SB SECURITY

Helios Staking V2
Security Review

Jan 3, 2025

Conducted by:
Blckhv, Lead Security Researcher
Slavcheww, Lead Security Researcher

Contents

1. About SBSecurity

SBSecurity is a duo of skilled smart contract security researchers. Based on the audits conducted and numerous vulnerabilities reported, we strive to provide the absolute best security service and client satisfaction. While it's understood that 100% security and bug-free code cannot be guaranteed by anyone, we are committed to giving our utmost to provide the best possible outcome for you and your product.

Book a Security Review with us at sbsecurity.net or reach out on Twitter @Slavcheww.

2. Disclaimer

A smart contract security review can only show the presence of vulnerabilities **but not their absence**. Audits are a time, resource, and expertise-bound effort where skilled technicians evaluate the codebase and their dependencies using various techniques to find as many flaws as possible and suggest security-related improvements. We as a company stand behind our brand and the level of service that is provided but also recommend subsequent security reviews, on-chain monitoring, and high whitehat incentivization.

3. Risk classification

	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

3.1. Impact

- **High** - leads to a significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - leads to a moderate loss of assets in the protocol or some disruption of the protocol's functionality.
- **Low** - funds are not at risk.

3.2. Likelihood

- **High** - almost **certain** to happen, easy to perform, or highly incentivized.
- **Medium** - only **conditionally possible**, but still relatively likely.
- **Low** - requires specific state or **little-to-no incentive**.

3.3. Action required for severity levels

- High - **Must** fix (before deployment if not already deployed).
- Medium - **Should** fix.
- Low - **Could** fix.

4. Executive Summary

Helios Staking V2 will award all Helios stakers with rewards, based on their stakes sizes.

Overview

Project	Helios Staking V2
Repository	Private
Commit Hash	9ef1d14722fdf860a1420dc68bc58724d0dbe8b2
Resolution	c3f8dff9d7ed2a85adc85463cc022ea57aa69d47
Timeline	January 2, 2025

Scope

HeliosStakingV2.sol

Issues Found

Critical Risk	1
High Risk	0
Medium Risk	1
Low/Info Risk	4

5. Findings

5.1. Critical severity

5.1.1. Users can updateCycle and stake in same second stealing rewards from other users

Severity: Critical Risk

Description: If `updateCycle` is called, followed by a Helios stake, after allocation has been calculated, both in the same `block.timestamp`, part of the rewards of other users will be stolen. This is possible because the following check is missing strict equality and accepts stakes that have happened in the same timestamp.

```
if (stake.stakeStartTs > _lastCycleT) revert StakeIneligible();
```

Recommendation: To fix the issue, strict equality should be added, this will allow only stakes that have happened before `updateCycle` has been called. Fix should be applied to both `claimRewards` and `getUserRewards`

Resolution: Fixed

5.2. Medium severity

5.2.1. To-be blacklisted address can grief the rewards of the other users

Severity: Medium Risk

Description: When an owner decides to blacklist a certain user, he can grief all the other stakers by frontrunning the `setBlacklisted` function with a call to `updateCycle`. Since he is still not blacklisted, his hlx shares will be 'active' and will be included in the cycle allocation. This will dilute the rewards of the other users for an entire cycle.

Note that this attack is only possible when a new cycle has begun in terms of days passed.

Recommendation: Make sure to update the cycle before blacklisting addresses.

Resolution: Acknowledged

5.3. Low/Info severity

5.3.1. `getUserRewards` is missing blacklist checks

Severity: Low Risk

Description: `getUserRewards` called for blacklisted accounts will return non-zero rewards, no matter if they are not eligible for them, because there is a missing check on whether they are contained in the `isBlacklisted` mapping.

Recommendation: Add a check to return 0 rewards and false as eligible if the account is blacklisted.

Resolution: Fixed

5.3.2. Global allocation instead of per cycle can cause rewards to fluctuate

Severity: Low Risk

Description: Since the allocation is recalculated each cycle, but the balance is not reserved, users with the same hlx shares can receive different rewards in different cycles. They will receive more if the balance is higher than in the previous or if global shares have decreased and less in the opposite cases.

Recommendation: Calculate the `allocation` per cycle and introduce the `getCycleRewards` function that will return the rewards per cycle based on the reserved and paid amounts.

Resolution: Acknowledged

5.3.3. `minCyclePool` should be based on token

Severity: Low Risk

Description: Since various ERC20 tokens will be used, `minCyclePool` will be ineffective when all the reward tokens have different prices. For example, if there are two tokens, one with a price of $1 and the other with a price of $0.00005, adjusting that variable will be crucial in order not to allocate cents as rewards.

Recommendation: Add mapping per token and setter function for it.

Resolution: Fixed

5.3.4. `getUserRewards` should be external

Severity: Informational Risk

Description: `getUserRewards` should be external instead of public since it's not caller in the contract.

Recommendation: Make `getUserRewards` external.

Resolution: Acknowledged