Full Audit Report

Dogens NFT-STAKING Security Assessment

Real Cybersecurity
Protecting digital assets

Made in Thailand

SECURI LAB
(THAILAND) contact@securi-lab.com

Audit Report

FULL AUDIT REPORT

Table of Contents

1
- Report Information 2
- Disclaimer 3
- Executive Summary 4

NVD CVSS Scoring

Audit Result

- Project Introduction 5
Scope Information
Audit Information
Audit Version History
- Initial Audit Scope 6-7
- Security Assessment Procedure 8
- Risk Rating 9
- Vulnerability Severity Summary 10
- Vulnerability Findings 11-35

SWC & SEC & Non-severity level

- SWC FIndings 36-38
- Visibility, Mutability, Modifier function testing 39-44

Component, Exposed Function

StateVariables, Capabilities, Contract Descripton Table

- Inheritate Function Relation Graph 45
- UML Diagram 46
- About Securi 47

FULL AUDIT REPORT

Report Information

About Report	**Dogens NFT-STAKING Security Assessment**
Version	**v1.1**
Client	**Dogens Project**
Language	**Solidity**
Confidentiality	**Public**
Contract File	**NFT-Staking.sol** SHA-1: 919019b6ddab5dbf5ff440f33ca9c5eb11f810e6 **stToken.sol** SHA-1: e437496faff65c88f0c32d2881c6754a5e2c3e42 **This audit uses the file as the client submitted. Please check with a differential checker after the smart contract code has been deployed and verified.**
Audit Method	**Whitebox**
Security Assessment Author	**Auditor** **Mark K. [Security Researcher | Redteam]** **Kevin N. [Security Researcher | Web3 Dev]** **Yusheng T. [Security Researcher | Incident Response]** **Approve Document** **Ronny C. CTO & Head of Security Researcher** **Chinnakit J. CEO & Founder**

*Audit Method

Whitebox: SECURI LAB Team receives all source code from the client to provide the assessment.
Blackbox: SECURI LAB Team receives only bytecode from the client to provide the assessment.

Digital Sign (Only Full Audit Report)

Digitally signed by CHIN J
DN: cn=CHIN J, o=SECURI LAB - Cybersecurity | Audit | KYC, ou=CEO, email=contact@securi-lab.com, c=TH
Date: 2023.07.18 17:17:58 +07'00'

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 **SECURI Lab** 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**.

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

SECURI LAB disclaims any liability incurred. Whether it's Rugpull, Abandonment, Soft Rugpull

The SECURI LAB team has conducted a comprehensive security assessment of the vulnerabilities.
This assessment is tested with an expert assessment. Using the following test requirements

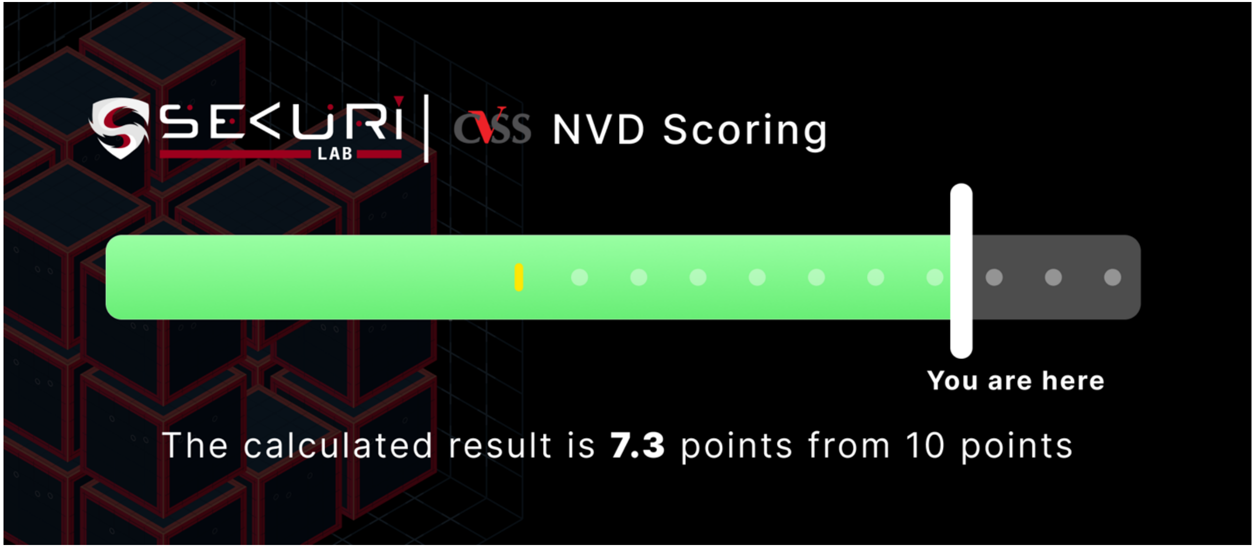
1. Smart Contract Testing with Expert Analysis By testing the most common and uncommon vulnerabilities.
2. Automated program testing It includes a sample vulnerability test and a sample of the potential vulnerabilities being used for the most frequent attacks.
3. Manual Testing with AST/WAS/ASE/SMT and reviewed code line by line
4. Visibility, Mutability, Modifier function testing, such as whether a function can be seen in general, or whether a function can be changed and if so, who can change it.
5. Function association test It will be displayed through the association graph.
6. This safety assessment is cross-checked prior to the delivery of the assessment results.

Executive Summary

For this security assessment, SECURI LAB received a request from Dogens Team on Thursday, July 06, 2023.

NVD CVSS Scoring

The score was calculated using the NVD (National Vulnerability Database) of NIST (National Institute of Standards and Technology) under the CVSS 3.1 standard, based on the CIA (Confidentiality, Integrity, Availability).

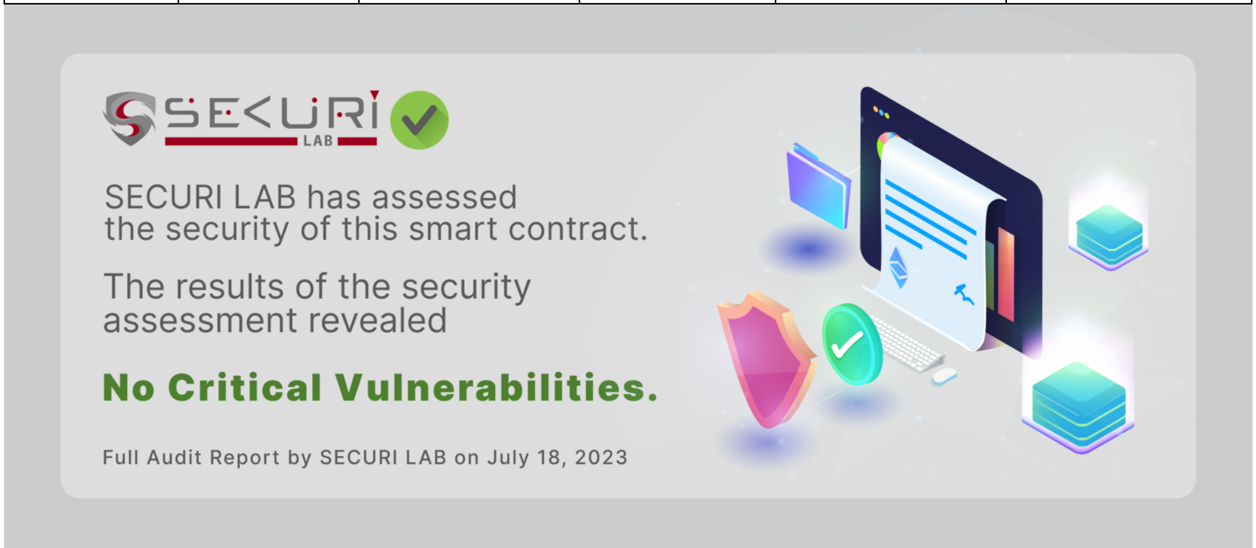


The calculated result is **7.3** points from 10 points

Audit Result

SECURI LAB evaluated the smart contract security of the project and found: **[Total : 11]**

Critical	High	Medium	Low	Very Low	Informational
0	1	1	5	0	4



SECURI LAB has assessed the security of this smart contract.

