

Syncclub Audit

Date: 30/06/2023

Date: 30/06/2023

Table of Contents

[Overview Project Audit Status:](#)

[Open Issues:](#)

[Important Notes:](#)

[Deployment Verification:](#)

[Audit Result](#)

[SnBNB.sol:](#)

[SnStakeManager.sol:](#)

Overview Project Audit Status:

Severity Level	Total Number of Issues
Informative	4
Discussion	1
Minor	2
Medium	3
Major	0
Critical	0

Open Issues:

Pending	Resolved
0	10

Confidential

Date: 30/06/2023

Audit Request:

Topic	Comments
Repo Link	https://github.com/agiledev624/synclub-contracts
Commit Id	https://github.com/agiledev624/synclub-contracts/commit/ab8e36ff7760caa635cc0197b6ebf1bcfd574b57
PRD Document	Not Shared
Last Audit	NA
Request Date	07/06/2023
Owner	Tom

Important Notes:

Notes
<ul style="list-style-type: none">• snBNB is not pausable.• Undelegate is only with reserve but delegate is with and without reserve.

Deployment Verification:

Deployment is verified

Verification Checks	Status
Repo code maps with Deployed code	Verified
Ownership Account	Multisig
Ownership Verified	Verified
Secondary Accounts	EOA and Multisig
Secondary Accounts Verified	Verified

Audit Result

SnBNB.sol:

This is the token contract for Synclub project.

SS#01	
StakeManager can be added as a role too	
Severity:	Informative
Line:	#10
Description:	<p>Stake Manager can be added as a role too similar to DEFAULT_ADMIN role.</p> <pre>1 // SPDX-License-Identifier: GPL-3.0 2 pragma solidity ^0.8.0; 3 4 import "@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol"; 5 import "@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol"; 6 7 import "./interfaces/ISnBnb.sol"; 8 9 contract SnBnb is ISnBnb, ERC20Upgradeable, AccessControlUpgradeable { 10 address private stakeManager; 11 12 /// @custom:oz-upgrades-unsafe-allow constructor 13 constructor() { 14 _disableInitializers(); 15 } 16 17 function initialize(address _admin) external override initializer { 18 __AccessControl_init(); 19 __ERC20_init("Synclub BNB", "SnBNB"); 20 21 require(_admin != address(0), "zero address provided"); 22 } 23 }</pre>
Suggestion:	Add in stake manager as a role
Status:	Confirmed
Team Comments:	We will keep this logic

SnStakeManager.sol:

This is the staking contract for Synclub project.

SS#02	
SafeMath is not required as the compiler is being used above 0.8.0	
Severity:	Informative
Line:	#10
Description:	<p>SafeMath is not required as the compiler is being used above 0.8.0</p> <pre> 1 //SPDX-License-Identifier: GPL-3.0 2 pragma solidity ^0.8.0; 3 4 import "@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol"; 5 import "@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol"; 6 import "@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol"; 7 import "@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol"; 8 import "@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol"; 9 import "@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol"; 10 import "@openzeppelin/contracts/utils/math/SafeMath.sol"; 11 12 import {IStakeManager} from "./interfaces/IStakeManager.sol"; 13 import {ISnBnb} from "./interfaces/ISnBnb.sol"; 14 import {IStaking} from "./interfaces/INativeStaking.sol"; 15 16 /** 17 * @title Stake Manager Contract 18 * @dev Handles Staking of BNB on BSC 19 */ 20 contract SnStakeManager is 21 IStakeManager, 22 Initializable,</pre>
Suggestion:	Remove SafeMath as it is not required
Status:	<p>Resolved</p> <p>https://github.com/agiledev624/synclub-contracts/commit/622e1e44fb2e9f3adcbdad158e50e0203653b169</p>

SS#03	
Additional event is not required.	
Severity:	Informative
Function:	Initialize

Line:	#107
Description:	<p>When setupRole is called it already emits an event for "RoleGranted"</p> <pre> 98 _setupRole(bot, _bot); 99 100 manager = _manager; 101 snBnb = _snBnb; 102 bcValidator = _validator; 103 synFee = _synFee; 104 revenuePool = _revenuePool; 105 106 emit SetManager(_manager); 107 emit SetBotRole(_bot); 108 emit SetBCValidator(bcValidator); 109 emit SetRevenuePool(revenuePool); 110 emit SetSynFee(_synFee); 111 } 112 113 /** 114 * @dev Allows user to deposit Bnb at BSC and mints SnBnb for the user 115 */ 116 function deposit() external payable override whenNotPaused { 117 uint256 amount = msg.value; 118 require(amount > 0, "Invalid Amount"); 119 120 uint256 snBnbToMint = revenuePoolToSnBnb(amount); </pre>
Suggestion:	No requirement for an additional event. This can be removed.
Status:	<p>Resolved</p> <p>https://github.com/agiledev624/synclub-contracts/commit/622e1e44fb2e9f3adcbdad158e50e0203653b169</p>

