Steaking Audit Report

Version 1.0

Owen Lee

August 27, 2024

Steaking Audit Report.

Owen Lee

August 27, 2024

Prepared by: Owen Lee. Lead Auditors: - Owen Lee.

Table of Contents

Protocol Summary

Steak is a yield farming protocol in its pre-launch phase. It boasts an attractive APY, various vault management strategies, and a strong and active community. Being in the pre-launch phase, Steak wants to bootstrap liquidity for its ERC4626 WETH vault and reward early adopters. For this, Steak has launched a points campaign where users can stake their ETH and earn points, which will allow users to be eligible for the $STEAK token airdrop in the future.

Disclaimer

I, Owen Lee, 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 team 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
Likelihood	High	H	H/M	M
	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

- In Scope:

Scope

```
src├──
 steaking-contracts│└──
    src│└──
        Steaking.vy└──
 steaking-server└──
     src├──
         models│└──
            steakPoints.js├──
         utils│├──
            connectToMongoDb.js│├──
```

```
            constants.js
            getConfig.js
        main.js
```

Roles

1. **Users**: Can stake and unstake raw ETH into the vault. After the staking period ends, users can convert ETH to WETH, and deposit it into the WETH vault.
2. **Steak protocol team multisig**: The multisig is the owner of the `Steaking` contract, and is responsible for setting the vault address after the staking period ends.

Executive Summary

Enjoyed the process of auditing this code, The developers of this codebase should spend a lot of time getting familiar with the common sets of vulnerabilities.

Issues found

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

Findings

High

[H-1] Incorrect handling of previous stakes.

Description:

The line `self.usersToStakes[_onBehalfOf] = msg.value` overwrites the user's previous stake with the new value instead of adding to the existing stake. If a user stakes multiple times, only the most recent staked amount will be recorded, effectively erasing their previous staked contributions.

```
def stake(_onBehalfOf: address):

    assert not self._hasStakingPeriodEnded(),
        STEAK__STAKING_PERIOD_ENDED
    assert msg.value >= MIN_STAKE_AMOUNT,
        STEAK__INSUFFICIENT_STAKE_AMOUNT
    assert _onBehalfOf != ADDRESS_ZERO, STEAK__ADDRESS_ZERO

@>  self.usersToStakes[_onBehalfOf] = msg.value
    self.totalAmountStaked += msg.value
    log Staked(msg.sender, msg.value, _onBehalfOf)
```

Impact:

This issue directly affects the accuracy of `user` staked balances and the overall integrity of the staking mechanism. It is a critical issue because it involves the potential loss of `user` staked funds, making it valid and severe.

Proof of Concept:

1. `User` has 16 ether as his balance.
2. `User` stakes 8 ether into the `steaking` contract.
3. `User` stakes another 8 ether into the `steaking` contract.
4. `User` expects to have staked 16 ether in the `steaking` contract.
5. `User` expects to have (16 * 1000) points accumulated over his staked ETH in the `steaking` contract.
6. `User` ends up getting (8 * 1000) points accumulated over his staked ETH due to wrong implementation of mapping.
7. The `steaking` contract recorded that the `user` has just staked just only 8 ETH instead of 16 ETH due to wrong implementation of mapping.

Code

Place the following into `Steaking.t.sol`.

```
function testUserMultipleStakeAttemptsAreRecorded() public {
        vm.deal(user1, 16 ether);

        vm.startPrank(user1);
        steaking.stake{value: 8 ether}(user1);
        steaking.stake{value: 8 ether}(user1);
        vm.stopPrank();
```

```
8
9           uint256 userStakedAmount = steaking.usersToStakes(user1);
10          uint256 totalAmountStaked = steaking.totalAmountStaked();
11
12          console2.log("This is how much the user Staked::",
               userStakedAmount);
13          console2.log("
               ------------------------------------------------------------");
14          console2.log("This is the total Amount Staked::",
               totalAmountStaked);
15          console2.log("--------------------------------------
               ------------------");
16
17          assertEq(userStakedAmount, 16 ether);
18          assertEq(totalAmountStaked, 16 ether);
19      }
```

Here are the logs that were shown:

```
[0] console::log("This is how much the user Staked::",
    8000000000000000000 [8e18]) [staticcall]│└─←
         [Stop]├─
    [0] console::log("
       ------------------------------------------------------------") [
       staticcall]│└─←
         [Stop]├─
    [0] console::log("This is the total Amount Staked::",
       16000000000000000000 [1.6e19]) [staticcall]│└─←
         [Stop]├─
    [0] console::log("
       ------------------------------------------------------------") [
       staticcall]│└─←
         [Stop]├─
    [0] VM::assertEq(8000000000000000000 [8e18], 16000000000000000000
       [1.6e19]) [staticcall]│└─←
         [Revert] assertion failed: 8000000000000000000 !=
            16000000000000000000└─←
     [Revert] assertion failed: 8000000000000000000 !=
       16000000000000000000
```

Here we can see an error where the mapping has recorded that the `user` had just staked just 8 ETH instead of 16 ETH. As the first assertion failed.

Recommended Mitigation:

Instead of overwriting, the function should add the new stake to the existing one:

```

def stake(_onBehalfOf: address):
```

```
3
4       assert not self._hasStakingPeriodEnded(),
            STEAK__STAKING_PERIOD_ENDED
5       assert msg.value >= MIN_STAKE_AMOUNT,
            STEAK__INSUFFICIENT_STAKE_AMOUNT
6       assert _onBehalfOf != ADDRESS_ZERO, STEAK__ADDRESS_ZERO
7
8   -   self.usersToStakes[_onBehalfOf] = msg.value
9   +   self.usersToStakes[_onBehalfOf] += msg.value
10      self.totalAmountStaked += msg.value
11      log Staked(msg.sender, msg.value, _onBehalfOf)
```