Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2023.06.19, the SlowMist security team received the Synclub team's security audit application for Synclub, 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:

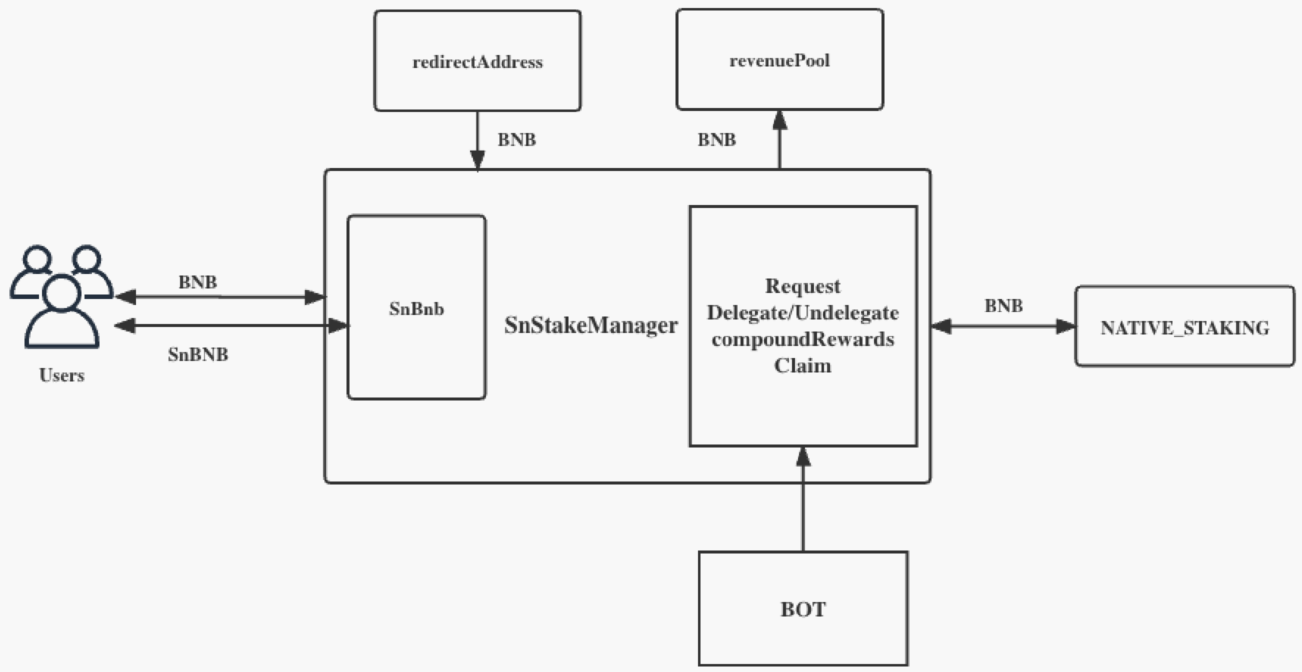
Serial Number	Audit Class	Audit Subclass
1	Overflow Audit	-
2	Reentrancy Attack Audit	-
3	Replay Attack Audit	-
4	Flashloan Attack Audit	-
5	Race Conditions Audit	Reordering Attack Audit
6	Permission Vulnerability Audit	Access Control Audit
		Excessive Authority Audit
7	Security Design Audit	External 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

These are the Synclub StakeManager contract, users can dposeit their BNB for liquidity staking deposit. The contracts are responsible for BNB deposits, claim, and withdrawals, minting and burning liquid tokens(SnBNB), delegating funds to validators, applying fees, and accepting updates from the BOT role.



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 authority	Authority Control Vulnerability Audit	Medium	Acknowledged
N2	The mint amount can be 0 in the deposit function	Design Logic Audit	Low	Fixed
N3	The BNB can be remained in the contract	Arithmetic Accuracy Deviation Vulnerability	Suggestion	Acknowledged
N4	The business logic is unclear	Design Logic Audit	Suggestion	Acknowledged
N5	Missing the validator check	Others	Suggestion	Fixed
N6	Missing the event records	Others	Suggestion	Fixed
N7	Preemptive initialization	Race Conditions Vulnerability	Suggestion	Acknowledged

NO	Title	Category	Level	Status
N8	Dev address setting enhancement suggestions	Others	Suggestion	Acknowledged

4 Code Overview

4.1 Contracts Description

Audit version:

https://github.com/helio-money/synclub-contracts

commit: ab8e36ff7760caa635cc0197b6ebf1bcfd574b57

Fixed version

https://github.com/helio-money/synclub-contracts

commit: 5487d2bbf8116ae8baca38a01ccf465003dc6ea7

The main network address of the contract is as follows:

Contract Address	
Contract Name	Address
Proxy SnBnb	0xB0b84D294e0C75A6abe60171b70edEb2EFd14A1B
Impl SnBnb	0xaF8DC8A33B60173693590BD867d571D88501CF81
Proxy SnStakeManager	0x1adB950d8bB3dA4bE104211D5AB038628e477fE6
Impl SnStakeManager	0xD24f4Bd59fd9C05520f58072a3d3dCF576aaC382

4.2 Visibility Description

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

SnBnb			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
mint	External	Can Modify State	onlyStakeManager
burn	External	Can Modify State	onlyStakeManager
setStakeManager	External	Can Modify State	onlyRole

SnStakeManager			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
deposit	External	Payable	whenNotPaused
delegate	External	Payable	whenNotPaused onlyRole
delegateWithReserve	External	Payable	whenNotPaused onlyRole
redelegate	External	Payable	whenNotPaused onlyManager
compoundRewards	External	Can Modify State	whenNotPaused onlyRole
requestWithdraw	External	Can Modify State	whenNotPaused
claimWithdraw	External	Can Modify State	whenNotPaused
undelegate	External	Payable	whenNotPaused onlyRole
claimUndelegated	External	Can Modify State	whenNotPaused onlyRole
claimFailedDelegation	External	Can Modify State	whenNotPaused onlyRole

SnStakeManager			
depositReserve	External	Payable	whenNotPaused onlyRedirectAddress
withdrawReserve	External	Can Modify State	whenNotPaused onlyRedirectAddress
setReserveAmount	External	Can Modify State	onlyManager
proposeNewManager	External	Can Modify State	onlyManager
acceptNewManager	External	Can Modify State	-
setBotRole	External	Can Modify State	onlyManager
revokeBotRole	External	Can Modify State	onlyManager
setBCValidator	External	Can Modify State	onlyManager
setSynFee	External	Can Modify State	onlyRole
setRedirectAddress	External	Can Modify State	onlyRole
setRevenuePool	External	Can Modify State	onlyRole
getTotalPooledBnb	Public	-	-
getContracts	External	-	-
getBotUndelegateRequest	External	-	-
getUserWithdrawalRequests	External	-	-
getUserRequestStatus	External	-	-
getSnBnbWithdrawLimit	External	-	-
getTokenHubRelayFee	Public	-	-
convertBnbToSnBnb	Public	-	-
convertSnBnbToBnb	Public	-	-

SnStakeManager			
togglePause	External	Can Modify State	onlyRole
<Receive Ether>	External	Payable	-

4.3 Vulnerability Summary

[N1] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

1.In the SnBnb contract, the DEFAULT_ADMIN_ROLE can set the stakeManager contract as the StakeManager role and the StakeManager role can call the mint and burn functions to mint tokens arbitrarily and burn any users' tokens.

Code location:

SnBnb.sol#26-54

```
    function mint(address _account, uint256 _amount)
        external
        override
        onlyStakeManager
    {
        _mint(_account, _amount);
    }

    function burn(address _account, uint256 _amount)
        external
        override
        onlyStakeManager
    {
        _burn(_account, _amount);
    }

    function setStakeManager(address _address)
        external
        override
        onlyRole(DEFAULT_ADMIN_ROLE)
    {
        require(stakeManager != _address, "Old address == new address");
        require(_address != address(0), "zero address provided");

        stakeManager = _address;

        emit SetStakeManager(_address);
    }
```

2.In the SnStakeManager contract, the Manager role can set/revoke the BOT contract, change the bcValidator contract, and the DEFAULT_ADMIN_ROLE can set the synFee parameter, the redirectAddress address, and the revenuePool address. The fee can be set as 10^10 and obtain all the amount(reward) from the claimReward to the revenuePool. And the BOT role controls the delegate, delegateWithReserve, compoundRewards, undelegate, claimUndelegated, and claimFailedDelegation functions to delegate/undelegate or claim. All these can affect the staking process. And the BOT contract is out of the audit scope.

