# SCRL

# Full Audit Report

## NBD Staking Security Assessment

NBD Staking Security Assessment

## FULL AUDIT REPORT

Security Assessment by SCRL on **Saturday, April 13, 2024**

SCRL is deliver a security solution for Web3 projects by expert security researchers.

SCRL

## Executive Summary

For this security assessment, SCRL received a request on Monday, April 1, 2024

| Client | Language | Audit Method | Confidential | Network Chain | Contract |
|---|---|---|---|---|---|
| **Never Back Down (Staking Contract)** | **Solidity** | **Whitebox** | **Public** | **BNB Chain** | 0x1F976D588300c12f79751145AC3290fb07544eB0 |

| Report Version | Twitter | Telegram | Website |
|---|---|---|---|
| **1.1** | https://x.com/NbdToken | https://t.me/NBDToken | https://neverbackdown.space/ |

**Scoring:**

| Scoring | | | | | | | |
|---|---|---|---|---|---|---|---|
| 8.6 | 8.8 | 9 | 9.2 | 9.4 | 9.6 | 9.8 | 10 |

## Vulnerability Summary

| 10 | 4 | 6 | 0 | 4 | 0 |
|---|---|---|---|---|---|
| Total Findings | Unresolved | Resolved | Mitigate | Acknowledge | Decline |

- **0 Critical** — Critical severity is assigned to security vulnerabilities that pose a severe threat to the smart contract and the entire blockchain ecosystem.

- **2 High** — 2 Resolved — High-severity issues should be addressed quickly to reduce the risk of exploitation and protect users' funds and data.

- **3 Medium** — 2 Resolved, 1 Unresolved — It's essential to fix medium-severity issues in a reasonable timeframe to enhance the overall security of the smart contract.

- **1 Low** — 1 Resolved — While low-severity issues can be less urgent, it's still advisable to address them to improve the overall security posture of the smart contract.

- **0 Very Low** — Very Low severity is used for minor security concerns that have minimal impact and are generally of low risk.

- **0 Informational** — Used to categorize security findings that do not pose a direct security threat to the smart contract or its users. Instead, these findings provide additional information, recommendations

- **4 Gas-optimization** — 1 Resolved, 3 Unresolved — Suggestions for more efficient algorithms or improvements in gas usage, even if the current code is already secure.

## Audit Scope:

| File | SHA-1 Hash |
| --- | --- |
| StakingNBD.sol | cfa42662fd898a601b3a880d5c88867d37616d3f |

## Audit Version History:

| Version | Date | Description |
| --- | --- | --- |
| 1.0 | Friday, April 5, 2024 | Preliminary Report |
| 1.1 | Saturday, 13 April R 2024 | Full Audit Report with re-assessment |

## Audit information:

| Request Date | Audit Date | Re-assessment Date |
| --- | --- | --- |
| Monday, April 1, 2024 | Friday, April 5, 2024 | Saturday, April 13, 2024 |

## Smart Contract Audit Summary



## Security Assessment Author

|  |  |  |
| --- | --- | --- |
| Auditor: | **Mark K.** | [Security Researcher \| Redteam] |
|  | **Kevin N.** | [Security Researcher \| Web3 Dev] |
|  | **Yusheng T.** | [Security Researcher \| Incident Response] |
| Document Approval: | **Ronny C.** | CTO & Head of Security Researcher |
|  | **Chinnakit J.** | CEO & Founder |

## Digital Sign

# Disclaimer

Regarding this security assessment, there are no guarantees about the security of the program instruction received from the client is hereinafter referred to as "**Source code**".

And **SCRL** hereinafter referred to as "**Service Provider**", the **Service Provider** will not be held liable for any legal liability arising from errors in the security assessment. The responsibility will be the responsibility of the **Client**, hereinafter referred to as "**Service User**" and the

**Service User** agrees not to be held liable to the **service provider** in any case. By contract

**Service Provider** to conduct security assessments with integrity with professional ethics, and transparency to deliver security assessments to users The **Service Provider** has the right to postpone the delivery of the security assessment. If the security assessment is delayed whether caused by any reason and is not responsible for any delayed security assessments.

