Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2025.12.29, the SlowMist security team received the htx-dao team's security audit application for htx-dao, 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 audit mainly focuses on an upgradable ERC4626-standard tokenized vault that allows users to stake HTX tokens to obtain sHTX shares. Stakers can earn protocol revenues, which are regularly distributed to this contract and become fully available only after a vesting period. A core feature of this contract is its withdrawal mechanism: it includes a configurable "cooldown period". If the cooldown period is zero, users can withdraw assets immediately. If a cooldown period is set, users must first initiate a withdrawal request and wait until the cooldown period ends to finally retrieve their assets.

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Loss of Precision in Vesting Calculation	Arithmetic Accuracy Deviation Vulnerability	Suggestion	Fixed
N2	Lack of Zero-address Check in addToBlacklist Function	Others	Suggestion	Fixed
N3	Lack of Return Value Check	Others	Suggestion	Fixed
N4	Risk of Excessive Authority	Authority Control Vulnerability Audit	Medium	Acknowledged

4 Code Overview

4.1 Contracts Description

Audit Version:

https://github.com/jerkoyz/htx-dao

commit: 8d191293e8e5090f562f2cb14fe3bc7f60212e2b

Fixed Version:

https://github.com/jerkoyz/htx-dao

commit: e263d199825515593c71822b62b789d0d9c306eb

Audit Scope:

```
./contracts
├── StakedHTX.sol
├── HTXSilo.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:

HTXSilo			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
withdraw	External	Can Modify State	onlyStakingVault

StakedHTX			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
transferInRewards	External	Can Modify State	nonReentrant onlyRole notZero
addToBlacklist	External	Can Modify State	onlyRole
removeFromBlacklist	External	Can Modify State	onlyRole
totalAssets	Public	-	-
getUnvestedAmount	Public	-	-
decimals	Public	-	-
_authorizeUpgrade	Internal	Can Modify State	onlyUpgrader
_update	Internal	Can Modify State	-
_checkMinShares	Internal	-	-
_deposit	Internal	Can Modify State	nonReentrant notZero notZero whenNotPaused
_withdraw	Internal	Can Modify State	nonReentrant notZero notZero whenNotPaused

StakedHTX			
_updateVestingAmount	Internal	Can Modify State	-
withdraw	Public	Can Modify State	ensureCooldownOff
redeem	Public	Can Modify State	ensureCooldownOff
unstake	External	Can Modify State	nonReentrant whenNotPaused
cooldownAssets	External	Can Modify State	ensureCooldownOn
cooldownShares	External	Can Modify State	ensureCooldownOn
setCooldownDuration	External	Can Modify State	onlyRole
setTransferEnabled	External	Can Modify State	onlyRole
setUpgrader	External	Can Modify State	onlyRole
pause	External	Can Modify State	onlyRole
unpause	External	Can Modify State	onlyRole

4.3 Vulnerability Summary

[N1] [Suggestion] Loss of Precision in Vesting Calculation

Category: Arithmetic Accuracy Deviation Vulnerability

Content

In the transferInRewards function of the StakedHTX.sol contract, this function is responsible for transferring rewards into the contract and initiating linear vesting. Since the amount parameter entered is not checked for being too small, when the amount value is small (such as less than VESTING_PERIOD), in the getUnvestedAmount function, during the calculation of (deltaT * vestingAmount) / VESTING_PERIOD, due to the downward - rounding characteristic of Solidity integer division, the result will be 0. This causes the system to misjudge that the vesting has been completed

even when it is within the vesting period (near the end of the vesting period), thus bypassing the check in _updateVestingAmount. Eventually, it allows for the update of new rewards before the vesting period ends.

Code Location:

src/contracts/StakedHTX.sol#L129

```
    function transferInRewards(uint256 cursor, uint256 amount) external nonReentrant
onlyRole(REWARDER_ROLE) notZero(amount) {
        _updateVestingAmount(cursor, amount);

        ...
    }

    function getUnvestedAmount() public view returns (uint256) {
        ...

