

I-Own-Hype Audit

Audited by <https://github.com/nhtyy>.

Repo Overview

The **I-Own-Hype** repo creates a “proof of funds” based on a known & trusted snapshot of a Hyperliquid block. The block snapshot is converted to JSON, then a merkle tree is created from the top balances.

The inputs to the proof assume the merkle root is trusted, the script to create the tree is in the repo and anyone can verify the root is valid by running the script with the known snapshot.



The holder makes a claim that they possess private keys (via a signature) of some \$HYPE holders, the sum of the balances of those addresses is taken.

Assumptions

- The state snapshots **MUST BE** correct and valid for the given block.
- A proof **MUST BE** generated with PLONK or GROTH modes
 - Any other proof types are NOT ZK!
- The committed digest **MUST BE** checked to be the fixed message.

Findings

- commit: 8a8d8f39a510a533b22fa35e66004f5c50faeaeef

| Severity | ID | Description | Recommendation |
|----------------------------------------------------------------------------------------------|---------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  Critical | #1 Arbitrary message digest allows impersonation of any HYPE holder | As the message digest is passed in by the user as input to the create a proof, an attacker can find any message signed by a HYPE holder in the list (like a transaction), and submit the digest to the proof program | <p>Sign off on the hash of the leaf you're proving inclusion of concatenated with a nonce, reconstruct the hash in the program.</p> <p>Instead of committing to the message digest, just commit to the nonce to prevent replay attacks.</p> <p>ie. a consumer of the proof would accept a proof iff it has not seen the nonce before.</p> |
|  High | #2 Small balance range | Given the small range that is hardcoded into the program an attacker would just search the list of balances, a small range means a smaller anonymity pool | For future use, a minimum bound |

| | | | |
|---------------|-----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | can lead to doxxing | | <p>should be passed in as input, commit to public values <code>'total_balance >= min_balance'</code> and <code>'min_balance'</code></p> <p>But be careful to make sure you choose a bound that includes at least ~100 other holders!</p> |
| Informational | #3 Unnecessary hex encoding | The public key is hex-encoded then immediately decoded in the next function. | Just return a <code>Vec<u8></code> of the pubkey bytes, then hash that directly |
| Informational | #4 Alloy could be used to simplify code | <p>Instead of reimplementing Pubkey → address yourself, a well known crate called <code>Alloy</code> has this for you already</p> <p>https://docs.rs/alloy-primitives/latest/alloy_primitives/struct.Address.html#method.from_raw_public_key</p> | Use alloy to convert to an address, store this type in the hash set instead of a string |

Remediations

#1: The verifier program was amended to always check that the digest corresponds to a known and expected message, replay attacks were considered out of scope, the digest we care about actually commits to the users twitter account.

#2: Acknowledged, given the distribution of the balances, and that the user is supplying multiple signers, the search space is too large to be considered feasible.

#3 Fixed

#4 Fixed