If **the service provider** finds a vulnerability The **service provider** will notify the **service user** via the Preliminary Report, which will be kept confidential for security. The **service provider** disclaims responsibility in the event of any attacks occurring whether before conducting a security assessment. Or happened later All responsibility shall be sole with the **service user**.

<mark>**Security Assessment Is Not Financial/Investment Advice Any loss arising from any investment in any project is the responsibility of the investor.**</mark>

**SCRL disclaims any liability incurred. Whether it's Rugpull, Abandonment, Soft Rugpull, Exploit, Exit Scam.**

# Security Assessment Procedure

1. **Request** The client must submit a formal request and follow the procedure. By submitting the source code and agreeing to the terms of service.
2. **Audit Process** Check for vulnerabilities and vulnerabilities from source code obtained by experts using formal verification methods, including using powerful tools such as Static Analysis, SWC Registry, Dynamic Security Analysis, Automated Security Tools, CWE, Syntax & Parameter Check with AI ,WAS (Warning Avoidance System a python script tools powered by SCRL).
3. **Security Assessment** Deliver Preliminary Security Assessment to clients to acknowledge the risks and vulnerabilities.
4. **Consulting** Discuss on risks and vulnerabilities encountered by clients to apply to their source code to mitigate risks.
   a. **Re-assessment** Reassess the security when the client implements the source code improvements and if the client is satisfied with the results of the audit. We will proceed to the next step.
5. **Full Audit Report** SCRL provides clients with official security assessment reports informing them of risks and vulnerabilities. Officially and it is assumed that the client has been informed of all the information.

## Risk Rating

Risk rating using this commonly defined: $Risk\ rating\ =\ impact\ *\ confidence$

**Impact** The severity and potential impact of an attacker attack
**Confidence** Ensuring that attackers expose and use this vulnerability

| Confidence<br><br>Impact [Likelihood] | Low | Medium | High |
|---|---|---|---|
| Low | Very Low | Low | Medium |
| Medium | Low | Medium | High |
| High | Medium | High | Critical |

**Severity** is a risk assessment It is calculated from the Impact and Confidence values using the following calculation methods,

$Risk\ rating\ =\ impact\ *\ confidence$

It is categorized into

**7 categories severity based**

| Gas-optimization | Informational | Very Low | Low | Medium | High | Critical |
|---|---|---|---|---|---|---|
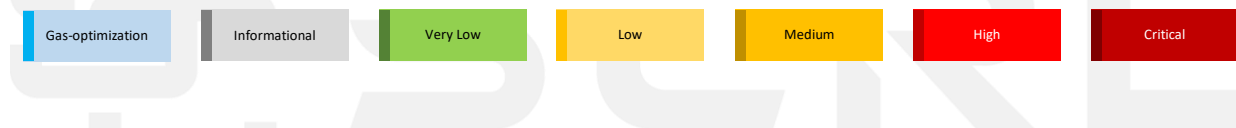
For **Informational** & **Non-class/Optimization/Best-practices will** <u>not be counted</u> as **severity**

## Category

| Centralization | Economics Risk | Logical Issue | Authorization | Mathematical | Naming Conventions |
|---|---|---|---|---|---|
| **Centralization Risk** is The risk incurred by a sole proprietor, such as the Owner being able to change something without permission | **Economics Risk** is Risks that may affect the economic mechanism system, such as the ability to increase Mint token | **Logical Issue** is that can cause errors to core processing, such as any prior operations that cause background processes to crash. | **Authorization** is Possible pitfalls from weak coding allows unrelated people to take any action to modify the values. | **Mathematical** Any erroneous arithmetic operations affect the operation of the system or lead to erroneous values. | **Naming Conventions** naming variables that may affect code understanding or naming inconsistencies |

| Security Risk | Coding Style | Best Practices | Optimization | Gas Optimization | Dead Code |
|---|---|---|---|---|---|
| **Security Risk** of loss or damage if it's no mitigate | **Coding Style** is Tips coding for efficiency performance | **Best Practices** is suggestions for improvement | **Optimization** is performance improvement | **Gas Optimization** is increase performance to avoid expensive gas | **Dead Code** having unused code This may result in wasted resources and gas fees. |

