

Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

2 Audit Methodology

3 Project Overview

3.1 Project Introduction

3.2 Vulnerability Information

4 Code Overview

4.1 Contracts Description

4.2 Visibility Description

4.3 Vulnerability Summary

5 Audit Result

6 Statement

1 Executive Summary

On 2026.08.24, the SlowMist security team received the PrimeUSD team's security audit application for Boring Vault Incremental Audit V2, 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.

Level	Description
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

Serial Number	Audit Class	Audit Subclass
7	Security Design 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
		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

The audited project is the Boring Vault contract of the PrimeUSD protocol. This incremental audit focuses on fund management and Merkle-verified execution for strategy sub-accounts.

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/SwellNetwork/boring-vault/

Initial Audit Version: e90252cf2cbf87374ed253e6b9a3064d09570751

Final Audit Version: 3209585e78aa69d4326740a6538fda5476e27244

Audit scope:

```
- src/base/Roles/SubAccountFundManager.sol
- src/base/Roles/SubAccountWithMerkleVerification.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:

SubAccountFundManager			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setStrategyLimitKey	External	Can Modify State	requiresAuth
transferFundFromVault	External	Can Modify State	requiresAuth
transferFundBackToVault	External	Can Modify State	requiresAuth
_parseUint	Internal	-	-

SubAccountWithMerkleVerification			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setManageRoot	External	Can Modify State	requiresAuth
pause	External	Can Modify State	requiresAuth
unpause	External	Can Modify State	requiresAuth
manageWithMerkleVerification	External	Can Modify State	requiresAuth
manage	Public	Can Modify State	requiresAuth
_verifyCallData	Internal	-	-
_verifyManageProof	Internal	-	-
<receive>	External	Payable / Can Modify State	-

4.3 Vulnerability Summary

[N1] [Medium] Risk of Excessive Privilege

Category: Authority Control Vulnerability Audit

Content

1.In the `SubAccountFundManager` contract, the authorized address can configure the funding limit for the specified `strategyAddress` using `setStrategyLimitKey()`.

- src/base/Roles/SubAccountFundManager.sol#L85-L91

```
    function setStrategyLimitKey(
        address strategyAddress,
        string calldata limitKey
    ) external requiresAuth {
        //...
    }
```

2.In the `SubAccountWithMerkleVerification` contract, authorized addresses can configure each Strategist's permissions through `setManageRoot()` or, if authorized to call `manage()`, bypass Merkle verification and directly manage the sub-account's assets.

- src/base/Roles/SubAccountWithMerkleVerification.sol#L73-L80, L141-L149

```
    function setManageRoot(
        address strategist,
        bytes32 _manageRoot
    ) external requiresAuth {
        //...
    }

    function manage(
        address target,
        bytes calldata data,
        uint256 value
    ) public requiresAuth returns (bytes memory result) {
        //...
    }
```

Solution

In the short term, transferring ownership and privileged roles to a multi-signature wallet or Timelock contract is recommended to mitigate single-point-of-failure risks. In the long term, privilege separation should assign upgrades, trust-anchor changes, compliance actions, fee policy, and emergency controls to distinct constrained roles. Critical verifier and implementation changes should use pending-and-accept activation with an on-chain delay that gives users and operators time to react.

Status

Acknowledged

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002608250001	SlowMist Security Team	2026.08.24 - 2026.08.24	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 vulnerability.

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