Soulmate Audit Report

Version 1.0

César Escribano

Soulmate Audit Report

César Escribano

February 14, 2024

Prepared by: César Escribano (@ceseshi)

Table of Contents

Protocol Summary

This is a security review of First Flight #9: Soulmate, a public contest from CodeHawks.

The Soulmate protocol allows participants to mint a shared Soulbound NFT with another person and earn “LoveToken” rewards as their relationship matures. The protocol includes functions for minting soulmate tokens, finding a soulmate, writing messages, and even getting divorced, which cancels LoveToken collection opportunities.

Disclaimer

The auditor makes all effort to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the findings provided in this document. A security audit by the auditor 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

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

Audit Details

Repository

https://github.com/Cyfrin/2024-02-soulmate

Commit Hash

```
b3f9227942ffd5c443ce6bccaa980fea0304c38f
```

Scope

```
#-- Airdrop.sol
#-- LoveToken.sol
#-- Soulmate.sol
#-- Staking.sol
#-- Vault.sol
```

Roles

Executive Summary

This audit took place in February 2024, over 4 days, totalling 16 hours. The tools used were Visual Studio Code and Foundry.

Issues found

Severity	Number of issues
High	4
Medium	2
Low	1
Info	0
Total	7

Findings

High

[H-01] Divorced users can claim staking rewards

Relevant GitHub Links https://github.com/Cyfrin/2024-02-soulmate/blob/b3f9227942ffd5c443ce6bccaa980fea0304c

Summary The Staking contract allows a divorced users to claim staking rewards.

Vulnerability Details The claimRewards() function in Staking.sol does not check if a user is divorced, so he can claim rewards, which is incorrect

```
function testClaimRewardsDivorced() public {

        _mintOneTokenForBothSoulmates();

        // Deposit and claim
        vm.startPrank(soulmate1);
        vm.warp(block.timestamp + 10 days);
        airdropContract.claim();
        uint256 initialBalance = loveToken.balanceOf(soulmate1);
        loveToken.approve(address(stakingContract), initialBalance);
        stakingContract.deposit(loveToken.balanceOf(soulmate1));

        // Get divorced
        soulmateContract.getDivorced();

        vm.warp(block.timestamp + 10 days);
        initialBalance = loveToken.balanceOf(soulmate1);
        stakingContract.claimRewards();

        // Compare balances
        console2.log("initialBalance", initialBalance);
        console2.log("balance", loveToken.balanceOf(soulmate1));
    }
```

Impact Divorced users can steal tokens from the staking vault.

Tools Used Foundry, Manual review

Recommendations Add a divorced check in Staking:claimRewards()

```
+error Staking__CoupleIsDivorced();

function claimRewards() public {
+   if (soulmateContract.isDivorced(msg.sender)) revert
    Staking__CoupleIsDivorced();
    uint256 soulmateId = soulmateContract.ownerToId(msg.sender);
    // first claim
    if (lastClaim[msg.sender] == 0) {
        lastClaim[msg.sender] = soulmateContract.idToCreationTimestamp(
            soulmateId
        );
    }
```

[H-02] Users without a soulmate can claim a large amount of staking rewards

Relevant GitHub Links https://github.com/Cyfrin/2024-02-soulmate/blob/b3f9227942ffd5c443ce6bccaa980fea0304c

Summary The Staking contract allows a user with love tokens but no soulmate, to deposit to stake and claim a large amount of staking rewards.

Vulnerability Details The claimRewards() function in Staking.sol does not check if a user has a soulmate, so the calculation takes a wrong number of weeks and sends and incorrect amount of tokens to the user.