# Table Of Content

## Source Units in Scope

Source Units Analyzed: **1**
Source Units in Scope: **1** (**100%**)

| Type | File | Logic Contracts | Interfaces | Lines | nLines | nSLOC | Comment Lines | Complex. Score | Capabilities |
|---|---|---|---|---|---|---|---|---|---|
| 📝🔍 | src/StakingNBD.sol | 2 | 2 | 346 | 313 | 246 | 3 | 206 | ⏏ |
| 📝🔍 | **Totals** | **2** | **2** | **346** | **313** | **246** | **3** | **206** | ⏏ |

Legend: [ — ]

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

# Visibility, Mutability, Modifier function testing

## Components

| 📝 Contracts | 📚 Libraries | 🔍 Interfaces | 🎨 Abstract |
|---|---|---|---|
| 2 | 0 | 2 | 1 |

## Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

| 🌐 Public | 💰 Payable |
|---|---|
| 26 | 0 |

| External | Internal | Private | Pure | View |
|---|---|---|---|---|
| 7 | 13 | 0 | 0 | 20 |

## StateVariables

| Total | 🌐 Public |
|---|---|
| 23 | 14 |

## Capabilities

| Solidity Versions observed | 🧪 Experimental Features | 💰 Can Receive Funds | 📋 Uses Assembly | 💣 Has Destroyable Contracts |
|---|---|---|---|---|
| 0.8.24 | | | | |

| 📤 Transfers ETH | ⚡ Low-Level Calls | 👥 DelegateCall | 🖼️ Uses Hash Functions | 🏷️ ECRecover | 🔄 New/Create/Create2 |
|---|---|---|---|---|---|
| yes | | | | | |

♻️ **TryCatch** | **Σ Unchecked**

# Vulnerability Findings

| ID | Vulnerability Detail | Severity | Category | Status |
|---|---|---|---|---|
| SEC-01 | Unchecked tokens transfer (unchecked-transfer) | High | Best Practices | Resolved |
| SEC-02 | Fees are still counting at usdValue | High | Logical Issue | Resolved |
| SEC-03 | Reentrancy vulnerabilities (no theft of ethers) (reentrancy-no-eth) | Medium | Best Practices | Resolved |
| SEC-04 | Imprecise arithmetic operations order (divide-before-multiply) | Medium | Mathematical | Acknowledge |
| SEC-05 | Tautology or contradiction (tautology) | Medium | Best Practices | Resolved |
| SEC-06 | Missing Events Arithmetic (events-maths) | Low | Best Practices | Resolved |
| GAS-01 | Cache array length outside of loop | Gas-optimization | Gas Optimization | Resolved |
| GAS-02 | For Operations that will not overflow, you could use unchecked | Gas-optimization | Gas Optimization | Acknowledge |
| GAS-03 | Use Custom Errors | Gas-optimization | Gas Optimization | Acknowledge |
| GAS-04 | `++i` costs less gas than `i++`, especially when it's used in `for`-loops (`--i`/`i--` too) | Gas-optimization | Gas Optimization | Acknowledge |

## SEC-01:    Unchecked tokens transfer (unchecked-transfer)

| Vulnerability Detail | Severity | Location | Category | Status |
|---|---|---|---|---|
| Unchecked tokens transfer (unchecked-transfer) | High | Check on finding | Best Practices | Resolved |

