



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 2024.12.23, the SlowMist security team received the litentry team's security audit application for HEI-token, 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

A wrapped ERC20 token that burns underlying tokens to mint wrapped ones.

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Missing zero address check	Design Logic Audit	Suggestion	Fixed
N2	Risk of excessive authority	Authority Control Vulnerability Audit	Low	Acknowledged

4 Code Overview

4.1 Contracts Description

<https://github.com/litentry/HEI-token/tree/audit-20241213>

Initial audit version: 6b0c33f1f39cbd0b5b45e28dd1a360151581e3a5

Final audit version: 0f7012b64df6499f827235f4297ee1f9a61dbeab

The main network address of the contract is as follows:

Ethereum address: 0xF8F173E20e15f3b6CB686FB64724D370689De083

BSC address: 0xF8F173E20e15f3b6CB686FB64724D370689De083

4.2 Visibility Description

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

HEI			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20 AccessControlDefaultAdminRules
decimals	Public	-	-
underlying	External	-	-
depositFor	External	Can Modify State	-
mint	External	Can Modify State	onlyRole
grantMinter	External	Can Modify State	onlyRole

4.3 Vulnerability Summary

[N1] [Suggestion] Missing zero address check

Category: Design Logic Audit

Content

In the HEI contract, the `constructor` function lacks zero address checks for the `underlyingToken` parameters.

- contracts/heima/HEI.sol#L28-L39

```

constructor(
    IERC20 underlyingToken,
    string memory _name,
    string memory _symbol,
    address _admin
) ERC20(_name, _symbol) AccessControlDefaultAdminRules(0, _admin) {
    if (underlyingToken == this) {
        revert ERC20InvalidUnderlying(address(this));
    }
    _underlying = underlyingToken;
    _setRoleAdmin(MINT_ROLE, DEFAULT_ADMIN_ROLE);
}

```

Solution

It is recommended to add a zero address check.

Status

Fixed

[N2] [Low] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

In the HEI contract, the `DEFAULT_ADMIN_ROLE` role can set the `MINT_ROLE` role through the `grantMinter` function; the `MINT_ROLE` role can mint tokens through the `mint` function.

- contracts/heima/HEI.sol#L83-L89, L94-L101

```

function mint(
    address account,
    uint256 value
) external onlyRole(MINT_ROLE) returns (bool) {
    _mint(account, value);
    return true;
}

```

```
function grantMinter(  
    address _minter  
) external onlyRole(DEFAULT_ADMIN_ROLE) returns (bool) {  
    if (_minter == address(0)) {  
        revert AccessControlInvalidMinter(address(0));  
    }  
    return _grantRole(MINT_ROLE, _minter);  
}
```

Solution

In the short term, during the early stages of the project, the protocol may need to frequently set various parameters to ensure the stable operation of the protocol. Therefore, transferring the ownership of core roles to a multisig management can effectively solve the single-point risk, but it cannot mitigate the excessive privilege risk. In the long run, after the protocol stabilizes, transferring the owner ownership to community governance and executing through a timelock can effectively mitigate the excessive privilege risk and increase the community users' trust in the protocol.

Status

Acknowledged; On-chain verification shows DEFAULT_ADMIN_ROLE has been transferred to and is managed by a multi-signature wallet.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002412270001	SlowMist Security Team	2024.12.23 - 2024.12.27	Low 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, 1 suggestion.

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](https://twitter.com/SlowMist_Team)



Github
<https://github.com/slowmist>