Code location:

SnStakeManager.sol#413-479

```
    function setBotRole(address _address) external override onlyManager {
        require(_address != address(0), "zero address provided");

        _setupRole(BOT, _address);

        emit SetBotRole(_address);
    }

    function revokeBotRole(address _address) external override onlyManager {
        require(_address != address(0), "zero address provided");

        _revokeRole(BOT, _address);

        emit RevokeBotRole(_address);
    }

    /// @param _address - Beck32 decoding of Address of Validator Wallet on Beacon
Chain with `0x` prefix
    function setBCValidator(address _address)
        external
        override
        onlyManager
    {
        require(bcValidator != _address, "Old address == new address");
        require(_address != address(0), "zero address provided");
        bcValidator = _address;

        emit SetBCValidator(_address);
    }

    function setSynFee(uint256 _synFee)
        external
        override
        onlyRole(DEFAULT_ADMIN_ROLE)
    {
        require(_synFee <= TEN_DECIMALS, "_synFee must not exceed 10000 (100%)");

        synFee = _synFee;

        emit SetSynFee(_synFee);
    }

    function setRedirectAddress(address _address)
        external
        override
        onlyRole(DEFAULT_ADMIN_ROLE)
    {
        require(redirectAddress != _address, "Old address == new address");
        require(_address != address(0), "zero address provided");

        redirectAddress = _address;

        emit SetRedirectAddress(_address);
    }

    function setRevenuePool(address _address)
        external
        override
        onlyRole(DEFAULT_ADMIN_ROLE)
    {
        require(revenuePool != _address, "Old address == new address");
        require(_address != address(0), "zero address provided");

        revenuePool = _address;

        emit SetRevenuePool(_address);
    }
```

3.The contracts are TransparentUpgradeableProxy contracs, the owner role of the proxy can upgrade the contract.

Solution

In the short term, transferring owner ownership to multisig contracts is an effective solution to avoid single-point risk. But in the long run, it is a more reasonable solution to implement a privilege separation strategy and set up multiple privileged roles to manage each privileged function separately. And the authority involving user funds should be managed by the community, and the authority involving emergency contract suspension can be managed by the EOA address. This ensures both a quick response to threats and the safety of user funds. Please also ensures the security and reliability of the external execution contract.

Status

Acknowledged

[N2] [Low] The mint amount can be 0 in the deposit function

Category: Design Logic Audit

Content

In the SnStakeManager contract, users deposit their BNB into this contract and obtain the SnBNB as the staking certificate. And the calculation of the `snBnbToMint` is dependent on the convertBnbToSnBnb function, the totalSupply of the SnBNB, and the totalPooledBnb in this contract. If the deposit amount of the BNB is small enough or the totalPooledBnb is big enough, the calculation of the `snBnbToMint` can be 0. But the amount of the BNB can still add to the amountToDelegate to cause the increment of the totalPooledBnb.

Code location:

SnStakeManager.sol#116-125

```
    function deposit() external payable override whenNotPaused {
        uint256 amount = msg.value;
        require(amount > 0, "Invalid Amount");

        uint256 snBnbToMint = convertBnbToSnBnb(amount);

        amountToDelegate += amount;

        ISnBnb(snBnb).mint(msg.sender, snBnbToMint);
    }
```

Solution

It's recommended to add the `require(snBnbToMint > 0, "MINT_ZERO");` check.

Status

Fixed

[N3] [Suggestion] The BNB can be remained in the contract

Category: Arithmetic Accuracy Deviation Vulnerability

Content

In the SnStakeManager contract when users claim to withdraw their BNB tokens in the claimWithdraw in the same uuid, the calculation of the `amount = (totalBnbToWithdraw_ * amountInSnBnb) / totalSnBnbToBurn_;` has the rounding to obtain one of the user's withdrawal amount. It will cause the rounded amount of the BNB to remain in this contract and can not be withdrawn.

Code location:

SnStakeManager.sol#286-287

```
    function claimWithdraw(uint256 _idx) external override whenNotPaused {
        address user = msg.sender;
        WithdrawalRequest[] storage userRequests = userWithdrawalRequests[user];

        require(_idx < userRequests.length, "Invalid index");
        ...
        uint256 totalBnbToWithdraw_ = botUndelegateRequest.amount;
        uint256 totalSnBnbToBurn_ = botUndelegateRequest.amountInSnBnb;
        uint256 amount = (totalBnbToWithdraw_ * amountInSnBnb) /
            totalSnBnbToBurn_;

        AddressUpgradeable.sendValue(payable(user), amount);

        emit ClaimWithdrawal(user, _idx, amount);
    }
```

Solution

It's recommended to add a global variable to record the total undelegatedAmount value in one uuid and decrease all the withdrawal amount of user in this uuid to sync the BNB amount. Or remain the rounding BNB in the contract as the gas price.

Status

Acknowledged

[N4] [Suggestion] The business logic is unclear

Category: Design Logic Audit

Content

In the SnStakeManager contract, the claimUndelegated function calculates the claimUndelegated withdrawal value in one uuid and assigns it to two temporary variables. The two temporary variables are just for recording and have no other usage.