**Finding:**

❌ Staking.claimReward(bool) (src/StakingNBD.sol:260-285) ignores return value by nbdToken.transfer(msg.sender,sameValueNbdAmount) (src/StakingNBD.sol#283)
❌ Staking.stakeTokens(uint256,uint256) (src/StakingNBD.sol:136-168) ignores return value by nbdToken.transferFrom(msg.sender,address(this),amount) (src/StakingNBD.sol#162)
❌ Staking.stakeTokens(uint256,uint256) (src/StakingNBD.sol:136-168) ignores return value by nbdToken.transferFrom(msg.sender,myreferer,refFeeAmount) (src/StakingNBD.sol#159)
❌ Staking.stakeTokens(uint256,uint256) (src/StakingNBD.sol:136-168) ignores return value by nbdToken.transferFrom(msg.sender,owner(),stakeFeeAmount) (src/StakingNBD.sol#153)

**Recommendation:**
Use `SafeERC20`, or ensure that the transfer/transferFrom return value is checked.

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

**Alleviation: NBD Team has resolved this issue.**

## SEC-02:    Fees are still counting at usdValue

| Vulnerability Detail | Severity | Location | Category | Status |
|---|---|---|---|---|
| Fees are still counting at **usdValue** | High | Check on finding | Logical Issue | **Resolved** |

**Finding:**

❌ Staking.claimReward(bool) (src/StakingNBD.sol:136-168):
  • uint256 usdValue = nbdToUsd(amount) (src/StakingNBD.sol#149)

**Recommendation:**

Move [uint256 usdValue = nbdToUsd(amount);] to under line 161

**Alleviation: NBD Team has resolved this issue.**

## SEC-03:    Reentrancy vulnerabilities (no theft of ethers) (reentrancy-no-eth)

| Vulnerability Detail | Severity | Location | Category | Status |
|---|---|---|---|---|
| Reentrancy vulnerabilities (no theft of ethers)<br><br>(reentrancy-no-eth) | Medium | Check on finding | Best Practices | Resolved |

**Finding:**

❌ Reentrancy in Staking.stakeTokens(uint256,uint256) (src/StakingNBD.sol:136-168):
- nbdToken.transferFrom(msg.sender,owner(),stakeFeeAmount) (src/StakingNBD.sol#153)
- nbdToken.transferFrom(msg.sender,myreferer,refFeeAmount) (src/StakingNBD.sol#159)
- nbdToken.transferFrom(msg.sender,address(this),amount) (src/StakingNBD.sol#162)
- stakeRecord[msg.sender].push(Stake(usdValue,0,block.timestamp)) (src/StakingNBD.sol#164)
- Staking.calculateReward(address,uint256) (src/StakingNBD.sol#242-255)
- Staking.claimReward(bool) (src/StakingNBD.sol#260-285)
- Staking.countTotalStakes(address) (src/StakingNBD.sol#235-238)
- Staking.getAvailableReward(address) (src/StakingNBD.sol#185-195)
- Staking.getPeriods(address,uint256) (src/StakingNBD.sol#214-220)
- Staking.getRewardIssued(address,uint256) (src/StakingNBD.sol#228-233)
- Staking.getStakeInfo(address,uint256) (src/StakingNBD.sol#222-225)
- Staking.getTotalRewardClaimed(address) (src/StakingNBD.sol#198-211)
- Staking.reInvestTokens(uint256) (src/StakingNBD.sol#291-301)
- Staking.stakeRecord (src/StakingNBD.sol#76)
- Staking.stakeTokens(uint256,uint256) (src/StakingNBD.sol#136-168)

**Recommendation:**
Apply the [`check-effects-interactions` pattern](http://solidity.readthedocs.io/en/v0.4.21/security-considerations.html#re-entrancy).

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

**Alleviation: NBD Team has resolved this issue.**

## SEC-04: Imprecise arithmetic operations order (divide-before-multiply)

| Vulnerability Detail | Severity | Location | Category | Status |
|---|---|---|---|---|
| Imprecise arithmetic operations order (divide-before-multiply) | Medium | Check on finding | Mathematical | Acknowledge |

**Finding:**

❌ Staking.calculateReward(address,uint256) (src/StakingNBD.sol:242-255) performs a multiplication on the result of a division:
- rewardPerPeriod = stakedAmount * dailyPercentage / 1000 (src/StakingNBD.sol#245)
- calculatedReward = rewardPerPeriod * periods (src/StakingNBD.sol#247)

**Recommendation:**

Consider ordering multiplication before division.