The results of the security assessment revealed

No Critical Vulnerabilities.

Full Audit Report by SECURI LAB on July 18, 2023

FULL AUDIT REPORT

Project Introduction

Scope Information:

Project Name	Dogens
Website	https://Dogens.io
Chain	-
Language	Solidity

Audit Information:

Request Date	Thursday, July 6, 2023
Audit Date	Monday, July 10, 2023
Re-assessment Date	-

Audit Version History:

Version	Date	Description
1.0	Tuesday, July 11, 2023	Preliminary Report
1.1	Tuesday, July 18, 2023	Full Audit Report

Initial Audit Scope:

Smart Contract File	**NFT-Staking.sol** SHA-1: 919019b6ddab5dbf5ff440f33ca9c5eb11f810e6 **stToken.sol** SHA-1: e437496faff65c88f0c32d2881c6754a5e2c3e42 **This audit uses the file as the client submitted. Please check with a differential checker after the smart contract code has been deployed and verified.**
Compiler Version	**^0.8.4 , ^0.8.0**

Source Units Analyzed: **2**
Source Units in Scope: **2** (**100%**)

Type	**File**	**Logic Contracts**	**Interfaces**	**Lines**	**nLines**	**nSLOC**	**Comment Lines**	**Complex. Score**	**Capabilities**
📝 🔍 🎨	contracts/stToken.sol	3	2	522	440	155	298	121	🔆Σ
📝 📚 🔍 🎨	contracts/NFT-Staking.sol	10	5	3103	2766	1358	1321	1267	🖥💰📤 🧮🔖🔆 ♻️Σ
📝 📚 🔍 🎨	**Totals**	**13**	**7**	**3625**	**3206**	**1513**	**1619**	**1388**	🖥💰📤 🧮🔖🔆 ♻️Σ

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, ...)

Description Report Files Description Table

File Name	SHA-1 Hash
contracts/stToken.sol	e437496faff65c88f0c32d2881c6754a5e2c3e42
contracts/NFT-Staking.sol	919019b6ddab5dbf5ff440f33ca9c5eb11f810e6

Security Assessment Procedure

Securi has the following procedures and regulations for conducting security assessments:

1.Request Audit Client submits a form request through the Securi channel. After receiving the request, Securi will discuss a security assessment. And drafting a contract and agreeing to sign a contract together with the Client

2.Auditing Securi performs security assessments of smart contracts obtained through automated analysis and expert manual audits.

3.Preliminary Report At this stage, Securi will deliver an initial security assessment. To report on vulnerabilities and errors found under Audit Scope will not publish preliminary reports for safety.

4.Reassessment After Securi has delivered the Preliminary Report to the Client, Securi will track the status of the vulnerability or error, which will be published to the Final Report at a later date with the following statuses:

- **a.Acknowledge** The client has been informed about errors or vulnerabilities from the security assessment.
- **b.Resolved** The client has resolved the error or vulnerability. Resolved is probably just a commit, and Securi is unable to verify that the resolved has been implemented or not.
- **c.Decline** Client has rejected the results of the security assessment on the issue.

5.Final Report Securi providing full security assessment report and public

Request Audit
Auditing
Preliminary Report
Reassessment
Final Report

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

Both have a total of 3 levels: **High**, **Medium**, **Low**. By *Informational* will not be classified as a level

Confidence Impact [Likelihood]	Low	Medium	High
Low	**Very Low**	**Low**	**Medium**
Medium	**Low**	**Medium**	**High**
High	**Medium**	**High**	**Critical**

FULL AUDIT REPORT

Vulnerability Severity Summary

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

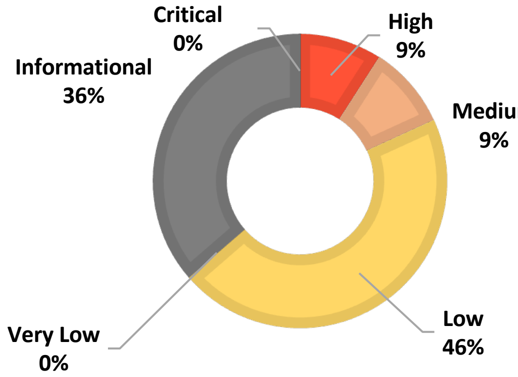
$Risk\ rating\ =\ impact\ *\ confidence$

It is categorized into

5 categories based on the **lowest severity**:

Very Low, Low, Medium, **High**, Critical.

For **Informational** & will **Non-class/Optimization/Best-practices will** not be counted as **severity**



Vulnerability Severity Level	Total
Critical	**0**
High	**1**
Medium	**1**
Low	**5**
Very Low	**0**
Informational	**4**
Non-class/Optimization/Best-practices	**0**

Category information:

Centralization

Centralization Risk is The risk incurred by a sole proprietor, such as the Owner being able to change something without permission

Economics Risk

Economics Risk is Risks that may affect the economic mechanism system, such as the ability to increase Mint token

Logical Issue

Logical Issue is that can cause errors to core processing, such as any prior operations that cause background processes to crash.

Authorization

Authorization is Possible pitfalls from weak coding allows unrelated people to take any action to modify the values.

Mathematical

Mathematical Any erroneous arithmetic operations affect the operation of the system or lead to erroneous values.

Naming Conventions

Naming Conventions naming variables that may affect code understanding or naming inconsistencies

Security Risk

Security Risk of loss or damage if it's no mitigate

Coding Style

Coding Style is Tips coding for efficiency performance

Best Practices

Best Practices is suggestions for improvement

Optimization

Optimization is performance improvement

Gas Optimization

Gas Optimization is increase performance to avoid expensive gas

Dead Code

Dead Code having unused code This may result in wasted resources and gas fees.