SS#04	
No event emitted on deposit	
Severity:	Informative
Function:	Deposit
Line:	#116
Description:	No event is emitted on call to deposit

Date: 30/06/2023

	<pre>107 emit SetBotRole(_bot); 108 emit SetBCValidator(bcValidator); 109 emit SetRevenuePool(revenuePool); 110 emit SetSynFee(_synFee); 111 } 112 113 /** 114 * @dev Allows user to deposit Bnb at BSC and mints SnBnb for the user 115 */ 116 function deposit() external payable override whenNotPaused { 117 uint256 amount = msg.value; 118 require(amount > 0, "Invalid Amount"); 119 120 uint256 snBnbToMint = convertBnbToSnBnb(amount); 121 122 amountToDelegate += amount; 123 124 ISnBnb(snBnb).mint(msg.sender, snBnbToMint); 125 } 126 127 /** 128 * @dev Allows bot to delegate users' funds to native staking contract without reserved BNB</pre>
Suggestion:	Do make sure that important events are emitted on deposit
Status:	Resolved https://github.com/agiledev624/synclub-contracts/commit/622e1e44fb2e9f3adcbdad158e50e0203653b169

SS#05	
Bot can pass bigger value for msg.value	
Severity:	Minor
Function:	Delegate, DelegateWithReserve, Redelgate
Line:	#132, #161, #181
Description:	What is the need to pass the whole msg.value to the the function call?

	<pre> function delegate() external payable override whenNotPaused onlyRole(BOT) returns (uint256 _amount) { uint256 relayFee = IStaking(NATIVE_STAKING).getRelayerFee(); uint256 relayFeeReceived = msg.value; _amount = amountToDelegate - (amountToDelegate % TEN_DECIMALS); require(relayFeeReceived >= relayFee, "Insufficient RelayFee"); require(_amount >= IStaking(NATIVE_STAKING).getMinDelegation(), "Insufficient Deposit Amount"); amountToDelegate = amountToDelegate - _amount; totalDelegated += _amount; // delegate through native staking contract IStaking(NATIVE_STAKING).delegate{value: _amount + msg.value}(bcValidator, _amount); emit Delegate(_amount); } </pre>
Suggestion:	Do make sure that only the relay fee is passed across and rest is returned to the caller
Status:	<p>Resolved</p> <p>https://github.com/agiledev624/synclub-contracts/commit/0644fcf89a0680507d4ee2bc43af40867d883fcd</p>

SS#06	
Usability - how would the user remember on if the last is moved to the claimed one, Is it maintained on FE?	
Severity:	Discussion
Function:	claimWithdraw
Line:	#281
Description:	<p>https://github.com/agiledev624/synclub-contracts/blob/master/contracts/SnStakeManager.sol#L281</p> <p>Is the list maintained on FE?</p>

	<pre> function claimWithdraw(uint256 _idx) external override whenNotPaused { address user = msg.sender; WithdrawalRequest[] storage userRequests = userWithdrawalRequests[user]; require(_idx < userRequests.length, "Invalid index"); WithdrawalRequest storage withdrawRequest = userRequests[_idx]; uint256 uuid = withdrawRequest.uuid; uint256 amountInSnBnb = withdrawRequest.amountInSnBnb; BotUndelegateRequest storage botUndelegateRequest = uuidToBotUndelegateRequestMap[uuid]; require(botUndelegateRequest.endTime != 0, "Not able to claim yet"); userRequests[_idx] = userRequests[userRequests.length - 1]; userRequests.pop(); uint256 totalBnbToWithdraw_ = botUndelegateRequest.amount; uint256 totalSnBnbToBurn_ = botUndelegateRequest.amountInSnBnb; uint256 amount = (totalBnbToWithdraw_ * amountInSnBnb) / totalSnBnbToBurn_; AddressUpgradeable.sendValue payable(user), amount); emit ClaimWithdrawal(user, _idx, amount); } </pre>
Suggestion :	Do check on the logic
Status:	Confirmed
Team Comments:	It's maintained on FE

SS#07	
claimFailedDelegation perform the same operation as claimUndelegated without the state updates	
Severity:	Minor
Function:	claimFailedDelegation
Line:	
Description:	When setupRole is called it already emits an event for "RoleGranted"
Suggestion:	Instead of putting this function under bot, it can be put under manager. Also for additional security, it is recommended that in case of failedDelegation the contract is paused and only then this is called to map the states back.