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

# SEC-05: Tautology or contradiction (tautology)

| Vulnerability Detail | Severity | Location | Category | Status |
|---|---|---|---|---|
| Tautology or contradiction (tautology) | Medium | Check on finding | Best Practices | Resolved |

**Finding:**

❌ Staking.getGenInfo() (src/StakingNBD.sol:172-182) contains a tautology or contradiction:
  • _totalStakedAmount < 0 (src/StakingNBD.sol#176)

**Recommendation:**

Fix the incorrect comparison by changing the value type or the comparison.

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-contradiction

**Alleviation: NBD Team has resolved this issue.**

## SEC-06:     Missing Events Arithmetic (events-maths)

| Vulnerability Detail | Severity | Location | Category | Status |
|---|---|---|---|---|
| Missing Events Arithmetic (events-maths) | Low | Check on finding | Best Practices | Resolved |

**Finding:**

❌ Staking.updateDailyPercentage(uint256) (src/StakingNBD.sol:128-132) should emit an event for:
  • dailyPercentage = _newValue (src/StakingNBD.sol#130)

**Recommendation:**
Emit an event for critical parameter changes.

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

**Alleviation: NBD Team has resolved this issue.**

## GAS-01:  Cache array length outside of loop

| Vulnerability Detail | Severity | Location | Category | Status |
|---|---|---|---|---|
| Cache array length outside of loop | - | Check on finding | Gas Optimization | **Resolved** |

### Finding:

```
File: StakingNBD.sol


86:        for(uint256 i=0; i<stakers.length; i++)
```

### Recommendation:

If not cached, the solidity compiler will always read the length of the array during each iteration. That is, if it is a storage array, this is an extra sload operation (100 additional extra gas for each iteration except for the first) and if it is a memory array, this is an extra mload operation (3 additional gas for each iteration except for the first).

### Alleviation:

**NBD Team has resolved this issue.**

## GAS-02:    For Operations that will not overflow, you could use unchecked

| Vulnerability Detail | Severity | Location | Category | Status |
|---|---|---|---|---|
| For Operations that will not overflow, you could use unchecked | - | Check on finding | Gas Optimization | Acknowledge |

### Finding:

```
File: StakingNBD.sol


86:        for(uint256 i=0; i<stakers.length; i++)


86:        for(uint256 i=0; i<stakers.length; i++)


88:            if(codeToAddress[_referrelCode]==addr) { count++; }


88:            if(codeToAddress[_referrelCode]==addr) { count++; }


99:    uint256 public dailyPercentage = 10; // 1%


99:    uint256 public dailyPercentage = 10; // 1%


139:          referrelSerial++;


139:          referrelSerial++;


152:      uint256 stakeFeeAmount = amount*stakingFee/100;


152:      uint256 stakeFeeAmount = amount*stakingFee/100;


155:      uint256 refFeeAmount = amount*stakingFee/100;


155:      uint256 refFeeAmount = amount*stakingFee/100;
```

```
158:        refReward[myreferer] += refFeeAmount;

161:        amount = amount-stakeFeeAmount-refFeeAmount;

161:        amount = amount-stakeFeeAmount-refFeeAmount;

166:        totalStakedAmount += usdValue;

189:        for(uint256 i = 0; i<len; i++)

189:        for(uint256 i = 0; i<len; i++)

191:           rewardAvailable += calculateReward(addr, i);

205:        for(uint256 i = 0; i<len; i++)

205:        for(uint256 i = 0; i<len; i++)

207:           _issuedReward += stakeRecord[addr][i].issuedReward;

208:           _stakedAmount  += stakeRecord[addr][i].stakedAmount;

217:        uint256 span = block.timestamp - stakingTime;

218:        uint256 periods = span/stakingPeriod;

230:        uint256 total =  stakeRecord[addr][index].stakedAmount*3;

245:        uint256 rewardPerPeriod =  stakedAmount*dailyPercentage/1000;

245:        uint256 rewardPerPeriod =  stakedAmount*dailyPercentage/1000;

247:        uint256 calculatedReward = rewardPerPeriod*periods;
```

```
248:      if(calculatedReward>stakedAmount*3)

250:          calculatedReward = stakedAmount*3;

253:      uint256 _reward = calculatedReward-alreadyIssuedReward;

267:      for(uint256 i=0; i<len; i++)

267:      for(uint256 i=0; i<len; i++)

271:          totalReward += _reward;

272:          stakeRecord[msg.sender][i].issuedReward += _reward;

275:      totalIssuedReward += totalReward;

300:      totalStakedAmount += usdValue;
```

