AUDITMOS

Secure your space

STARHEROES

audit / code review report

February 26, 2024

TABLE OF CONTENTS

contact@auditmos.com

AUDITMOS

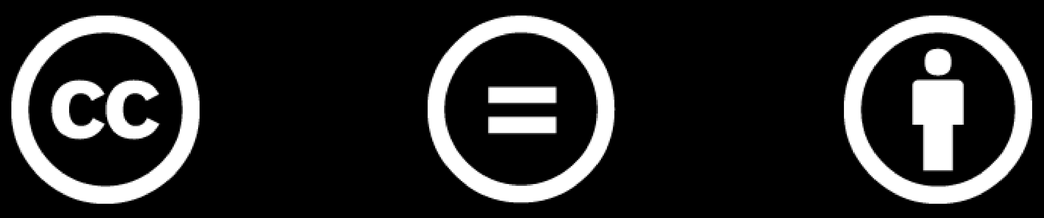
Secure your space

STARHEROES

audit / code review report

February 26, 2024

LICENSE



contact@auditmos.com

AUDITMOS

Secure your space

STARHEROES

audit / code review report

February 26, 2024

DISCLAIMER

contact@auditmos.com

AUDITMOS

Secure your space

STARHEROES

audit / code review report

February 26, 2024

APPROACH AND METHODOLOGY

PURPOSE

1. Determine the correct operation of the protocol, according to the design specification.
2. Identify possible vulnerabilities that could be exploited by an attacker.
3. Detect errors in the smart contract that could lead to unexpected behavior.
4. Analyze whether best practices were followed during development.
5. Make recommendations to improve security and code readability.

CODEBASE

Repository	https://github.com/starheroescommunity/sh-contracts
Branch	main
Commit hash	2807c97cbfa0e8f672e2364b288c011be5071fdf

METHODOLOGY

1. Reading the available documentation and understanding the code.
2. Doing automated code analysis and reviewing dependencies.
3. Checking manually source code line by line for security vulnerabilities.
4. Following guildlines and recommendations.
5. Preparing this report.

AUDITMOS

Secure your space

STARHEROES

audit / code review report

February 26, 2024

DESCRIPTION

Issues Categories:

Severity	Description
CRITICAL	vulnerability that can lead to loss of funds, failure to recover blocked funds, or catastrophic denial of service.
HIGH	vulnerability that can lead to incorrect contract state or unpredictable operation of the contract.
MEDIUM	failure to adhere to best practices, incorrect usage of primitives, without major impact on security.
LOW	recommendations or potential optimizations which can lead to better user experience or readability.

Each issue can be in the following state:

State	Description
PENDING	still waiting for resolving
ACKNOWLEDGED	know but not planned to resolve for some reasons
RESOLVED	fixed and deployed

contact@auditmos.com

AUDITMOS

Secure your space

STARHEROES

audit / code review report

February 26, 2024

AUDIT SCOPE

1. getting to know the project	✓
2. research into architecture	✓
3. manual code read	✓
4. permissions of state changing functions	✓
5. identify common Solidity vulnerabilities	✓
6. test coverage	✓
7. static analysis	✓
8. storage key overlaps	✓
9. DOS possibilities by malicious attacker	✓
10. steal funds possibilities	✓

contact@auditmos.com

AUDITMOS

Secure your space

STARHEROES

audit / code review report

February 26, 2024

FINDINGS

Finding	Severity	Status
#1 - It is possible to create UnbondInfo with _amount == 0	MEDIUM	RESOLVED
#2 - Unnecessary external call in checkEthFeeAndRefundDust modifier	MEDIUM	RESOLVED
#3 - Include old fee value in event	LOW	RESOLVED

contact@auditmos.com

AUDITMOS

Secure your space

STARHEROES

audit / code review report

February 26, 2024

#1 - IT IS POSSIBLE TO CREATE UNBONDINFO WITH _AMOUNT == 0

Severity	Status
MEDIUM	RESOLVED

Every user can call the startUnstaking function with _amount equal to 0 and to create an UnbondInfo position. They do not need to stake anyamount before that and it will emit the UnstakeStarted(msg.sender, _amount) event.

RECOMMENDATION

Recommended mitigation steps:

```
function startUnstaking(uint256 _amount) public payable
checkEthFeeAndRefundDust(msg.value) nonReentrant { UserInfo storage user =
userInfo[msg.sender];
require(_amount > 0, "Zero amount");
require(user.unbondings.length < unbondLimit, "startUnstaking: limit reached");
require(user.amount >= _amount, "startUnstaking: not enough staked amount");
```

contact@auditmos.com

STARHEROES

audit / code review report

February 26, 2024

#2 - UNNECESSARY EXTERNAL CALL IN CHECKETHFEEANDREFUNDDUST MODIFIER

Severity	Status
MEDIUM	RESOLVED

When a user calls payable function, he needs to pay a fee that is greater than or equal to the storage variable ethFee. If msg.value is largerthan ethFee, the remaining fee will be returned back to the user dust = value - ethFee. In the case where msg.value == ethFee, thevariable dust will be zero, and an unnecessary external call to msg.sender will be executed. This will increase the transaction cost.

RECOMMENDATION

It is recommend to add more stricly condiiton for fee.

Instead of:

```
require(value >= ethFee, "Insufficient fee: the required fee must be covered");
```

Use:

```
require(value == ethFee, "Insufficient fee: the required fee must be covered");
```

STARHEROES

audit / code review report

February 26, 2024

#3 - INCLUDE OLD FEE VALUE IN EVENT

Severity	Status
LOW	RESOLVED

After changing the fee value in the updateEthFee function, it would be good to include the old value of the fee in the UpdateFee event.

Instead of:

```
emit UpdateFee(_newFee);
```

Use:

```
emit UpdateFee(oldFee, _newFee);
```

auditmos.com

Secure your space

contact@auditmos.com