FULL AUDIT REPORT

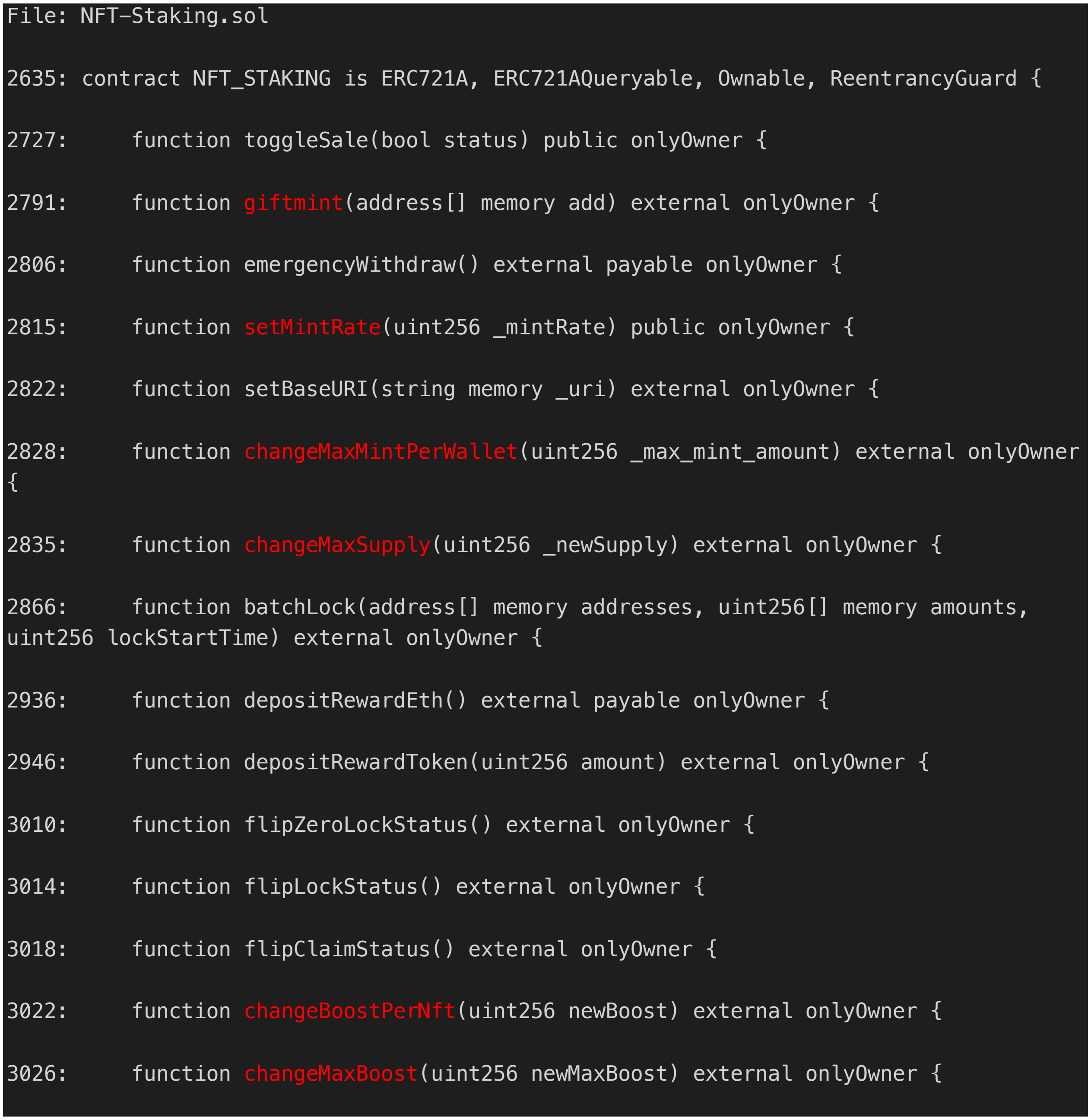
Vulnerability Findings

ID	Vulnerability Detail	Severity	Category	Status
SEC-01	Centralization Risk	High	Centralization	**Acknowledge**
SEC-02	Reentrancy vulnerabilities (no theft of ethers) (reentrancy-no-eth)	Medium	Security Risk	**Acknowledge**
SEC-03	Dangerous usage of `block.timestamp` (timestamp)	Low	Security Risk	**Acknowledge**
SEC-04	Multiple calls in a loop (calls-loop)	Low	Logical Issue	**Acknowledge**
SEC-05	Missing Events Arithmetic (events-maths)	Low	Best Practices	**Acknowledge**
SEC-06	Missing Zero Address Validation (missing-zero-check)	Low	Best Practices	**Acknowledge**
SEC-07	Reentrancy vulnerabilities leading to out-of-order Events (reentrancy-events)	Low	Best Practices	**Acknowledge**
SEC-08	Benign reentrancy vulnerabilities (reentrancy-benign)	Informational	Best Practices	**Acknowledge**
SEC-09	Missing inheritance (missing-inheritance)	Informational	Best Practices	**Acknowledge**
SEC-10	Unlocked pragma	Informational	Best Practices	**Acknowledge**
SEC-11	If different pragma directives are used (pragma)	Informational	Best Practices	**Acknowledge**

SEC-01: Centralization Risk

Vulnerability Detail	Severity	Location	Category	Status
Centralization Risk	High	Check on finding	Centralization	Acknowledge

Finding:



```
File: NFT-Staking.sol

2635: contract NFT_STAKING is ERC721A, ERC721AQueryable, Ownable, ReentrancyGuard {

2727:     function toggleSale(bool status) public onlyOwner {

2791:     function giftmint(address[] memory add) external onlyOwner {

2806:     function emergencyWithdraw() external payable onlyOwner {

2815:     function setMintRate(uint256 _mintRate) public onlyOwner {

2822:     function setBaseURI(string memory _uri) external onlyOwner {

2828:     function changeMaxMintPerWallet(uint256 _max_mint_amount) external onlyOwner
{

2835:     function changeMaxSupply(uint256 _newSupply) external onlyOwner {

2866:     function batchLock(address[] memory addresses, uint256[] memory amounts,
uint256 lockStartTime) external onlyOwner {

2936:     function depositRewardEth() external payable onlyOwner {

2946:     function depositRewardToken(uint256 amount) external onlyOwner {

3010:     function flipZeroLockStatus() external onlyOwner {

3014:     function flipLockStatus() external onlyOwner {

3018:     function flipClaimStatus() external onlyOwner {

3022:     function changeBoostPerNft(uint256 newBoost) external onlyOwner {

3026:     function changeMaxBoost(uint256 newMaxBoost) external onlyOwner {
```

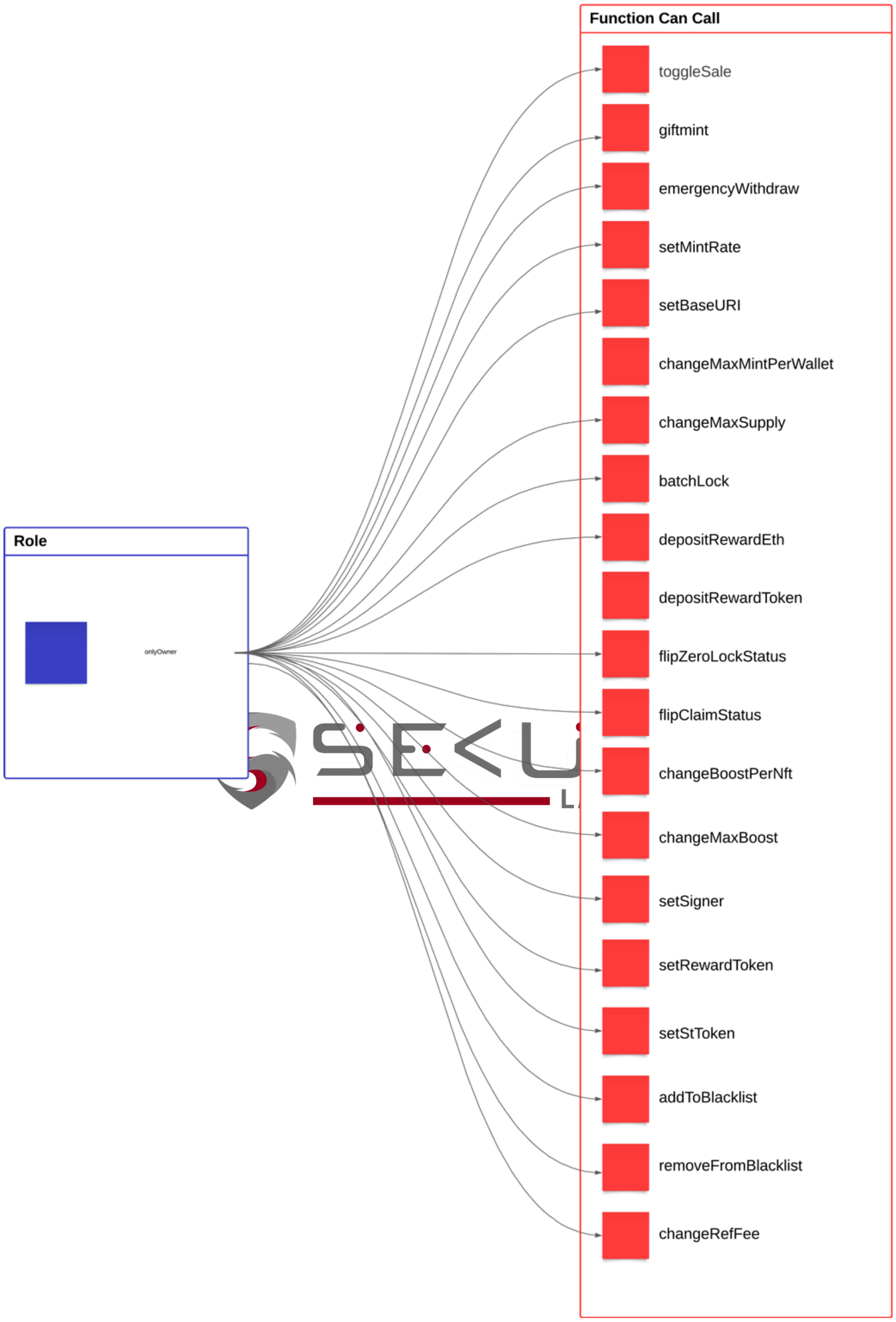
```
3030:     function setSigner(address _signer) external onlyOwner {

3034:     function setRewardToken(address _rewardToken) external onlyOwner {

3040:     function setStToken(address _stToken) external onlyOwner {

3046:     function addToBlacklist(address[] memory users) external onlyOwner {

3053:     function removeFromBlacklist(address[] memory users) external onlyOwner {

3060:     function changeRefFee(uint8 _newRefFee) external onlyOwner {

3092:     ) public onlyOwner {

` ` `
```

Scenario:

Centralized risk refers to the potential security risks that arise when a smart contract is controlled by a central entity or a single point of failure. If the contract is controlled by a central authority, then the contract may be vulnerable to attacks that target the centralized entity.