## Recommendation:
-

## Alleviation:
-

## GAS-03:     Use Custom Errors

| Vulnerability Detail | Severity | Location | Category | Status |
|---|---|---|---|---|
| Use Custom Errors | - | Check on finding | Gas Optimization | Acknowledge |

### Finding:

```
File: StakingNBD.sol


36:        require(_owner == _msgSender(), "Ownable: caller is not the owner");


48:        require(newOwner != address(0), "Ownable: new owner is the zero address");


131:        require(dailyPercentage>=5 && dailyPercentage<=50, "Not in range");
```

### Recommendation:

Instead of using error strings, to reduce deployment and runtime cost, you should use Custom Errors. This would save both deployment and runtime cost.

Reference: https://blog.soliditylang.org/2021/04/21/custom-errors/

### Alleviation:

-

## GAS-04:    `++i` costs less gas than `i++`, especially when it's used in `for`-loops (`--i`/`i--` too)

| Vulnerability Detail | Severity | Location | Category | Status |
|---|---|---|---|---|
| `++i` costs less gas than `i++`, especially when it's used in `for`-loops (`--i`/`i--` too) | - | Check on finding | Gas Optimization | Acknowledge |

## Finding:

```
File: StakingNBD.sol

86:        for(uint256 i=0; i<stakers.length; i++)

88:          if(codeToAddress[_referrelCode]==addr) { count++; }

139:           referrelSerial++;

189:        for(uint256 i = 0; i<len; i++)

205:        for(uint256 i = 0; i<len; i++)

267:        for(uint256 i=0; i<len; i++)
```

## Recommendation:

-

## Alleviation:

-

## SWC Findings

| ID | Title | Scanning | Result |
|----|-------|----------|--------|
| SWC-100 | Function Default Visibility | Complete | No risk |
| SWC-101 | Integer Overflow and Underflow | Complete | No risk |
| SWC-102 | Outdated Compiler Version | Complete | No risk |
| SWC-103 | Floating Pragma | Complete | No risk |
| SWC-104 | Unchecked Call Return Value | Complete | No risk |
| SWC-105 | Unprotected Ether Withdrawal | Complete | No risk |
| SWC-106 | Unprotected SELFDESTRUCT Instruction | Complete | No risk |
| SWC-107 | Reentrancy | Complete | No risk |
| SWC-108 | State Variable Default Visibility | Complete | No risk |
| SWC-109 | Uninitialized Storage Pointer | Complete | No risk |
| SWC-110 | Assert Violation | Complete | No risk |
| SWC-111 | Use of Deprecated Solidity Functions | Complete | No risk |
| SWC-112 | Delegatecall to Untrusted Callee | Complete | No risk |
| SWC-113 | DoS with Failed Call | Complete | No risk |
| SWC-114 | Transaction Order Dependence | Complete | No risk |
| SWC-115 | Authorization through tx.origin | Complete | No risk |

