ANASTASIA LABS

Security Audit Report

Business Confidential

Date: December 14, 2023
Project: Fluid Tokens Boosted Staking
Version 1.0

Contents

Confidentiality statement

Scope and Disclaimer

The scope of this audit is limited to the on-chain code specified in the files audited section. The code responsible for off-chain transaction building was not reviewed in this audit.

Aiken, the language that was used to write the Fluid Tokens Boosted Staking smart contracts is relatively new; as such, the UPLC that it generates may not correctly correspond to the intentions of the Aiken code. The correctness of the Aiken compilation pipeline and the Aiken standard library are both out of the scope of this audit.

Anastasia Labs prioritized identifying the weakest security vectors as well as prominent areas where code quality can be improved.

The scope of the audit did not include the creation of additional unit or property-based testing of the contracts.

The findings and recommendations contained herein reflect the information gathered by Anastasia Labs during the course of the assessment, and exclude any changes or modifications made outside of that period.

Assessment overview

From November 15th, 2023 to December 14th, 2023, FluidTokens FT Labs GmbH engaged Anastasia Labs to evaluate and conduct a security assessment of its Fluid Tokens Boosted Staking protocol. All code revision was performed following industry best practices.

Phases of code auditing activities include the following:

- Planning – Customer goals are gathered.
- Discovery – Perform code review to identify potential vulnerabilities, weak areas, and exploits.
- Attack – Confirm potential vulnerabilities through testing and perform additional discovery upon new access.
- Reporting – Document all found vulnerabilities.

The engineering team has also conducted a comprehensive review of protocol optimization strategies.

Assessment components

Manual revision

Our manual code auditing is focused on a wide range of attack vectors, including but not limited to.

- UTXO Value Size Spam (Token Dust Attack)
- Large Datum or Unbounded Protocol Datum
- EUTXO Concurrency DoS
- Unauthorized Data modification
- Multisig PK Attack
- Infinite Mint
- Incorrect Parameterized Scripts
- Other Redeemer
- Other Token Name
- Arbitrary UTXO Datum
- Unbounded protocol value
- Foreign UTXO tokens
- Double or Multiple satisfaction
- Locked Ada
- Locked non Ada values
- Missing UTXO authentication
- UTXO contention

Code base

Repository

https://github.com/FluidTokens/ft-audit-sc-renting-v3

Commit

739335d1f1580fcfaa453503466db8e3d858b755

Files audited

SHA256 Checksum	Files
01c3835e2342275eb69d379572be690eb8b961 ffd7e692ada8dc42660f0b5344	validators/boostedstaking.ak

Finding severity ratings

The following table defines levels of severity and score range that are used throughout the document to assess vulnerability and risk impact.

	Level	Severity	Findings
🟥	5	Critical	0
🟥	4	Major	1
🟧	3	Medium	1
🟨	2	Minor	0
🟦	1	Informational	1

Executive summary

The Fluid Tokens Boosted Staking Protocol is one of the most innovative protocols ever made on Cardano and it allows tokens holders to rent their assets to anyone that needs them.

This is extremely useful for different NFT types:

- Utility tokens: receiving airdrops and other benefits by holding a specific token
- Metaverse assets: renting land plots, skins and assets of your favourite games
- NFT memberships: NFTs that allow the access to exclusive events and communities
- Real world assets: rent real world objects such as apartments, cars, certificates and more

Security Claims

- Owners (aka renters) of any NFT can create a renting offer, locking the NFT in the FluidShare smart contract and deciding the parameters of the renting: * Daily Rent Amount: how much each day of renting costs * Expiration Offer: the time limit after which the NFT cannot be rented anymore
- Once the renting offer is available on the platform, any user (aka tenant) can accept the offer, decide how many days to rent it (within the limit of the Expiration Offer) and pay upfront the decided amount.
- At this point, thanks to Cardano unique features, the NFT appears in the tenant wallet and it will stay there until the renter will withdraw it or another tenant will rent it. In other words it is guaranteed that the NFT will be in the tenant wallet for the whole renting period.
- Renters can always edit the parameters of their renting offers, even when the renting is in progress. The effects will be visible on the next rent.
- If the tenant needs the NFT for more additional days, he can extend the renting period - paying the new amount to the renter - up to the Expiration Offer limit.

Findings

ID-401 Missing pool value check

	Level	Severity	Status
🟥	4	Major	Resolved

Description

The boostedstaking.ak validator script has a vulnerability that allows attackers to create fake pools and locking zero assets into this contract with a duplicated valid datum (from a valid pool). The malicious actor can use these fake pools to steal funds by renting from these fake pools, so that the tx builder is going to take the assets from either the transaction batcher or whoever is building the transaction.

The following code is the source of the exploit.

```
fn validate_activation_datums_and_values(
   datum: Datum,
   transaction: Transaction,
   tenant: Address,
   days: Int,
   index: Int,
   wanted: Int,
   contract: ByteArray,
) {
   let leftover = datum.pool_amount - wanted
```

Recommendation

The pool amount should be calculated from the values of the input transaction, by reading out the quantity of assets stored in the output value of the input transaction and checked against the pool amount stored in the datum.

Resolution

Resolved

ID-301 Missing polarity checks for Integer values

	Level	Severity	Status
🟨	3	Medium	Resolved

Description

The boostedstaking.ak validator script does not have sufficient checks to prevent sideeffects by using negative Integer values.

Recommendation

Add checks if following Integer values are greater than/or zero

- daily_rent_amount
- pool_amount
- pool_divider
- deadline_date
- expiration_offer
- fee_percentage
- min_days
- multiplier

Resolution

Resolved

ID-101 Validation of conditions are replicated

	Level	Severity	Status
🟦	1	Informational	Resolved

Description

There're some places where the validation of some conditions are replicated:

```
active_output.address == active_rent_address,
active_output_pool.address == pool_rent_address,
begin > datum.deadline_date,
active_datum == parsed_datum,
pool_datum == parsed_datum_pool,
end - begin <= 3600000,
active_datum.deadline_date < active_datum.expiration_offer,
```

Same for the one below:

```
validate_index_redemeer(
ctx.transaction.inputs,
current_contract.hash,
index,
current_contract.current_validation,
)
```

Recommendation

It's recommended to move these code lines one level up so that the script size can be lowered.

Resolution

Resolved