Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2025.09.11, the SlowMist security team received the Puffer Finance team's security audit application for Vesting Contract, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

The test method information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project team should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

Serial Number	Audit Class	Audit Subclass
1	Overflow Audit	-
2	Reentrancy Attack Audit	-
3	Replay Attack Audit	-
4	Flashloan Attack Audit	-
5	Race Conditions Audit	Reordering Attack Audit
6	Permission Vulnerability Audit	Access Control Audit
		Excessive Authority Audit
7	Security Design Audit	External Module Safe Use Audit
		Compiler Version Security Audit
		Hard-coded Address Security Audit
		Fallback Function Safe Use Audit
		Show Coding Security Audit
		Function Return Value Security Audit
		External Call Function Security Audit

Serial Number	Audit Class	Audit Subclass
7	Security Design Audit	Block data Dependence Security Audit
		tx.origin Authentication Security Audit
8	Denial of Service Audit	-
9	Gas Optimization Audit	-
10	Design Logic Audit	-
11	Variable Coverage Vulnerability Audit	-
12	"False Top-up" Vulnerability Audit	-
13	Scoping and Declarations Audit	-
14	Malicious Event Log Audit	-
15	Arithmetic Accuracy Deviation Audit	-
16	Uninitialized Storage Pointer Audit	-

3 Project Overview

3.1 Project Introduction

This is the CarrotVesting module for the puffer protocol.

3.2 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

NO	Title	Category	Level	Status
N1	Risk of excessive privilege	Authority Control Vulnerability Audit	Medium	Acknowledged

4 Code Overview

4.1 Contracts Description

https://github.com/PufferFinance/puffer-contracts

Initial audit version: 7f9e2b2ed98b8fe94e0530eb79262255fe220ab2

Final audit version: 7f9e2b2ed98b8fe94e0530eb79262255fe220ab2

Audit Scope:

```
- puffer-contracts/mainnet-contracts/src/CarrotVesting.sol
```

The main network address of the contract is as follows:

The code was not deployed to the mainnet.

4.2 Visibility Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

CarrotVesting			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
initializeVesting	External	Can Modify State	onlyOwner, reinitializer(2)
startVesting	External	Can Modify State	whenNotPaused
startVestingWithPermit	External	Can Modify State	whenNotPaused
claim	External	Can Modify State	whenNotPaused
recoverPuffer	External	Can Modify State	onlyOwner
pause	External	Can Modify State	onlyOwner

CarrotVesting			
unpause	External	Can Modify State	onlyOwner
getVestings	External	-	-
getTotalDepositedAmount	External	-	-
getIsDismantled	External	-	-
getStartTimestamp	External	-	-
getDuration	External	-	-
getSteps	External	-	-
calculateClaimableAmount	External	-	-
_calculateClaimableAmount	Internal	-	-
_startVesting	Internal	Can Modify State	whenNotPaused
_authorizeUpgrade	Internal	Can Modify State	onlyOwner

4.3 Vulnerability Summary

[N1] [Medium] Risk of excessive privilege

Category: Authority Control Vulnerability Audit

Content

The `Owner` role has three key permissions in the CarrotVesting contract:

1.Set the start time, duration, and number of steps of the vesting through the initializeVesting function.

2.All remaining PUFFER tokens in the contract can be withdrawn at any time through the `recoverPuffer` function.

Once this operation is performed, the user will no longer be able to start new vesting or claim any rewards.

3.The `pause` and `unpause` functions can be used to pause and unpause contract functions. In the paused state,

users cannot start new attribution or receive rewards.

4.The contract inherits from the Ownable2StepUpgradeable module of openzeppelin and can transfer contract ownership.

5.The contract inherits from the UUPSUpgradeable module of openzeppelin and can upgrade the contract implementation.

- mainnet-contracts/src/CarrotVesting.sol#L4-L7, L9, L99-L115, L168-L176, L182-L184, L190-L192, L313

```
    import {
    Ownable2StepUpgradeable,
    OwnableUpgradeable
} from "@openzeppelin/contracts-upgradeable/access/Ownable2StepUpgradeable.sol";

    import { UUPSUpgradeable } from "@openzeppelin-contracts-
upgradeable/proxy/utils/UUPSUpgradeable.sol";

    function initializeVesting(uint48 startTimestamp, uint32 duration, uint32 steps)
        external
        onlyOwner
        reinitializer(2)
    {}

    function recoverPuffer(address to) external onlyOwner returns (uint256) {}

    function pause() external onlyOwner {}

    function unpause() external onlyOwner {}

    function _authorizeUpgrade(address newImplementation) internal virtual override
onlyOwner { }
```

Solution

In the short term, to satisfy business requirements, managing the privileged role through a multi-signature scheme can effectively mitigate single-point risk. In the long term, entrusting these privileged roles to DAO governance can effectively resolve the risk of excessive privilege. During the transition period, managing through a multi-signature scheme combined with delayed transaction execution via a timelock can significantly alleviate the risk of excessive privilege.

Status

Acknowledged

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002509150002	SlowMist Security Team	2025.09.11 - 2025.09.15	Medium Risk

Summary conclusion: The SlowMist security team use a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 medium risk.

6 Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist