Surge Security Audit

Report Version 0.1

January 23, 2025

Conducted by **Hunter Security**

Table of Contents

1 About Hunter Security

Hunter Security is an industry-leading smart contract security company. Having conducted over 100 security audits protecting over $1B of TVL, our team delivers top-notch security services to the best DeFi protocols. For security audit inquiries, you can reach out on Telegram or Twitter at *@georgehntr*.

2 Disclaimer

Audits are a time-, resource-, and expertise-bound effort where trained experts evaluate smart contracts using a combination of automated and manual techniques to identify as many vulnerabilities as possible. Audits can reveal the presence of vulnerabilities, but cannot guarantee their absence.

3 Risk classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	High	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** - involves a small loss of funds or affects a core functionality of the protocol.
- **Low** - encompasses any unexpected behavior that is non-critical.

3.2 Likelihood

- **High** - a direct attack vector; the cost is relatively low compared to the potential loss of funds.
- **Medium** - only a conditionally incentivized attack vector, with a moderate likelihood.
- **Low** - involves too many or unlikely assumptions; offers little to no incentive.

3.3 Actions required by severity level

- **High** - client **must** fix the issue.
- **Medium** - client **should** fix the issue.
- **Low** - client **could** fix the issue.

4 Executive summary

Overview

Project Name	Surge
Repository	-
Commit hash	-
Resolution	-
Methods	Manual review & testing

Scope

src/BonusMath.sol
src/PositionManager.sol
src/RewardMath.sol
src/StakingVault.sol
src/UniswapHelper.sol

Issues Found

High risk	0
Medium risk	0
Low risk	1
Informational	16

5 Findings

5.1 Low

5.1.1 userShares may return outdated information

Severity: Low

Files: src/StakingVault.sol

Description: In *StakingVault.userShares* multiple checks are implemented to ensure the passed cycle is currently active. The problem is that if we've entered a new cycle, but *endCycleIfNeeded* has not been called yet, *StakingVault.userShares* will return 0 for the actual current cycle as the *rewardPoolById(poolId).currentCycleId* has not been updated yet.

Recommendation: Consider using *_poolCalculateCurrentCycleId* instead of *currentCycleId*.

Resolution: Pending.

5.2 Informational

5.2.1 Typographical mistakes, non-critical issues or centralization vulnerabilities

Severity: Informational

Files: src/*

Description: The contracts contain one or more typographical mistakes, non-critical issues or centralization vulnerabilities. In an effort to keep the report size reasonable, we enumerate these below:

1. There is no need to apply a *nonReentrant* modifier to *depositPosition* as it is permissioned.
2. The admin should not be able to call *setFeeAmountTickSpacing* for fee levels that have already been set.
3. Adding liquidity should only happen via private RPC in order to protect from MEV attacks.
4. Consider using the *forceApprove* and *safeTransfer*/*safeTransferFrom* methods from OpenZeppelin's *SafeERC20* library in order to support non-standard ERC20 tokens such as USDT.
5. Consider granting approval of just the needed amount of tokens rather than *type(uint256).max* to reduce attack surface.
6. Using 100 as the *BONUS_DENOMINATOR* and *_roundToNearest* makes the bonus multiplier be the same over the course of multiple consecutive days. Consider whether this is the intended behavior.
7. There is no need to pass *Math.Rounding.Floor* when using *mulDiv* as this is the default option.
8. Consider casting one of the multipliers to *uint256* just for additional safety on the following line: *uint256 cycleDuration = poolInfo.cycleDurationDays * 1 days;*
9. Use *.transfer*, instead of *.transferFrom(address(this)* as some tokens may require allowance even when using *transferFrom* with own address.
10. The *ClaimableReward.poolId* property is never used.
11. No need to cast *address(router)*.
12. Forgotten *console.sol* import statement.
13. *nftManager*, *voltToken* and *stakingVault* could be marked *immutable*.
14. *_getNftUniV3* needs to return just *token0*, *token1*, *tickLower*, *tickUpper* and *fee*.

15. Repeated line: *_setAllowanceMaxIfNeeded(tokenIn, amountIn, config.routerAddress);*
16. Redundant check: *if (poolInfo.cycleDurationDays == 0) revert StakingVaultInvalidCycleDuration();*

Recommendation: Consider fixing the above typographical mistakes, non-critical issues or centralization vulnerabilities.

Resolution: Pending.