SB SECURITY

Lotus

Security Review

Oct 19, 2024

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

Lotus is a DeFi token built on top of TitanX and Volt, utilizing a virtual mining model to power its ecosystem. The protocol offers staking options, allowing users to stake Lotus for durations ranging from 40 to 1480 days, earning rewards in TitanX.

Overview

Project	Lotus
Repository	Private
Commit Hash	00ca0cdc16fba80df71214ff010d472d389bffdd
Resolution	a0cb7a5e56e08a52087276ebc1094d378d710a95
Timeline	Audit: October 16 - October 19, 2024

Scope

contracts/*

Issues Found

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

5. Findings

5.1. Low/Info severity

5.1.1. `LotusBloom` request is missing `requestConfirmations` setter

Severity: Low Risk

Description: `requestRandomWords.requestConfirmations` should not be hardcoded because there is a `minimumRequestConfirmations` that can be set greater than `3` and the protocol will lock forever.

```
requestId = s_vrfCoordinator.requestRandomWords(
    VRFV2PlusClient.RandomWordsRequest({
        keyHash: keyHash,
        subId: subscriptionId,
        requestConfirmations: 3,  <--------------------------
        callbackGasLimit: 250_000,
        numWords: 1,
        extraArgs: VRFV2PlusClient._argsToBytes(VRFV2PlusClient.ExtraArgsV1({nativePayment: false}))
    })
);
```

Inside `VRF_COORDINATOR.requestRandomWords` there is a check ensuring that the `requestConfirmations` passed is not lower than the `minimumRequestConfirmations`. Since this value is configured at the `VRF_COORDINATOR` level, when this value becomes greater than `3` (`requestConfirmations`), `pickWinner()` will revert.

Recommendation: Add function to allow changing `requestConfirmations`.

Resolution: Fixed

5.1.2. `compoundRewards` is missing proper slippage check

Severity: Low Risk

Description: `Staking::compoundRewards` has no effective slippage checks and gives the ability to users with more capital to move the price of `Volt/Lotus` by decreasing the `Lotus` price. Although flash loan can't be used since we need multi block MEV, i.e. swapping in multiple blocks, which is achievable, given the relatively small liquidity that these pools hold after launch. By doing that users will receive inflated amount `Lotus` which will take greater percentage of the rewards.

Recommendation: You can hardcode `0` instead of passing `_amountLotusMin` and rely on the 15 min TWAP lookback.

Resolution: Acknowledged

5.1.3. `LotusBloom` participation is wrong compared to docs

Severity: Low Risk

Description: According to the docs, in order to participate in the `LotusBloom` one should stake 150,000 `Lotus` tokens, but in the deployment scripts we have `_minSharesToBloom` = 10, which, according to the initial Lotus price will be $420 less than intended.

Recommendation: In `DeployLotus.s.sol` change the `_minSharesToBloom` argument to be 150000 Lotus.

Resolution: Fixed

5.1.4. Informational issues

Severity: Informational Risk

Description:

- in `Staking::compoundRewards` change `_owneOfPosition` to `_ownerOfPosition`

Resolution: Fixed