Centralized risk that can lead to rug pulls typically arises from the centralization of control or ownership of a project's assets, particularly in decentralize d finance (DeFi) projects built on blockchain platforms like Ethereum.

Contract NFT_STAKING (File: NFT-Staking.sol)



The aforementioned function in the NFT_STAKING contract can only be invoked by the onlyOwner. This contract permits calling of all above functions. Additionally, the implementation of a multi-signature feature adds another layer of security to safeguard the owner's account.

For those who participated in the project Please carefully check the transparency of the implementation of the project.

Recommendation:

In terms of timeframes, there are three categories: short-term, long-term, and permanent.

For short-term solutions, a combination of timelock and multi-signature (2/3 or 3/5) can be used to mitigate risk by delaying sensitive operations and avoiding a single point of failure in key management. This includes implementing a timelock with a reasonable latency, such as 48 hours, for privileged operations; assigning privileged roles to multi-signature wallets to prevent private key compromise; and sharing the timelock contract and multi-signer addresses with the public via a medium/blog link.

For long-term solutions, a combination of timelock and DAO can be used to apply decentralization and transparency to the system. This includes implementing a timelock with a reasonable latency, such as 48 hours, for privileged operations; introducing a DAO/governance/voting module to increase transparency and user involvement; and sharing the timelock contract, multi-signer addresses, and DAO information with the public via a medium/blog link.

Finally, permanent solutions should be implemented to ensure the ongoing security and protection of the system.

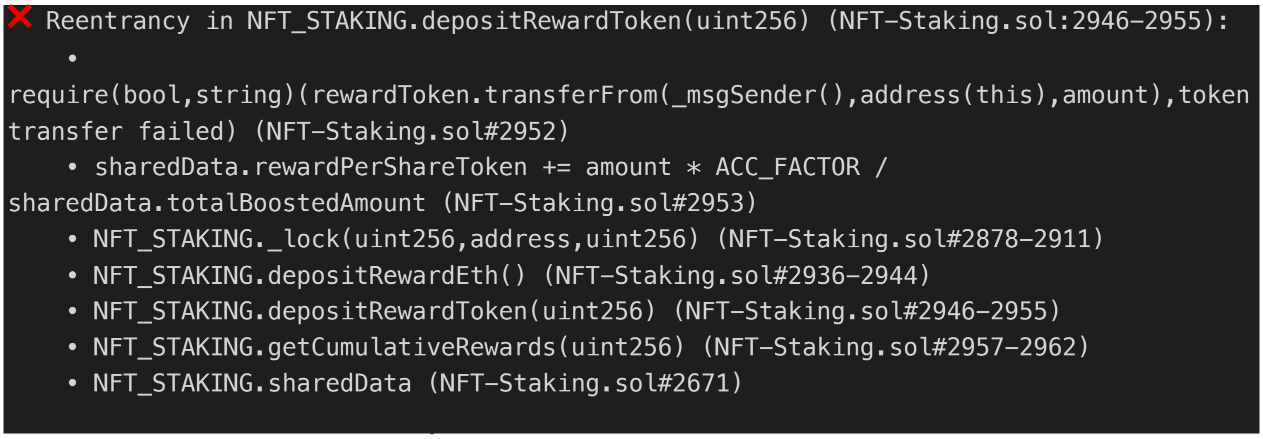
Alleviation:

Dogens Team has acknowledge this issue.

SEC-02: Reentrancy vulnerabilities (no theft of ethers) (reentrancy-no-eth)

Vulnerability Detail	Severity	Location	Category	Status
Reentrancy vulnerabilities (no theft of ethers) (reentrancy-no-eth)	Medium	Check on finding	Security Risk	Acknowledge

Finding:



```
✕ Reentrancy in NFT_STAKING.depositRewardToken(uint256) (NFT-Staking.sol:2946-2955):
    •
require(bool,string)(rewardToken.transferFrom(_msgSender(),address(this),amount),token
transfer failed) (NFT-Staking.sol#2952)
    • sharedData.rewardPerShareToken += amount * ACC_FACTOR /
sharedData.totalBoostedAmount (NFT-Staking.sol#2953)
    • NFT_STAKING._lock(uint256,address,uint256) (NFT-Staking.sol#2878-2911)
    • NFT_STAKING.depositRewardEth() (NFT-Staking.sol#2936-2944)
    • NFT_STAKING.depositRewardToken(uint256) (NFT-Staking.sol#2946-2955)
    • NFT_STAKING.getCumulativeRewards(uint256) (NFT-Staking.sol#2957-2962)
    • NFT_STAKING.sharedData (NFT-Staking.sol#2671)
```

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

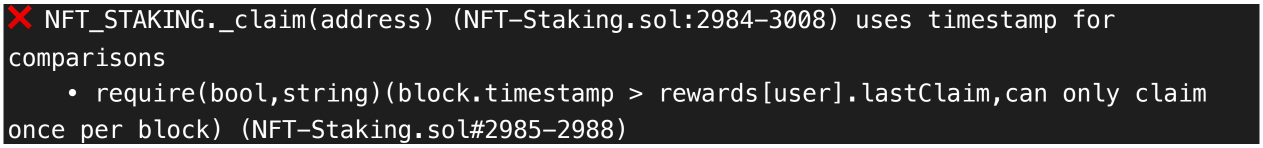
Alleviation:

Dogens Team has acknowledge this issue.

SEC-03: Dangerous usage of `block.timestamp` (timestamp)

Vulnerability Detail	Severity	Location	Category	Status
Dangerous usage of `block.timestamp` (timestamp)	Low	Check on finding	Security Risk	**Acknowledge**

Finding:



```
✗ NFT_STAKING._claim(address) (NFT-Staking.sol:2984-3008) uses timestamp for comparisons
    • require(bool,string)(block.timestamp > rewards[user].lastClaim,can only claim once per block) (NFT-Staking.sol#2985-2988)
```

Recommendation:

Avoid relying on `block.timestamp`.

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

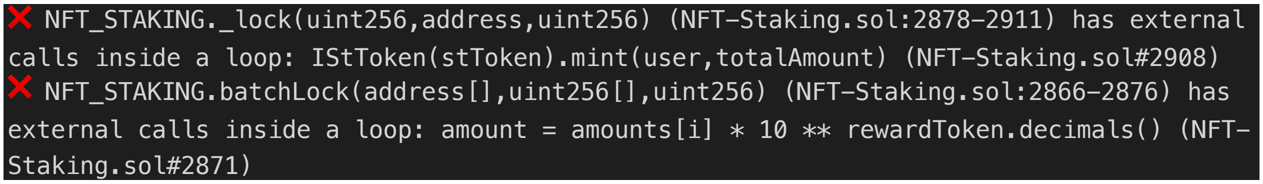
Alleviation:

Dogens Team has acknowledge this issue.

SEC-04: Multiple calls in a loop (calls-loop)

Vulnerability Detail	Severity	Location	Category	Status
Multiple calls in a loop (calls-loop)	Low	Check on finding	Logical Issue	Acknowledge

Finding:



```
❌ NFT_STAKING._lock(uint256,address,uint256) (NFT-Staking.sol:2878-2911) has external calls inside a loop: IStToken(stToken).mint(user,totalAmount) (NFT-Staking.sol#2908)
❌ NFT_STAKING.batchLock(address[],uint256[],uint256) (NFT-Staking.sol:2866-2876) has external calls inside a loop: amount = amounts[i] * 10 ** rewardToken.decimals() (NFT-Staking.sol#2871)
```

Recommendation:

Favor [pull over push](https://github.com/ethereum/wiki/wiki/Safety#favor-pull-over-push-for-external-calls) strategy for external calls.