Code location:

SnStakeManager.sol#355-356

```
    function claimUndelegated()
        external
        override
        whenNotPaused
        onlyRole(BOT)
        returns (uint256 _uuid, uint256 _amount)
    {
        uint256 undelegatedAmount = IStaking(NATIVE_STAKING).claimUndelegated();
        require(undelegatedAmount > 0, "Nothing to undelegate");
        for (uint256 i = confirmedUndelegatedUUID; i <= nextUndelegateUUID - 1; i++)
  {
            BotUndelegateRequest
                storage botUndelegateRequest = uuidToBotUndelegateRequestMap[i];
            botUndelegateRequest.endTime = block.timestamp;
            confirmedUndelegatedUUID++;
        }
        _uuid = confirmedUndelegatedUUID;
        _amount = undelegatedAmount;
    }
```

Solution

It is recommended to clarify the logic implementation.

Status

Acknowledged

[N5] [Suggestion] Missing the validator check

Category: Others

Content

In the SnStakeManager contract, the Manager role can change the Validator through the redelegate function, and this check is done by the NATIVE_STAKING contract, and if the call of the redelegate function failed, it will consume the gas of this call.

Code location:

SnStakeManager.sol#187-207

```
    function redelegate(address srcValidator, address dstValidator, uint256 amount)
        external
        payable
        override
        whenNotPaused
        onlyManager
        returns (uint256 _amount)
    {
        ...
        // redelegate through native staking contract
        IStaking(NATIVE_STAKING).redelegate{value: msg.value}(srcValidator,
dstValidator, amount);

        emit ReDelegate(srcValidator, dstValidator, amount);

        return amount;
    }
```

Solution

It's recommended to add the `require(srcValidator != dstValidator, "invalid redelegation");` check.

Status

Fixed

[N6] [Suggestion] Missing the event records

Category: Others

Content

There are no event logs of the claimUndelegated and claimFailedDelegation in this SnStakeManager contract.

Code location:

SnStakeManager.sol#340,359

```
    function claimUndelegated()
        external
        override
        whenNotPaused
        onlyRole(BOT)
        returns (uint256 _uuid, uint256 _amount)
    {
        uint256 undelegatedAmount = IStaking(NATIVE_STAKING).claimUndelegated();
        ...
        _uuid = confirmedUndelegatedUUID;
        _amount = undelegatedAmount;
    }

    function claimFailedDelegation()
        external
        override
        whenNotPaused
        onlyRole(BOT)
        returns (uint256 _amount)
    {
        uint256 failedAmount = IStaking(NATIVE_STAKING).claimUndelegated();
        amountToDelegate += failedAmount;
        return failedAmount;
    }
```

Solution

It is recommended to record events when sensitive parameters are modified for self-inspection or community review.

Status

Fixed

[N7] [Suggestion] Preemptive initialization

Category: Race Conditions Vulnerability

Content

By calling the initialize function to initialize the contracts, there is a potential issue that malicious attackers preemptively call the initialize function to initialize.

Code location:

SnStakeManager.sol#73

SnBnb.sol#17

```
        function initialize(
         address _snBnb,
         address _admin,
         address _manager,
         address _bot,
         uint256 _synFee,
         address _revenuePool,
         address _validator
     ) external override initializer {
         ...
     }

     function initialize(address _admin) external override initializer {
         ...
     }
```

Solution

It is suggested that the initialize operation can be called in the same transaction immediately after the contract is created to avoid being maliciously called by the attacker.

Status

Acknowledged

[N8] [Suggestion] Dev address setting enhancement suggestions

Category: Others

Content

In the SnStakeManager contract, the GOVERNANCE_ROLE role can set the revenuePool address to receive the fee. If the address is an EOA address, in a scenario where the private keys are leaked, the team's revenue will be stolen.

Code location:

SnStakeManager.sol#468-479

```
     function setRevenuePool(address _address)
         external
         override
```

```
        onlyRole(DEFAULT_ADMIN_ROLE)
    {
        require(revenuePool != _address, "Old address == new address");
        require(_address != address(0), "zero address provided");

        revenuePool = _address;

        emit SetRevenuePool(_address);
    }
```

Solution

It is recommended to set the insurance address as a multi-signature contract to avoid the leakage of private keys and the theft of team rewards.

Status

Acknowledged

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002306210001	SlowMist Security Team	2023.06.19 - 2023.06.21	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, 1 low risk, 6 suggestion vulnerabilities.

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.