CryptoCritters
(OctoStake)

Security Review

June 6, 2024

Conducted by:
Blckhv, Independent Security Researcher
Slavcheww, Independent 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

OctoStake is a staking contract that allows its users to increase their daily yield by staking $BBC and $BWB, accumulating a daily fixed reward, additionally if they have AQUA NFTs that can be purchased on Opensea, this will further increase their yield.

Detailed information can be found - https://zibars-organization.gitbook.io/cryptocritters/octostake

Overview

Project	CryptoCritters (OctoStake)
Repository	[Private]
Commit Hash	—
Resolution	—
Date	June 5, 2024

Scope

ClaimAquatic.sol

Issues Found

Critical Risk	1
High Risk	0
Medium Risk	0
Low/Info Risk	3

5. Findings

5.1. Critical severity

5.1.1. Anyone can burn any AQUA nft

Severity: Critical Risk

Context: ClaimAquatic.sol#L76

Description: User can stake without having AQUA NFT by passing 0 as `nftTokenId` in the `stake()`. However if he passes any real NFT id, the staking contract will burn it as it has the rights to do so.

```
function stake(address token, uint256 amount, uint256 nftTokenId) external {
    require(!stopped, "The contract is stopped");
    require(token == BWB || token == BBC, "Incorrect token");
    if (token == BWB) {
        require(amount >= minimumBWBStakeAmount, "Low stake amount");
    } else {
        require(amount >= minimumBBCStakeAmount, "Low stake amount");
    }
    require(IERC20(token).transferFrom(msg.sender, address(this), amount), "Token transfer failed");
    IERC20(token).transfer(teamWallet, amount * stakeFee / 100);
    if (nftTokenId > 0) IERC721Burnable(AQUA).burn(nftTokenId); // <-------
    uint256 stakeId = globalStakeId++;
    stakes[stakeId] = Stake(msg.sender, block.timestamp, amount, nftTokenId, token, 0, true);
    _walletStakes[msg.sender].add(stakeId);
    emit Staked(msg.sender, token, amount, stakeId);
}
```

Recommendation: If `nftTokenId` is non-0, it should check if `msg.sender` is also its owner.

Resolution: Fixed

5.2. Low/Info severity

5.2.1. ERC20 operations aren't safe

Severity: Low Risk

Context: ClaimAquatic.sol#L74-75, L92, L107, L110-111, L150

Description: The code uses 3 ERC20 tokens, although they are all protocol related, it is better to change `ERC20.transfer()` to `ERC20.safeTransfer()`.

Tokens used in the code are pre-determined and no issues with their transfer functionality can be observed, however, it is advisable to use the safe transfer functionalities from OpenZeppelin.

Resolution: Fixed

5.2.2. Move the percentages to basis point representation

Severity: Low Risk

Context: ClaimAquatic.sol#L75, L110-111

Description: In the contract, `rewardMultipliers` and `bonusPoints` are denominated in basis points (10000 = 100%), but the protocol fees are normal (100 = 100%). It will be better for readability and multiplication safety if the `stake/unstake` fees are also representation in basis points.

Resolution: Fixed

5.2.3. Missing zero address checks

Severity: Low Risk

Context: ClaimAquatic.sol#L59

Description: All critical addresses are set in constructors with no way to change them, be sure to add zero address checks before setting them.

Resolution: Fixed