| SWC-116 | Block values as a proxy for time | Complete | No risk |
|---|---|---|---|
| SWC-117 | Signature Malleability | Complete | No risk |
| SWC-118 | Incorrect Constructor Name | Complete | No risk |
| SWC-119 | Shadowing State Variables | Complete | No risk |
| SWC-120 | Weak Sources of Randomness from Chain Attributes | Complete | No risk |
| SWC-121 | Missing Protection against Signature Replay Attacks | Complete | No risk |
| SWC-122 | Lack of Proper Signature Verification | Complete | No risk |
| SWC-123 | Requirement Violation | Complete | No risk |
| SWC-124 | Write to Arbitrary Storage Location | Complete | No risk |
| SWC-125 | Incorrect Inheritance Order | Complete | No risk |
| SWC-126 | Insufficient Gas Griefing | Complete | No risk |
| SWC-127 | Arbitrary Jump with Function Type Variable | Complete | No risk |
| SWC-128 | DoS With Block Gas Limit | Complete | No risk |
| SWC-129 | Typographical Error | Complete | No risk |
| SWC-130 | Right-To-Left-Override control character (U+202E) | Complete | No risk |
| SWC-131 | Presence of unused variables | Complete | No risk |
| SWC-132 | Unexpected Ether balance | Complete | No risk |

| SWC-133 | Hash Collisions With Multiple Variable Length Arguments | Complete | No risk |
|---------|--------------------------------------------------------|----------|---------|
| SWC-134 | Message call with hardcoded gas amount | Complete | No risk |
| SWC-135 | Code With No Effects | Complete | No risk |
| SWC-136 | Unencrypted Private Data On-Chain | Complete | No risk |

Contracts Description Table

| Contract | Type | Bases | | |
|---|---|---|---|---|
| └ | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
| | | | | |
| **IERC20** | Interface | | | |
| └ | totalSupply | External ❗ | | NO❗ |
| └ | balanceOf | External ❗ | | NO❗ |
| └ | transfer | External ❗ | 🛑 | NO❗ |
| └ | allowance | External ❗ | | NO❗ |
| └ | transferFrom | External ❗ | 🛑 | NO❗ |
| | | | | |
| **Ownable** | Implementation | | | |
| └ | _msgSender | Internal 🔒 | | |
| └ | | Public ❗ | 🛑 | NO❗ |
| └ | owner | Public ❗ | | NO❗ |
| └ | renounceOwnership | Public ❗ | 🛑 | onlyOwner |
| └ | transferOwnership | Public ❗ | 🛑 | onlyOwner |
| | | | | |
| **Router** | Interface | | | |
| └ | getAmountsOut | External ❗ | | NO❗ |
| | | | | |
| **ReentrancyGuard** | Implementation | | | |
| └ | | Public ❗ | 🛑 | NO❗ |
| | | | | |
| **StakingNBD** | Implementation | Ownable, ReentrancyGuard | | |

| Contract | Type | Bases | | |
|---|---|---|---|---|
| L | | Public ❗ | 🛑 | Ownable |
| L | getUserState | Public ❗ | | NO❗ |
| L | countRef | Public ❗ | | NO❗ |
| L | updateDailyPercentage | External ❗ | 🛑 | onlyOwner |
| L | stakeTokens | Public ❗ | 🛑 | nonReentrant |
| L | getGenInfo | Public ❗ | | NO❗ |
| L | getAvailableReward | Public ❗ | | NO❗ |
| L | getTotalRewardClaimed | Public ❗ | | NO❗ |
| L | getPeriods | Public ❗ | | NO❗ |
| L | getStakeInfo | Public ❗ | | NO❗ |
| L | getRewardIssued | Public ❗ | | NO❗ |
| L | countTotalStakes | Public ❗ | | NO❗ |
| L | calculateReward | Public ❗ | | NO❗ |
| L | claimReward | Public ❗ | 🛑 | nonReentrant |
| L | reInvestTokens | Internal 🔒 | 🛑 | |
| L | nbdToUsd | Public ❗ | | NO❗ |
| L | usdToNbd | Public ❗ | | NO❗ |
| L | getSellingBnbAmount | Public ❗ | | NO❗ |
| L | getBuyingNbdAmount | Public ❗ | | NO❗ |

Legend

| Symbol | Meaning |
|---|---|
| 🛑 | Function can modify state |
| 💵 | Function is payable |

