bug bounty platform for web3 with the world’s biggest payouts.