This test claims rewards without a soulmate.

```
function testClaimRewardsWithoutSoulmate() public {
    // Jump to an current timestamp in mainnet
    uint256 timestamp = 1707775379;
    vm.warp(timestamp);

    // Get a new user with tokens
    address alice = makeAddr("alice");
    uint256 initialBalance = 100 ether;
    deal(address(loveToken), alice, initialBalance);

    // Deposit and claim
    vm.startPrank(alice);
    loveToken.approve(address(stakingContract), initialBalance);
    stakingContract.deposit(initialBalance);
    stakingContract.claimRewards();

    // Compare balances
    console2.log("initialBalance", initialBalance);
```

```
19      console2.log("balance", loveToken.balanceOf(alice));
20  }
```

The test shows the tokens claimed by alice.

```
1  Running 1 test for test/unit/AuditTest1.t.sol:AuditTest1
2  [PASS] testClaimRewardsWithoutSoulmate() (gas: 268174)
3  Logs:
4    initialBalance 10000000000000000000
5    balance     28230000000000000000000
```

Impact Users can steal tokens from the staking vault.

Tools Used Foundry, Manual review

Recommendations Add a soulmate check in Staking:claimRewards()

```
1  +error Staking__NoSoulmate();
2
3  function claimRewards() public {
4  +   if (soulmateContract.soulmateOf(msg.sender) == address(0)) revert
       Staking__NoSoulmate();
5      uint256 soulmateId = soulmateContract.ownerToId(msg.sender);
6      // first claim
7      if (lastClaim[msg.sender] == 0) {
8          lastClaim[msg.sender] = soulmateContract.idToCreationTimestamp(
9              soulmateId
10         );
11     }
```

[H-03] Divorced soulmates can claim airdrop tokens

Relevant GitHub Links https://github.com/Cyfrin/2024-02-soulmate/blob/b3f9227942ffd5c443ce6bccaa980fea0304c

Summary The Airdrop contract allows users who already divorced to claim tokens, which should not be possible.

Vulnerability Details The soulmateContract.isDivorced() function is incorrectly implemented, in a way that always returns false, so the claim() function allows to continue receiving love tokens after divorce.

isDivorced() should receive the address of the user to check, but instead it gets msg.sender, which is the address of the Airdrop contract.

This test calls claim() after divorce.

```
function testClaimAfterGetDivorced() public {
    // Get a soulmate
    _mintOneTokenForBothSoulmates();

    // Skip 100 days
    uint256 _days = 100;
    vm.warp(block.timestamp + (_days * 1 days));

    // Get divorced
    vm.startPrank(soulmate1);
    soulmateContract.getDivorced();

    // Should not be able to claim
    airdropContract.claim();

    console2.log("balance", loveToken.balanceOf(soulmate1));
}
```

The test shows the tokens sent to alice.

```
Running 1 test for test/unit/AuditTest1.t.sol:AuditTest1
[PASS] testClaimAfterGetDivorced() (gas: 333280)
Logs:
  balance 10000000000000000000
```

Impact Divorced soulmates can steal tokens from the airdrop vault.

Tools Used Foundry, Manual review

Recommendations Fix isDivorced() call in Airdrop:claim()

```
function claim() public {
    // No LoveToken for people who don't love their soulmates anymore.
-   if (soulmateContract.isDivorced()) revert Airdrop__CoupleIsDivorced
    ();
+   if (soulmateContract.isDivorced(msg.sender)) revert
    Airdrop__CoupleIsDivorced();

    // Calculating since how long soulmates are reunited
    uint256 numberOfDaysInCouple = (block.timestamp -
        soulmateContract.idToCreationTimestamp(
            soulmateContract.ownerToId(msg.sender)
```

```
        )) / daysInSecond;

    uint256 amountAlreadyClaimed = _claimedBy[msg.sender];
```

Fix interface in ISoulmate.sol

```
-function isDivorced() external view returns (bool);
+function isDivorced(address) external view returns (bool);
```

Fix Soulmate:isDivorced() to pass the address to check

```
-function isDivorced() public view returns (bool) {
-    return divorced[msg.sender];
+function isDivorced(address soulmate) public view returns (bool) {
+    return divorced[soulmate];
}
```

[H-04] Users without a soulmate can claim a large amount of airdrop tokens

Relevant GitHub Links https://github.com/Cyfrin/2024-02-soulmate/blob/b3f9227942ffd5c443ce6bccaa980fea0304c

Summary The Airdrop contract allows users without a soulmate to claim a large amount of tokens, which should not be possible.

Vulnerability Details The claim() function in Airdrop.sol does not check if the user has a soulmate, so the calculation takes a wrong number of days and sends and incorrect amount of tokens to the user.

This test calls claim() without a soulmate.

```
function testClaimAirdropWithoutSoulmate() public {
    // Jump to an actual timestamp in mainnet
    uint256 timestamp = 1707775379;
    vm.warp(timestamp);

    // Get a new user
    address alice = makeAddr("alice");

    // Claim
    vm.prank(alice);
    airdropContract.claim();

    console2.log("balance", loveToken.balanceOf(alice));
}
```

The test shows the tokens sent to alice.

```
Running 1 test for test/unit/AuditTest1.t.sol:AuditTest1
[PASS] testClaimAirdropWithoutSoulmate() (gas: 93299)
Logs:
  balance 1976500000000000000000
```

Impact Users without a soulmate can steal tokens from the airdrop vault.

Tools Used Foundry, Manual review

Recommendations Add a soulmate check in Airdrop:claim()

```
+error Airdrop__NoSoulmate();

function claim() public {
    // No LoveToken for people who don't love their soulmates anymore.
+   if (soulmateContract.soulmateOf(msg.sender) == address(0)) revert
    Airdrop__NoSoulmate();
    if (soulmateContract.isDivorced()) revert Airdrop__CoupleIsDivorced
        ();

    // Calculating since how long soulmates are reunited
    uint256 numberOfDaysInCouple = (block.timestamp -
        soulmateContract.idToCreationTimestamp(
            soulmateContract.ownerToId(msg.sender)
        )) / daysInSecond;

    uint256 amountAlreadyClaimed = _claimedBy[msg.sender];
```

Medium

[M-01] Any user can write a message in first shared space

Relevant GitHub Links https://github.com/Cyfrin/2024-02-soulmate/blob/b3f9227942ffd5c443ce6bccaa980fea0304

Summary Users without a soulmate can write a message in the shared space at the first position.

Vulnerability Details The writeMessageInSharedSpace() function in Soulmate.sol does not check if the sender has a soulmate. This allows any user to write a message in the shared space of the first soulbound token.

```
function testAnyUserCanWriteMessage() public {
    string memory message = "Hello!";
    address alice = makeAddr("alice");

    // Create a user without a soulmate
    vm.prank(alice);

    // Write and read a message
    soulmateContract.writeMessageInSharedSpace(message);
    uint256 tokenId = 0;
    string memory sharedMessage = soulmateContract.sharedSpace(tokenId)
        ;

    assertTrue(keccak256(abi.encodePacked(sharedMessage)) == keccak256(
        abi.encodePacked(message)));
}
```

The test confirms the message was written to position zero.

```
Running 1 test for test/unit/AuditTest1.t.sol:AuditTest1
[PASS] testAnyUserCanWriteMessage() (gas: 40523)
Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 1.79ms
```

Impact Any user can leave a message in the private space of soulbound token zero, as if it was any of the soulmates.

Tools Used Foundry, Manual review

Recommendations Add a soulmate check in Soulmate:mintSoulmateToken()

```
function writeMessageInSharedSpace(string calldata message) external {
+   if(soulmateOf[msg.sender] == address(0)) revert
    Soulmate__noSoulmate();
    uint256 id = ownerToId[msg.sender];
    sharedSpace[id] = message;
    emit MessageWrittenInSharedSpace(id, message);
}
```

[M-02] A user can mint a soulbound token with its same address as soulmate

Relevant GitHub Links https://github.com/Cyfrin/2024-02-soulmate/blob/b3f9227942ffd5c443ce6bccaa980fea0304

Summary A user can mint a soulbound token with the same address as soulmate1 and soulmate2.

Vulnerability Details The mintSoulmateToken() function does not check if the user is already minting as soulmate1, so if he calls mintSoulmateToken() again, he will mint a soulbound token with the same address as soulmate2.

