Three Sigma Labs

Code Audit

Blast Off Blast Based Yield Aggregator and Launchpad

Disclaimer

Code Audit

Blast Off Blast Based Yield Aggregator and Launchpad

Disclaimer

The ensuing audit offers no assertions or assurances about the code's security. It cannot be deemed an adequate judgment of the contract's correctness on its own. The authors of this audit present it solely as an informational exercise, reporting the thorough research involved in the secure development of the intended contracts, and make no material claims or guarantees regarding the contract's post-deployment operation. The authors of this report disclaim all liability for all kinds of potential consequences of the contract's deployment or use. Due to the possibility of human error occurring during the code's manual review process, we advise the client team to commission several independent audits in addition to a public bug bounty program.

Table of Contents

Code Audit

Blast Off Blast Based Yield Aggregator and Launchpad

Table of Contents

Summary

Code Audit

Blast Off Blast Based Yield Aggregator and Launchpad

Summary

Three Sigma Labs audited Blast Off in a 1 day engagement. The audit was conducted on 11-06-2024.

Protocol Description

A native-yield-based launchpad and yield aggregator, unlocking a new world of idle earnings.

Blastoff StakingContract allows staking OFF tokens for rewards.

Scope

Code Audit

Blast Off Blast Based Yield Aggregator and Launchpad

Scope

StakingContract.sol

Assumptions

OpenZeppelin is secure.

Methodology

Code Audit

Blast Off Blast Based Yield Aggregator and Launchpad

Methodology

To begin, we reasoned meticulously about the contract's business logic, checking security-critical features to ensure that there were no gaps in the business logic and/or inconsistencies between the aforementioned logic and the implementation. Second, we thoroughly examined the code for known security flaws and attack vectors. Finally, we discussed the most catastrophic situations with the team and reasoned backwards to ensure they are not reachable in any unintentional form.

Taxonomy

In this audit we report our findings using as a guideline Immunefi's vulnerability taxonomy, which can be found at immunefi.com/severity-updated/. The final classification takes into account the severity, according to the previous link, and likelihood of the exploit. The following table summarizes the general expected classification according to severity and likelihood; however, each issue will be evaluated on a case-by-case basis and may not strictly follow it.

Severity / Likelihood	LOW	MEDIUM	HIGH
NONE	None		
LOW	Low		
MEDIUM	Low	Medium	Medium
HIGH	Medium	High	High
CRITICAL	High	Critical	Critical

Project Dashboard

Code Audit

Blast Off Blast Based Yield Aggregator and Launchpad

Project Dashboard

Application Summary

Name	Blast Off
Commit	757024c2c17a76d0a073e1e8bd2cf1d68959ad81
Language	Solidity
Platform	Ethereum

Engagement Summary

Timeline	11-06-2024
№ of Auditors	2
Review Time	1 day

Vulnerability Summary

Issue Classification	Found	Addressed	Acknowledged
Critical	0	0	0
High	1	1	0
Medium	0	0	0
Low	0	0	0

None	1	0	1

Category Breakdown

Suggestion	0
Documentation	0
Bug	2
Optimization	0
Good Code Practices	0

Code Maturity Evaluation

Code Audit

Blast Off Blast Based Yield Aggregator and Launchpad

Code Maturity Evaluation

Code Maturity Evaluation Guidelines

Category	Evaluation
Access Controls	The use of robust access controls to handle identification and authorization and to ensure safe interactions with the system.
Arithmetic	The proper use of mathematical operations and semantics.
Centralization	The presence of a decentralized governance structure for mitigating insider threats and managing risks posed by contract upgrades
Code Stability	The extent to which the code was altered during the audit.
Upgradeability	The presence of parameterizations of the system that allow modifications after deployment.
Function Composition	The functions are generally small and have clear purposes.
Front-Running	The system's resistance to front-running attacks.
Monitoring	All operations that change the state of the system emit events, making it simple to monitor the state of the system. These events need to be correctly emitted.
Specification	The presence of comprehensive and readable codebase documentation.
Testing and Verification	The presence of robust testing procedures (e.g., unit tests, integration tests, and verification methods) and sufficient test coverage.

Code Maturity Evaluation Results

Category	Evaluation
Access Controls	Satisfactory. All access control is correctly implemented.
Arithmetic	Moderate. Some rounding errors were found.
Centralization	Satisfactory. No significant points of centralization are found.
Code Stability	Satisfactory. The code was stable throughout the audit.
Upgradeability	Weak. The contracts are not upgradeable.
Function Composition	Satisfactory. The code was correctly split into helper functions.
Front-Running	Satisfactory. No front-running issues are present.
Monitoring	Satisfactory. Most events are emitted.
Specification	Satisfactory. The code follows the specifications.
Testing and Verification	Satisfactory. The codebase implements unit and fuzz tests.

Findings

Code Audit

Blast Off Blast Based Yield Aggregator and Launchpad

Findings

3S-BlastOff-H01

Staking 30 days or less before **`emissionEnd`** will lead to locked funds without rewards for some time

Id	3S-BlastOff-H01
Classification	High
Severity	High
Likelihood	High
Category	Bug
Status	Addressed in #a0f41f9.

Description

Rewards are emitted until **`emissionEnd`**. Whenever users stake, they have to wait **`30`** days to be able to unstake. Thus, staking close to **`emissionEnd`** will leave the funds locked without receiving rewards.

Recommendation

Either let users unstake if **`emissionEnd`** is reached or do not let users stake **`30`** days prior to **`emissionEnd`**.

3S-BlastOff-N01

Users can game **`minStakeTime`** by staking 1 wei, waiting 30 **`days`** and then staking a real amount

Id	3S-BlastOff-N01
Classification	None
Category	Bug
Status	Acknowledged

Description

The staking contract requires **`30`** days to pass since the first stake for each wallet to be able to unstake. Thus, users can stake 1 wei for **`30`** days, and only then stake the real amount. This means they can effectively ignore the staking time and allocate the funds elsewhere for the first 30 days.

Recommendation

Technically users can trick the **`minStakingTime`**, but it means they will not also receive rewards during the first **`30`** days, so they are still taking a loss. Thus, even if users decide to do this, other users profit from this as they get more rewards in the first **`30`** days. It would make more sense to have to wait **`30`** days for each staked amount, but this would increase complexity, and the current design works well either way.