Soulmate Audit Report

Version 1.0

Keyword

March 1, 2024

Soulmate Audit Report

xKeywordx

February 11, 2024

Prepared by: xKeywordx

Table of Contents

Protocol Summary

Valentine’s day is approaching, and with that, it’s time to meet your soulmate!

We’ve created the Soulmate protocol, where you can mint your shared Soulbound NFT with an unknown person, and get `LoveToken` as a reward for staying with your soulmate. A staking contract is available to collect more love. Because if you give love, you receive more love.

Disclaimer

I make all effort to find as many vulnerabilities in the code in the given time period, but hold no responsibilities for the findings provided in this document. A security audit is not an endorsement of the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the Solidity implementation of the contracts.

Risk Classification

		Impact		
		High	Medium	Low
	High	H	H/M	M
Likelihood	Medium	H/M	M	M/L
	Low	M	M/L	L

I use the CodeHawks severity matrix to determine severity. See the documentation for more details.

Scope

```
-- interfaces
|   -- ILoveToken.sol
|   -- ISoulmate.sol
|   -- IStaking.sol
|   -- Ivault.sol
-- protocol
|   -- Airdrop.sol
|   -- LoveToken.sol
|   -- Soulmate.sol
|   -- Staking.sol
|   -- Vault.sol
```

Roles

None

Executive Summary

Known issues

- Eventually, the counter used to give ids will reach the type(uint256).max and no more will be able to be minted. This is known and can be ignored.

Issues found

Severity	Number of issues found
High	1
Medium	0
Low	0
Gas	0
Info	0
Total	1

Findings

Highs

[H-1] `Vault.sol` can be drained because of flawed logic in `Staking.sol::claimRewards` function.

Description: The `Staking.sol::claimRewards` function checks the amount that a user is eligible to claim based on the amount deposited, multiplied by the number of weeks the user has staked for, but without checking if the user actually kept the full amount deposited the whole time. A user can stake just 1 token in order to "start the timer" and then they can deposit a greater amount right before withdrawing in order to manipulate the rewards.

Impact: All the funds from the vault could be drained.

Proof of concept:

1. Imagine a pool that has 1999 tokens.
2. Attacker deposits 1 token.

3. Two weeks pass.
4. Attacker takes a flash loan and buys 2000 tokens.
5. Attacker deposits the funds.
6. Now the pool holds 4000 tokens, and the attacker is eligible to claim 2001 (total tokens deposited) * 2. Two because the time elapsed since the initial deposit is 2 weeks, even if he only deposited 1 single token back then.
7. Attacker calls `claimRewards` function, and drains the vault before returning the flash loan.
8. Pool has 0 tokens now, and the attacker has 2000 tokens after returning the flash loan.

Recommended mitigation: A potential mitigation strategy involves tracking the amount staked and the time it was staked for, ensuring rewards are calculated based on “staked time” rather than only looking at the current deposited amount. This involves rethinking the logic of the function