Puffer contracts

Security Review

Cantina Managed review by:

Windhustler, Security Researcher
0xWeiss, Security Researcher

February 5, 2025

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

Puffer is a decentralized native liquid restaking protocol (nLRP) built on Eigenlayer. It makes native restaking on Eigenlayer more accessible, allowing anyone to run an Ethereum Proof of Stake (PoS) validator while supercharging their rewards.

From Feb 4th to Feb 5th the Cantina team conducted a review of puffer-contracts on commit hash 036fd961. The team identified a total of **4** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	2	2	0
Gas Optimizations	0	0	0
Informational	2	1	1
Total	**4**	**3**	**1**

3 Findings

3.1 Low Risk

3.1.1 Use of OZ `Ownable` is risky

Severity: Low Risk

Context: CarrotStaker.sol#L16

Description: OpenZeppelin Ownable is used by CarrotStaker to manage contract ownership. However, this is risky because it allows single-step ownership transfers.

Recommendation: Consider using Ownable2Step that, as documented "*includes a two-step mechanism to transfer ownership, where the new owner must call acceptOwnership in order to replace the old one. This can help prevent common mistakes, such as transfers of ownership to incorrect accounts, or to contracts that are unable to interact with the permission system*".

Puffer Finance: Fixed in PR 98.

Cantina Managed: Fix verified.

3.1.2 Missing input validation in constructor

Severity: Low Risk

Context: *(No context files were provided by the reviewer)*

Description: The address of the carrot token is not validated:

```
constructor(address carrot, address initialOwner) ERC20("Staked Carrot", "sCarrot") Ownable(initialOwner) {
    CARROT = IERC20(carrot);
}
```

Recommendation: Check that the address of carrot can't be 0:

```
  constructor(address carrot, address initialOwner) ERC20("Staked Carrot", "sCarrot") Ownable(initialOwner) {
+     require(carrot != address(0),"not address 0");
      CARROT = IERC20(carrot);
  }
```

Puffer Finance: Fixed in PR 98.

Cantina Managed: Fix verified.

3.2 Informational

3.2.1 Incorrect symbol for staked token

Severity: Informational

Context: CarrotStaker.sol#L37

Description: The current symbol from staked carrot is set to `sCarrot`, in lowercase:

```
constructor(address carrot, address initialOwner) ERC20("Staked Carrot", "sCarrot") Ownable(initialOwner) {
```

while if the symbol of the CARROT token is set uppercase.

Recommendation: Make the following change to the symbol:

```
- constructor(address carrot, address initialOwner) ERC20("Staked Carrot", "sCarrot") Ownable(initialOwner) {
+ constructor(address carrot, address initialOwner) ERC20("Staked Carrot", "sCARROT") Ownable(initialOwner) {
```

Puffer Finance: Fixed in PR 98.

Cantina Managed: Fix verified.

3.2.2 Floating Pragma

Severity: Informational

Context: CarrotStaker.sol#L2

Description: The solidity compiler version should be fixed to clearly identify the Solidity version with which the contracts will be compiled instead of having a floating pragma:

```
pragma solidity >=0.8.0 <0.9.0;
```

Recommendation: Update the version to the fix version that will be used to compile the contract.

Puffer Finance: Acknowledged.

Cantina Managed: Acknowledged.