        return (deltaT * vestingAmount) / VESTING_PERIOD;
    }
```

Solution

Add a minimum value check for the amount in the transferInRewards function. For example, require that amount must be no less than 1e18 to prevent loss of calculation precision.

Status

Fixed

[N2] [Suggestion] Lack of Zero-address Check in addToBlacklist Function

Category: Others

Content

In the addToBlacklist function of the StakedHTX.sol contract, this function allows administrators with the BLACKLIST_MANAGER_ROLE to add addresses to the blacklist to restrict their operations. When adding the target address to the blacklist mapping, the function does not check whether the target is the zero address (address(0)). If an administrator accidentally adds the zero address to the blacklist, it will damage the core functions of the contract. For example, the contract's internal hook _update checks the blacklist every time a token transfer (including minting and burning) occurs. Since the from address for minting operations is address(0) and the to address for burning

operations is address(0), blacklisting the zero address will cause all minting and burning related operations (such as staking, redemption) to fail due to the OperationNotAllowed error, resulting in a functional denial-of-service.

Code Location:

src/contracts/StakedHTX.sol#L139-144

```
function addToBlacklist(address target)external onlyRole(BLACKLIST_MANAGER_ROLE){
     if (hasRole(DEFAULT_ADMIN_ROLE, target)) revert CantBlacklist();
     if(blacklist[target]) revert InvalidParam();
     blacklist[target] = true;
     emit BlacklistUpdated(target, true);
 }
```

Solution

Add a validation for the target parameter in the addToBlacklist function to ensure it is not the zero address.

Status

Fixed

[N3] [Suggestion] Lack of Return Value Check

Category: Others

Content

In the withdraw function of the HTXSilo.sol contract, this function is responsible for withdrawing assets (ASSET) to a specified address. The function directly calls IERC20(ASSET).transfer(to, amount) without checking its boolean return value.

Code Location:

src/contracts/HTXSilo.sol#L31

```
function withdraw(address to, uint256 amount) external onlyStakingVault {
     IERC20(ASSET).transfer(to, amount);
 }
```

Solution

Use the OpenZeppelin's SafeERC20 library and replace the transfer call with safeTransfer.

Status

Fixed

[N4] [Medium] Risk of Excessive Authority

Category: Authority Control Vulnerability Audit

Content

1.In the setUpgrader function of the StakedHTX.sol contract, this function allows an address with the DEFAULT_ADMIN_ROLE to unilaterally set a new upgrader address. The upgrader address has the highest authority to upgrade the contract logic. If the private key of the DEFAULT_ADMIN_ROLE or the upgrader role is stolen, an attacker can immediately update the contract code to malicious logic, thereby stealing all the funds in the contract.

2.In the addToBlacklist function of the StakedHTX.sol contract, this function allows an address with the BLACKLIST_MANAGER_ROLE to unilaterally add any user address to the blacklist. The blacklisted address will be unable to perform core operations such as transfers. If this permission is abused or the relevant private key is stolen, it may lead to malicious censorship, freezing the assets of innocent users, and damaging users' asset ownership and the censorship-resistance of the protocol.

Code Location:

src/contracts/StakedHTX.sol

```
    function addToBlacklist(address target)external onlyRole(BLACKLIST_MANAGER_ROLE){
        ...
    }

    function removeFromBlacklist(address target) external
onlyRole(BLACKLIST_MANAGER_ROLE) {
        ...
    }

    function setUpgrader(address _upgrader) external onlyRole(DEFAULT_ADMIN_ROLE) {
        ...
    }
```

Solution

In the short term, transferring the ownership of core roles to time-locked contracts and managing them by multi-signatures is an effective solution to avoid single-point risks. However, in the long run, a more reasonable solution is

to implement a permission separation strategy and set up multiple privileged roles to manage each privileged function separately. Permissions involving user funds and contract core parameter updates should be managed by the community, while permissions involving emergency contract suspensions can be managed by EOA addresses. This ensures the safety of user funds while responding quickly to threats.

Status

Acknowledged; The project team responded: For the relevant role permissions, we will ensure their security by leveraging the multi-signature feature of TRON accounts. The DEFAULT_ADMIN will adopt a 2/3 multi-signature setup, and the upgrader will also have a 3/3 multi-signature configuration.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002512290001	SlowMist Security Team	2025.12.29 - 2025.12.29	Medium Risk

Summary conclusion: The SlowMist security team uses a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 medium risk and 3 suggestion. All the findings were fixed or acknowledged. Since the project has not yet been deployed to the mainnet and the permissions of the core roles have not yet been transferred, the risk level reported is temporarily medium.

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