# Call Graph

# UML Class Diagram

**<<Interface>>**
**IERC20**
StakingNBD.sol

External:
- totalSupply(): uint256
- balanceOf(account: address): uint256
- transfer(recipient: address, amount: uint256): bool
- allowance(owner: address, spender: address): uint256
- transferFrom(sender: address, recipient: address, amount: uint256): bool

**Ownable**
StakingNBD.sol

Private:
- _owner: address

Internal:
- _msgSender(): (payable: address)

Public:
- <<event>> OwnershipTransferred(previousOwner: address, newOwner: address)
- <<modifier>> onlyOwner()
- constructor(addr: address)
- owner(): address
- renounceOwnership() <<onlyOwner>>
- transferOwnership(newOwner: address) <<onlyOwner>>

**<<Interface>>**
**Router**
StakingNBD.sol

External:
- getAmountsOut(amountIn: uint, path: address[]): (amounts: uint256[])

**<<Abstract>>**
**ReentrancyGuard**
StakingNBD.sol

Private:
- _NOT_ENTERED: uint256
- _ENTERED: uint256
- _status: uint256

Public:
- <<modifier>> nonReentrant()
- constructor()

**<<Struct>>**
**Stake**
StakingNBD.sol

- stakedAmount: uint256
- issuedReward: uint256
- stakingTimestamp: uint256

**StakingNBD**
StakingNBD.sol

Private:
- referrelSerial: uint256

Public:
- nbdTokenAddress: address
- routerAddress: address
- wbnbAddress: address
- usdtAddress: address
- stakers: address[]
- stakeRecord: mapping(address=>Stake[])
- referrelCode: mapping(address=>uint256)
- codeToAddress: mapping(uint256=>address)
- referer: mapping(address=>uint256)
- refReward: mapping(address=>uint256)
- stakingPeriod: uint256
- totalStakedAmount: uint256
- totalIssuedReward: uint256
- dailyPercentage: uint256
- stakingFee: uint256
- refFee: uint256
- nbdToken: IERC20
- router: Router

Internal:
- reInvestTokens(usdValue: uint256)

External:
- updateDailyPercentage(_newValue: uint256) <<onlyOwner>>

Public:
- <<event>> DailyPercentageUpdated(uint256, uint256)
- <<event>> Staked(addr: address, index: uint256, nbdAmount: uint256, usdValue: uint256)
- <<event>> Reinvested(addr: address, index: uint256, amount: uint256)
- constructor(addr: address)
- getUserState(addr: address): (data: uint256[6])
- countRef(addr: address): (data: uint256[2])
- stakeTokens(amount: uint256, _referrelCode: uint256) <<nonReentrant>>
- getGenInfo(): (uint256, uint256, uint256, uint256)
- getAvailableReward(addr: address): uint256
- getTotalRewardClaimed(addr: address): (uint256, uint256)
- getPeriods(addr: address, index: uint256): uint256
- getStakeInfo(addr: address, index: uint256): Stake
- getRewardIssued(addr: address, index: uint256): (uint256, uint256)
- countTotalStakes(addr: address): uint256
- calculateReward(addr: address, index: uint256): uint256
- claimReward(reInvest: bool) <<nonReentrant>>
- nbdToUsd(amount: uint256): uint256
- usdToNbd(amount: uint256): uint256
- getSellingBnbAmount(amount: uint256): uint256
- getBuyingNbdAmount(amount: uint256): uint256

## About SCRL

SCRL (Previously name SECURI LAB) was established in 2020, and its goal is to deliver a security solution for Web3 projects by expert security researchers. To verify the security of smart contracts, they have developed internal tools and KYC solutions for Web3 projects using industry-standard technology. SCRL was created to solve security problems for Web3 projects. They focus on technology for conciseness in security auditing. They have developed Python-based tools for their internal use called WAS and SCRL. Their goal is to drive the crypto industry in Thailand to grow with security protection technology.



**Follow Us On:**

| Website | https://scrl.io/ |
|---|---|
| Twitter | https://twitter.com/scrl_io |
| Telegram | https://t.me/scrl_io |
| Medium | https://scrl.medium.com/ |