Status:	Confirmed
Team Comments:	This is monitored at our end.

SS#08	
Circumventing the admin for the role	
Severity:	Medium
Function:	setBotRole
Line:	#413
Description:	<p>setupRole is used under the function to set the role. "setupRole" bypass the admin check for the role which was set in constructor.</p> <pre>); manager = proposedManager; proposedManager = address(0); emit SetManager(manager); } 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 wit function setBCValidator(address _address) </pre>
Suggestion:	<p>Either make the manager as admin for role "BOT" or use Default Admin to call setBotRole and under the function, call for "grantRole" instead of "setupRole" to check for role admin.</p> <p>Do check the documentation for proper implementation of access control: https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/release-v4.0/contracts/access/AccessControlUpgradeable.sol </p>

	Apart from that event is already emitted with access control, there won't be a need to emit an extra event for this.
Status:	Resolved https://github.com/agiledev624/synclub-contracts/commit/f6f98728e6ae93a100859741ec98dbf2d93d265e

SS#09	
Circumventing the admin for the role	
Severity:	Medium
Function:	revokeBotRole
Line:	#413
Description:	<p><code>_revokeRole</code> is used under the function to revoke the role. “<code>revokeRole</code>” bypass the admin check for the role which was set in constructor.</p> <pre>); manager = proposedManager; proposedManager = address(0); emit SetManager(manager); } 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 wit function setBCValidator(address _address) </pre>
Suggestion:	<p>Either make the manager as admin for role “BOT” or use Default Admin to call <code>setBotRole</code> and under the function, call for “<code>revokeRole</code>” instead of “<code>_revokeRole</code>” to check for role admin.</p> <p>Do check the documentation for proper implementation of access control:</p>

	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/release-v4.0/contracts/access/AccessControlUpgradeable.sol Apart from that event is already emitted with access control, there won't be a need to emit an extra event for this.
Status:	Resolved https://github.com/agiledev624/synclub-contracts/commit/f6f98728e6ae93a100859741ec98dbf2d93d265e

SS#10	
availableReserveAmount is not updated on “delegateWithReserve” and “undelegate”	
Severity:	Medium
Function:	delegateWithReserve, undelegate
Line:	
Description:	<p>If availableReservedAmount is higher then reserveAmount, this function will never fail. For example if availableReservedAmount is set to 2 BNB and reserveAmount to 1 BNB, then on each call reserveAmount will be less than availableReservedAmount, so even if availableReservedAmount is set 2 BNB, delegateWithReserve can run indefinitely with it.</p> <pre> 161 function delegateWithReserve() 162 external 163 payable 164 override 165 whenNotPaused 166 onlyRole(BOT) 167 returns (uint256 _amount) 168 { 169 uint256 relayFee = IStaking(NATIVE_STAKING).getRelayerFee(); 170 uint256 relayFeeReceived = msg.value; 171 _amount = amountToDelegate - (amountToDelegate % TEN_DECIMALS); 172 173 require(relayFeeReceived >= relayFee, "Insufficient RelayFee"); 174 require(availableReserveAmount >= reserveAmount, "Insufficient Reserve Amount"); 175 require(_amount + reserveAmount >= IStaking(NATIVE_STAKING).getMinDelegation(), "Ins 176 177 amountToDelegate = amountToDelegate - _amount; 178 totalDelegated += _amount; 179 180 // delegate through native staking contract 181 IStaking(NATIVE_STAKING).delegate{value: _amount + msg.value + reserveAmount}(bcVali 182 183 emit Delegate(_amount); 184 emit DelegateReserve(reserveAmount); 185 } 186 187 function redelegate(address srcValidator, address dstValidator, uint256 amount) </pre>

Date: 30/06/2023

Suggestion:	<p>Consider adding in ($\text{availableReserveAmount} = \text{availableReserveAmount} - \text{reserveAmount}$)</p> <p>To make sure that only up-to the availableReserveAmount, the function is able to delegate</p>
Status:	Confirmed
Team Comments:	<p>The controls for this is added on the Bot end and not at contract end.</p> <p>In addition to that the deposit and withdrawl amount has to be same, so it won't affect the users. The checks external to the contract code and it's recommended to audit the same.</p>