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

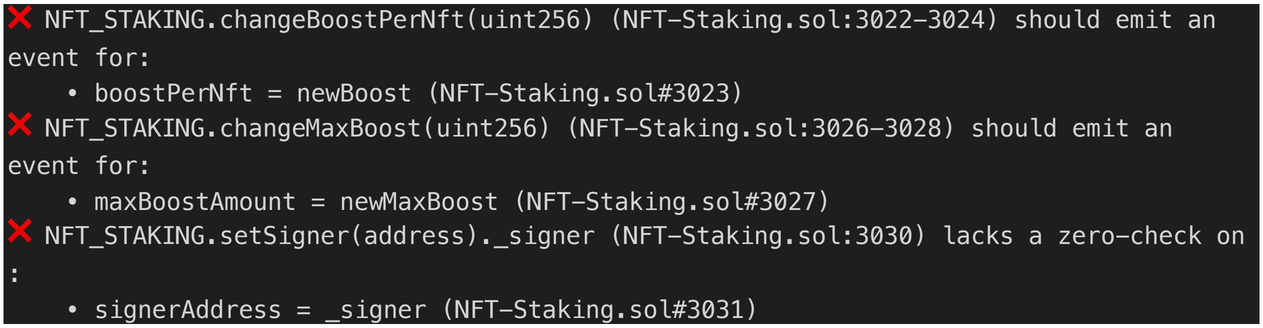
Alleviation:

Dogens Team has acknowledge this issue.

SEC-05: Missing Events Arithmetic (events-maths)

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

Finding:



```
✗ NFT_STAKING.changeBoostPerNft(uint256) (NFT-Staking.sol:3022-3024) should emit an
event for:
    • boostPerNft = newBoost (NFT-Staking.sol#3023)
✗ NFT_STAKING.changeMaxBoost(uint256) (NFT-Staking.sol:3026-3028) should emit an
event for:
    • maxBoostAmount = newMaxBoost (NFT-Staking.sol#3027)
✗ NFT_STAKING.setSigner(address)._signer (NFT-Staking.sol:3030) lacks a zero-check on
:
    • signerAddress = _signer (NFT-Staking.sol#3031)
```

Recommendation:

Recommendation: Emit an event for critical parameter changes.

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

Exploit Scenario:

-

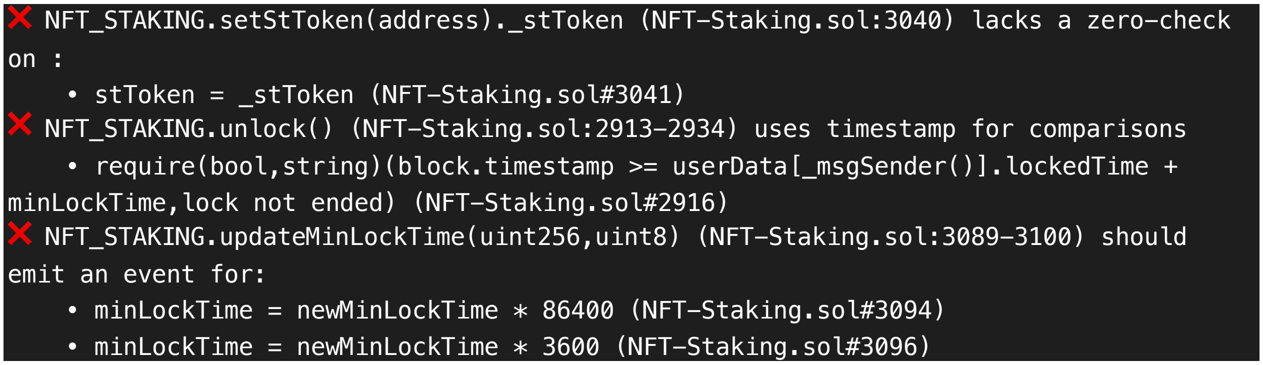
Alleviation:

Dogens Team has acknowledge this issue.

SEC-06: Missing Zero Address Validation (missing-zero-check)

Vulnerability Detail	Severity	Location	Category	Status
Missing Zero Address Validation (missing-zero-check)	Low	Check on finding	Best Practices	Acknowledge

Finding:



```
❌ NFT_STAKING.setStToken(address)._stToken (NFT-Staking.sol:3040) lacks a zero-check
on :
    • stToken = _stToken (NFT-Staking.sol#3041)
❌ NFT_STAKING.unlock() (NFT-Staking.sol:2913-2934) uses timestamp for comparisons
    • require(bool,string)(block.timestamp >= userData[_msgSender()].lockedTime +
minLockTime,lock not ended) (NFT-Staking.sol#2916)
❌ NFT_STAKING.updateMinLockTime(uint256,uint8) (NFT-Staking.sol:3089-3100) should
emit an event for:
    • minLockTime = newMinLockTime * 86400 (NFT-Staking.sol#3094)
    • minLockTime = newMinLockTime * 3600 (NFT-Staking.sol#3096)
```

Recommendation:

Check that the address is not zero.

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Exploit Scenario:

-

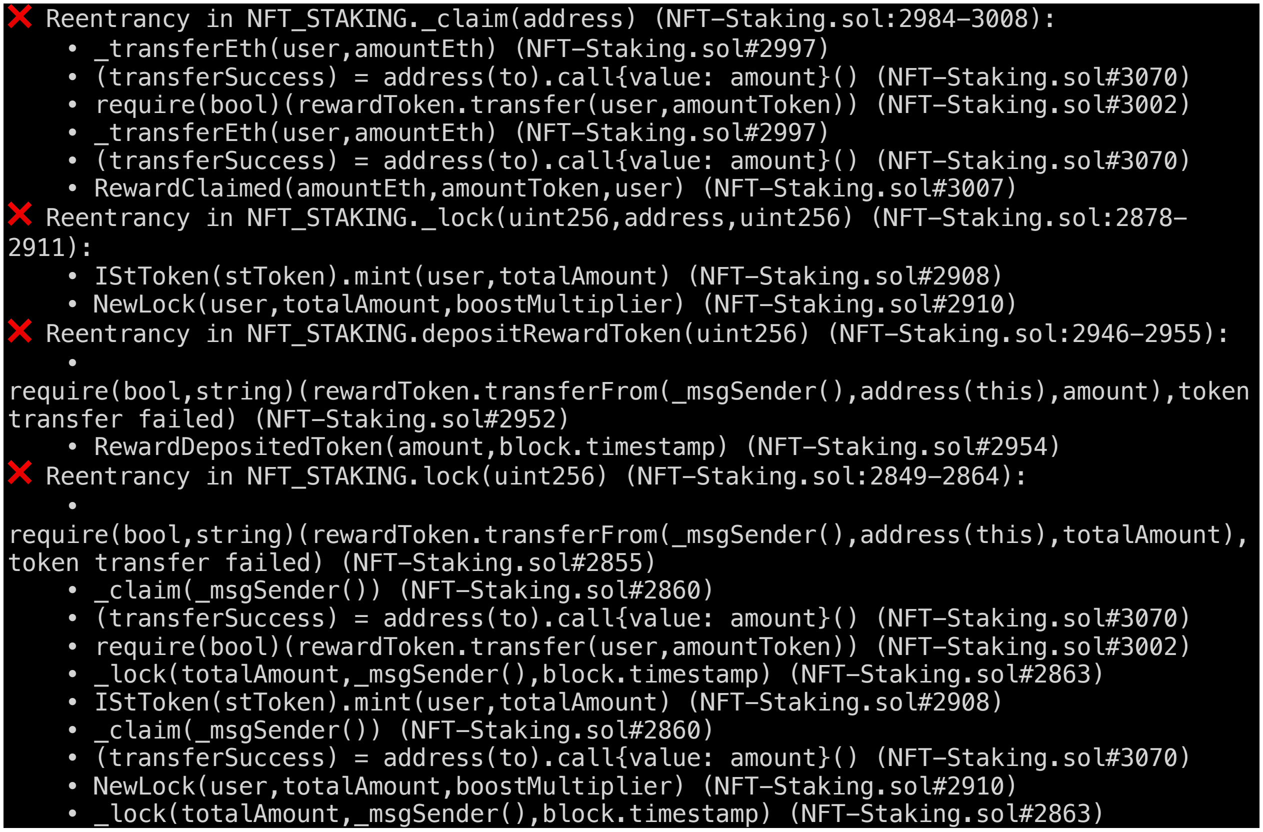
Alleviation:

Dogens Team has acknowledge this issue.