This test mints a soulbound token with the same address as soulmate1 and soulmate2.

```
function testMintSoulmateTokenSameAddress() public {
    vm.startPrank(soulmate1);
    soulmateContract.mintSoulmateToken();
    soulmateContract.mintSoulmateToken();

    console2.log(soulmate1);
    console2.log(soulmateContract.soulmateOf(soulmate1));
}
```

soulmate1 and soulmateOf(soulmate1) are the same.

```
Running 1 test for test/unit/AuditTest1.t.sol:AuditTest1
[PASS] testMintSoulmateTokenSameAddress() (gas: 181670)
Logs:
  0x65629adcc2F9C857Aeb285100Cc00Fb41E78DC2f
  0x65629adcc2F9C857Aeb285100Cc00Fb41E78DC2f
```

Impact Users can mint soulbound tokens as singles and alter the functionality of the contract.

Tools Used Foundry, Manual review

Recommendations Fix mintSoulmateToken() in Soulmate.sol

```
+error Soulmate__alreadyMinting();
...

function mintSoulmateToken() public returns (uint256) {
    // Check if people already have a soulmate, which means already
       have a token
    address soulmate = soulmateOf[msg.sender];
    if (soulmate != address(0))
        revert Soulmate__alreadyHaveASoulmate(soulmate);

    address soulmate1 = idToOwners[nextID][0];
    address soulmate2 = idToOwners[nextID][1];
    if (soulmate1 == address(0)) {
        idToOwners[nextID][0] = msg.sender;
        ownerToId[msg.sender] = nextID;
        emit SoulmateIsWaiting(msg.sender);
    } else if (soulmate2 == address(0)) {
+       if(soulmate1 == msg.sender) revert Soulmate__alreadyMinting();
```

```
18          idToOwners[nextID][1] = msg.sender;
19          // Once 2 soulmates are reunited, the token is minted
20          ownerToId[msg.sender] = nextID;
21          soulmateOf[msg.sender] = soulmate1;
22          soulmateOf[soulmate1] = msg.sender;
23          idToCreationTimestamp[nextID] = block.timestamp;
24
25          emit SoulmateAreReunited(soulmate1, soulmate2, nextID);
26
27          _mint(msg.sender, nextID++);
28      }
```

Low

[L-01] Any user can get divorced without having a soulmate

Relevant GitHub Links https://github.com/Cyfrin/2024-02-soulmate/blob/b3f9227942ffd5c443ce6bccaa980fea0304c

Summary Any user can get divorced without having a soulmate, which should not be possible.

Vulnerability Details The getDivorced() function in Soulmate.sol does not check if the user has a soulmate, so he will be marked as divorced even if he does not have a soulmate.

Also, the function should check if the user is already divorced, to avoid marking the couple as divorced again.

This test calls getDivorced() without having a soulmate.

```
function testGetDivorcedWithoutSoulmate() public {
    address alice = makeAddr("alice");
    vm.prank(alice);
    soulmateContract.getDivorced();
    bool isDivorced = soulmateContract.isDivorced(alice);
    assertTrue(isDivorced);
}
```

The test passes, confirming that the user can get divorced without having a soulmate.

```
Running 1 test for test/unit/AuditTest1.t.sol:AuditTest1
[PASS] testGetDivorcedWithoutSoulmate() (gas: 59162)
Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 1.93ms
```

Impact Users may make mistakes and get divorced before they even have a soulmate, or when they are already divorced.

Tools Used Foundry, Manual review

Recommendations Add soulmate checks in Soulmate:getDivorced()

```
+error Soulmate__alreadyDivorced();

function getDivorced() public {
    address soulmate2 = soulmateOf[msg.sender];
+   if(soulmate2 == address(0)) revert Soulmate__noSoulmate();
+   if(divorced[msg.sender]) revert Soulmate__alreadyDivorced();
    divorced[msg.sender] = true;
    divorced[soulmateOf[msg.sender]] = true;
    emit CoupleHasDivorced(msg.sender, soulmate2);
}
```