SEC-07: Reentrancy vulnerabilities leading to out-of-order Events (reentrancy-events)

Vulnerability Detail	Severity	Location	Category	Status
Reentrancy vulnerabilities leading to out-of-order Events (reentrancy-events)	Low	Check on finding	Best Practices	**Acknowledge**

Finding:



```
❌ Reentrancy in NFT_STAKING._claim(address) (NFT-Staking.sol:2984-3008):
    • _transferEth(user,amountEth) (NFT-Staking.sol#2997)
    • (transferSuccess) = address(to).call{value: amount}() (NFT-Staking.sol#3070)
    • require(bool)(rewardToken.transfer(user,amountToken)) (NFT-Staking.sol#3002)
    • _transferEth(user,amountEth) (NFT-Staking.sol#2997)
    • (transferSuccess) = address(to).call{value: amount}() (NFT-Staking.sol#3070)
    • RewardClaimed(amountEth,amountToken,user) (NFT-Staking.sol#3007)
❌ Reentrancy in NFT_STAKING._lock(uint256,address,uint256) (NFT-Staking.sol:2878-
2911):
    • IStToken(stToken).mint(user,totalAmount) (NFT-Staking.sol#2908)
    • NewLock(user,totalAmount,boostMultiplier) (NFT-Staking.sol#2910)
❌ Reentrancy in NFT_STAKING.depositRewardToken(uint256) (NFT-Staking.sol:2946-2955):
    •
require(bool,string)(rewardToken.transferFrom(_msgSender(),address(this),amount),token
transfer failed) (NFT-Staking.sol#2952)
    • RewardDepositedToken(amount,block.timestamp) (NFT-Staking.sol#2954)
❌ Reentrancy in NFT_STAKING.lock(uint256) (NFT-Staking.sol:2849-2864):
    •
require(bool,string)(rewardToken.transferFrom(_msgSender(),address(this),totalAmount),
token transfer failed) (NFT-Staking.sol#2855)
    • _claim(_msgSender()) (NFT-Staking.sol#2860)
    • (transferSuccess) = address(to).call{value: amount}() (NFT-Staking.sol#3070)
    • require(bool)(rewardToken.transfer(user,amountToken)) (NFT-Staking.sol#3002)
    • _lock(totalAmount,_msgSender(),block.timestamp) (NFT-Staking.sol#2863)
    • IStToken(stToken).mint(user,totalAmount) (NFT-Staking.sol#2908)
    • _claim(_msgSender()) (NFT-Staking.sol#2860)
    • (transferSuccess) = address(to).call{value: amount}() (NFT-Staking.sol#3070)
    • NewLock(user,totalAmount,boostMultiplier) (NFT-Staking.sol#2910)
    • _lock(totalAmount,_msgSender(),block.timestamp) (NFT-Staking.sol#2863)
```

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-3

Exploit Scenario:

-

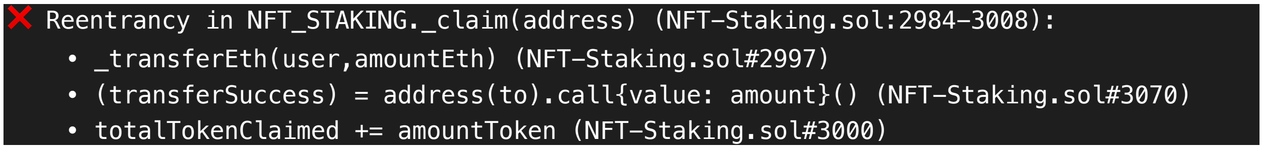
Alleviation:

Dogens Team has acknowledge this issue.

SEC-08: Benign reentrancy vulnerabilities (reentrancy-benign)

Vulnerability Detail	Severity	Location	Category	Status
Benign reentrancy vulnerabilities (reentrancy-benign)	Informational	Check on finding	Best Practices	**Acknowledge**

Finding:



```
✗ Reentrancy in NFT_STAKING._claim(address) (NFT-Staking.sol:2984-3008):
    • _transferEth(user,amountEth) (NFT-Staking.sol#2997)
    • (transferSuccess) = address(to).call{value: amount}() (NFT-Staking.sol#3070)
    • totalTokenClaimed += amountToken (NFT-Staking.sol#3000)
```

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-2

Exploit Scenario:

Alleviation:

Dogens Team has acknowledge this issue.

SEC-09: Missing inheritance (missing-inheritance)

Vulnerability Detail	Severity	Location	Category	Status
Missing inheritance (missing-inheritance)	Informational	Check on finding	Best Practices	**Acknowledge**

Finding:

❌ stToken (stToken.sol:501-523) should inherit from IStToken (NFT-Staking.sol#2630-2633)

Recommendation:

Inherit from the missing interface or contract.

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

Exploit Scenario:

-

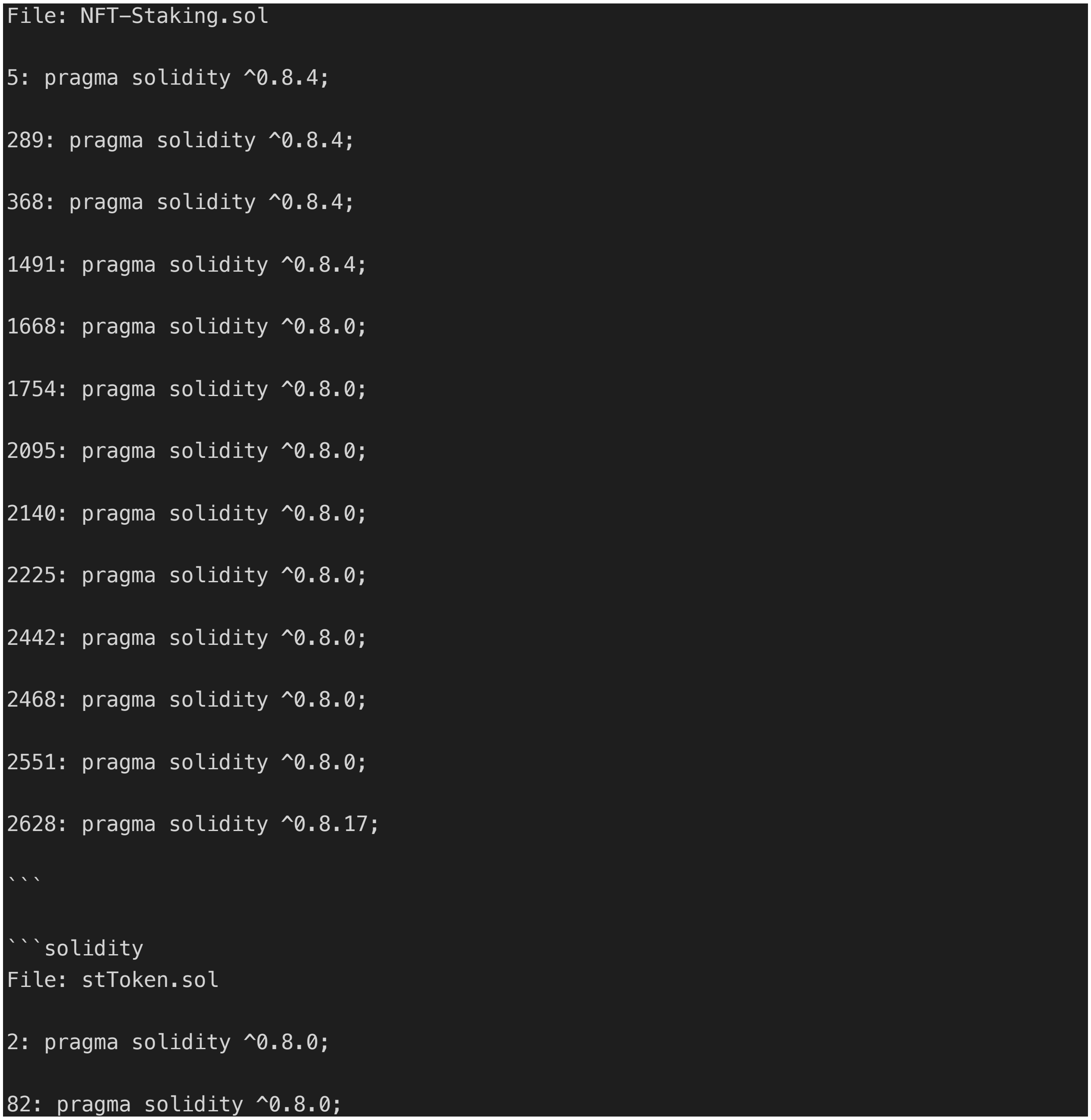
Alleviation:

Dogens Team has acknowledge this issue.

SEC-10: Unlocked pragma

Vulnerability Detail	Severity	Location	Category	Status
Unlocked pragma	Informational	Check on finding	Best Practices	**Acknowledge**

Finding:



```
File: NFT-Staking.sol

5: pragma solidity ^0.8.4;

289: pragma solidity ^0.8.4;

368: pragma solidity ^0.8.4;

1491: pragma solidity ^0.8.4;

1668: pragma solidity ^0.8.0;

1754: pragma solidity ^0.8.0;

2095: pragma solidity ^0.8.0;

2140: pragma solidity ^0.8.0;

2225: pragma solidity ^0.8.0;

2442: pragma solidity ^0.8.0;

2468: pragma solidity ^0.8.0;

2551: pragma solidity ^0.8.0;

2628: pragma solidity ^0.8.17;

```

```solidity
File: stToken.sol

2: pragma solidity ^0.8.0;

82: pragma solidity ^0.8.0;
```

~~~
110: pragma solidity ^0.8.0;

136: pragma solidity ^0.8.0;

499: pragma solidity ^0.8.0;

```
~~~

Exploit Scenario:

-

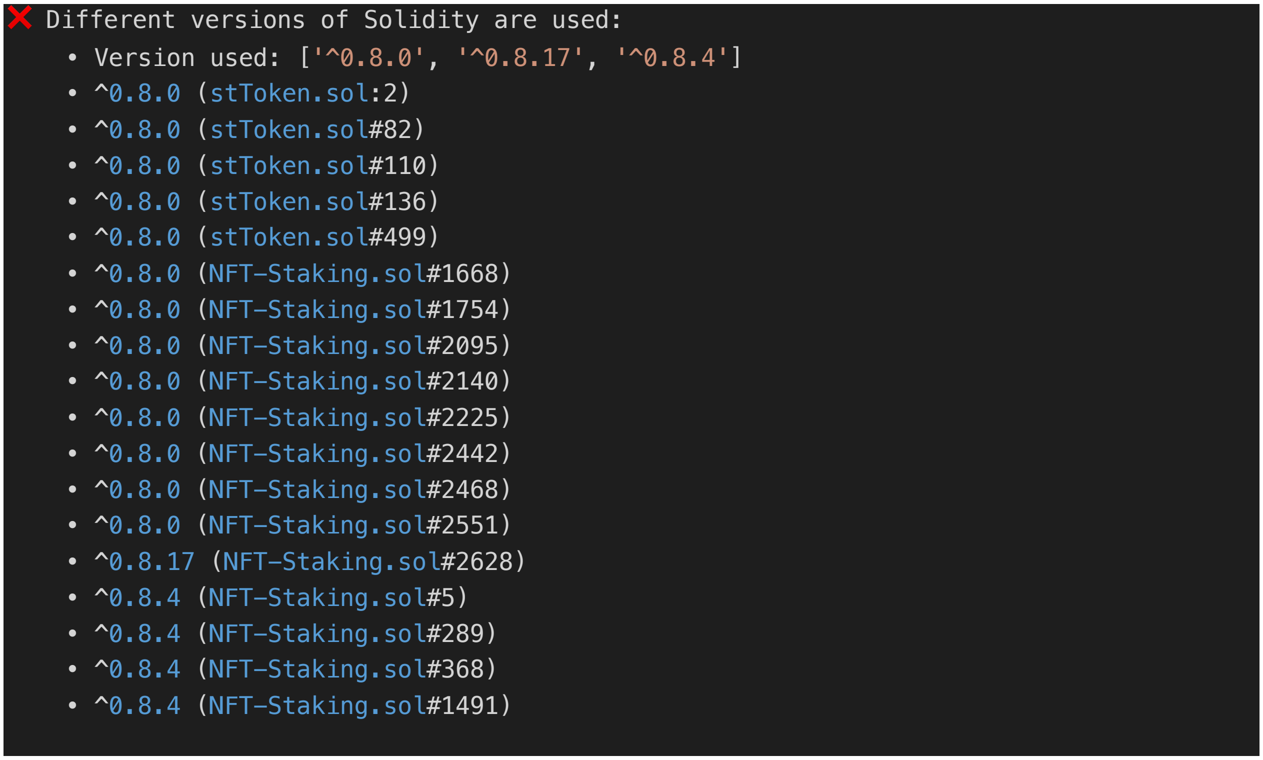
Alleviation:

Dogens Team has acknowledge this issue.

SEC-11: If different pragma directives are used (pragma)

Vulnerability Detail	Severity	Location	Category	Status
If different pragma directives are used (pragma	Informational	Check on finding	Best Practices	**Acknowledge**

Finding:



```
❌ Different versions of Solidity are used:
    • Version used: ['^0.8.0', '^0.8.17', '^0.8.4']
    • ^0.8.0 (stToken.sol:2)
    • ^0.8.0 (stToken.sol#82)
    • ^0.8.0 (stToken.sol#110)
    • ^0.8.0 (stToken.sol#136)
    • ^0.8.0 (stToken.sol#499)
    • ^0.8.0 (NFT-Staking.sol#1668)
    • ^0.8.0 (NFT-Staking.sol#1754)
    • ^0.8.0 (NFT-Staking.sol#2095)
    • ^0.8.0 (NFT-Staking.sol#2140)
    • ^0.8.0 (NFT-Staking.sol#2225)
    • ^0.8.0 (NFT-Staking.sol#2442)
    • ^0.8.0 (NFT-Staking.sol#2468)
    • ^0.8.0 (NFT-Staking.sol#2551)
    • ^0.8.17 (NFT-Staking.sol#2628)
    • ^0.8.4 (NFT-Staking.sol#5)
    • ^0.8.4 (NFT-Staking.sol#289)
    • ^0.8.4 (NFT-Staking.sol#368)
    • ^0.8.4 (NFT-Staking.sol#1491)
```

Recommendation:

Use one Solidity version.

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Exploit Scenario:

-

Alleviation:

Dogens Team has acknowledge this issue.

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

Visibility, Mutability, Modifier function testing

Components

📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
4	4	7	5

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
103	13

External	Internal	Private	Pure	View
66	163	11	41	73

StateVariables

Total	🌐Public
58	24

Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts

`^0.8.0`
`^0.8.4`
`^0.8.17`
		`yes`	`yes`
			(20 asm blocks)	

📤 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	🔆 ECRecover	🌀 New/Create/Create2
`yes`			`yes`	`yes`	

♻ TryCatch	Σ Unchecked
yes	yes

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❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
IERC20Metadata	Interface	IERC20		
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
ERC20	Implementation	Context, IERC20, IERC20Metadata		
└		Public ❗	🛑	NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗

Contract	Type	Bases		
└	decimals	Public ❗		NO ❗
└	totalSupply	Public ❗		NO ❗
└	balanceOf	Public ❗		NO ❗
└	transfer	Public ❗	🛑	NO ❗
└	allowance	Public ❗		NO ❗
└	approve	Public ❗	🛑	NO ❗
└	transferFrom	Public ❗	🛑	NO ❗
└	increaseAllowance	Public ❗	🛑	NO ❗
└	decreaseAllowance	Public ❗	🛑	NO ❗
└	_transfer	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_spendAllowance	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
└	_afterTokenTransfer	Internal 🔒	🛑	
stToken	Implementation	ERC20		
└		Public ❗	🛑	ERC20
└	mint	External ❗	🛑	onlyStaking
└	burn	External ❗	🛑	onlyStaking
└	_beforeTokenTransfer	Internal 🔒	🛑	onlyStaking

Contract	Type	Bases		
IERC721A	Interface			
└	totalSupply	External ❗		NO ❗
└	supportsInterface	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	ownerOf	External ❗		NO ❗
└	safeTransferFrom	External ❗	💵	NO ❗
└	safeTransferFrom	External ❗	💵	NO ❗
└	transferFrom	External ❗	💵	NO ❗
└	approve	External ❗	💵	NO ❗
└	setApprovalForAll	External ❗	🛑	NO ❗
└	getApproved	External ❗		NO ❗
└	isApprovedForAll	External ❗		NO ❗
└	name	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	tokenURI	External ❗		NO ❗
IERC721AQueryable	Interface	IERC721A		
└	explicitOwnershipOf	External ❗		NO ❗
└	explicitOwnershipsOf	External ❗		NO ❗
└	tokensOfOwnerIn	External ❗		NO ❗
└	tokensOfOwner	External ❗		NO ❗
ERC721A__IERC721Receiver	Interface			

Contract	Type	Bases		
└	onERC721Received	External ❗	🛑	NO ❗
ERC721A	Implementation	IERC721A		
└		Public ❗	🛑	NO ❗
└	_startTokenId	Internal 🔒		
└	_nextTokenId	Internal 🔒		
└	totalSupply	Public ❗		NO ❗
└	_totalMinted	Internal 🔒		
└	_totalBurned	Internal 🔒		
└	balanceOf	Public ❗		NO ❗
└	_numberMinted	Internal 🔒		
└	_numberBurned	Internal 🔒		
└	_getAux	Internal 🔒		
└	_setAux	Internal 🔒	🛑	
└	supportsInterface	Public ❗		NO ❗
└	name	Public ❗		NO ❗
└	symbol	Public ❗		NO ❗
└	tokenURI	Public ❗		NO ❗
└	_baseURI	Internal 🔒		
└	ownerOf	Public ❗		NO ❗
└	_ownershipOf	Internal 🔒		
└	_ownershipAt	Internal 🔒		
└	_initializeOwnershipAt	Internal 🔒	🛑	
└	_packedOwnershipOf	Private 🔐		

Contract	Type	Bases		
└	_unpackedOwnership	Private 🔐		
└	_packOwnershipData	Private 🔐		
└	_nextInitializedFlag	Private 🔐		
└	approve	Public ❗	💵	NO ❗
└	getApproved	Public ❗		NO ❗
└	setApprovalForAll	Public ❗	🛑	NO ❗
└	isApprovedForAll	Public ❗		NO ❗
└	_exists	Internal 🔒		
└	_isSenderApprovedOrOwner	Private 🔐		
└	_getApprovedSlotAndAddress	Private 🔐		
└	transferFrom	Public ❗	💵	NO ❗
└	safeTransferFrom	Public ❗	💵	NO ❗
└	safeTransferFrom	Public ❗	💵	NO ❗
└	_beforeTokenTransfers	Internal 🔒	🛑	
└	_afterTokenTransfers	Internal 🔒	🛑	
└	_checkContractOnERC721Received	Private 🔐	🛑	
└	_mint	Internal 🔒	🛑	
└	_mintERC2309	Internal 🔒	🛑	
└	_safeMint	Internal 🔒	🛑	
└	_safeMint	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	

FULL AUDIT REPORT

Contract	Type	Bases		
└	_burn	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_setExtraDataAt	Internal 🔒	🛑	
└	_extraData	Internal 🔒		
└	_nextExtraData	Private 🔐		
└	_msgSenderERC721A	Internal 🔒		
└	_toString	Internal 🔒		
ERC721AQueryable	Implementation	ERC721A, IERC721AQueryable		
└	explicitOwnershipOf	Public ❗		NO ❗
└	explicitOwnershipsOf	External ❗		NO ❗
└	tokensOfOwnerIn	External ❗		NO ❗
└	tokensOfOwner	External ❗		NO ❗
IERC20	Interface			
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
└	decimals	External ❗		NO ❗
Math	Library			

Contract	Type	Bases		
└	max	Internal 🔒		
└	min	Internal 🔒		
└	average	Internal 🔒		
└	ceilDiv	Internal 🔒		
└	mulDiv	Internal 🔒		
└	mulDiv	Internal 🔒		
└	sqrt	Internal 🔒		
└	sqrt	Internal 🔒		
└	log2	Internal 🔒		
└	log2	Internal 🔒		
└	log10	Internal 🔒		
└	log10	Internal 🔒		
└	log256	Internal 🔒		
└	log256	Internal 🔒		
SignedMath	Library			
└	max	Internal 🔒		
└	min	Internal 🔒		
└	average	Internal 🔒		
└	abs	Internal 🔒		
Strings	Library			
└	toString	Internal 🔒		
└	toString	Internal 🔒		

Contract	Type	Bases		
└	toHexString	Internal 🔒		
└	toHexString	Internal 🔒		
└	toHexString	Internal 🔒		
└	equal	Internal 🔒		
ECDSA	Library			
└	_throwError	Private 🔐		
└	tryRecover	Internal 🔒		
└	recover	Internal 🔒		
└	tryRecover	Internal 🔒		
└	recover	Internal 🔒		
└	tryRecover	Internal 🔒		
└	recover	Internal 🔒		
└	toEthSignedMessageHash	Internal 🔒		
└	toEthSignedMessageHash	Internal 🔒		
└	toTypedDataHash	Internal 🔒		
└	toDataWithIntendedValidatorHash	Internal 🔒		
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		

Contract	**Type**	**Bases**		
└		Public ❗	🛑	NO ❗
└	owner	Public ❗		NO ❗
└	_checkOwner	Internal 🔒		
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_transferOwnership	Internal 🔒	🛑	
ReentrancyGuard	Implementation			
└		Public ❗	🛑	NO ❗
└	_nonReentrantBefore	Private 🔐	🛑	
└	_nonReentrantAfter	Private 🔐	🛑	
└	_reentrancyGuardEntered	Internal 🔒		
IStToken	Interface			
└	mint	External ❗	🛑	NO ❗
└	burn	External ❗	🛑	NO ❗
NFT_STAKING	Implementation	ERC721A, ERC721AQueryable, Ownable, ReentrancyGuard		
└		Public ❗	🛑	ERC721A
└	_startTokenId	Internal 🔒		
└	TotalBurned	Public ❗		NO ❗

Contract	Type	Bases		
└	next	Public ❗		NO ❗
└	toggleSale	Public ❗	🛑	onlyOwner
└	getSigner	Internal 🔒		
└	mint	External ❗	💵	NO ❗
└	mintbyref	External ❗	💵	NO ❗
└	giftmint	External ❗	🛑	onlyOwner
└	emergencyWithdraw	External ❗	💵	onlyOwner
└	_baseURI	Internal 🔒		
└	setMintRate	Public ❗	🛑	onlyOwner
└	setBaseURI	External ❗	🛑	onlyOwner
└	changeMaxMintPerWallet	External ❗	🛑	onlyOwner
└	changeMaxSupply	External ❗	🛑	onlyOwner
└	tokenURI	Public ❗		NO ❗
└	lock	External ❗	🛑	NO ❗
└	batchLock	External ❗	🛑	onlyOwner
└	_lock	Internal 🔒	🛑	
└	unlock	Public ❗	🛑	nonReentrant
└	depositRewardEth	External ❗	💵	onlyOwner

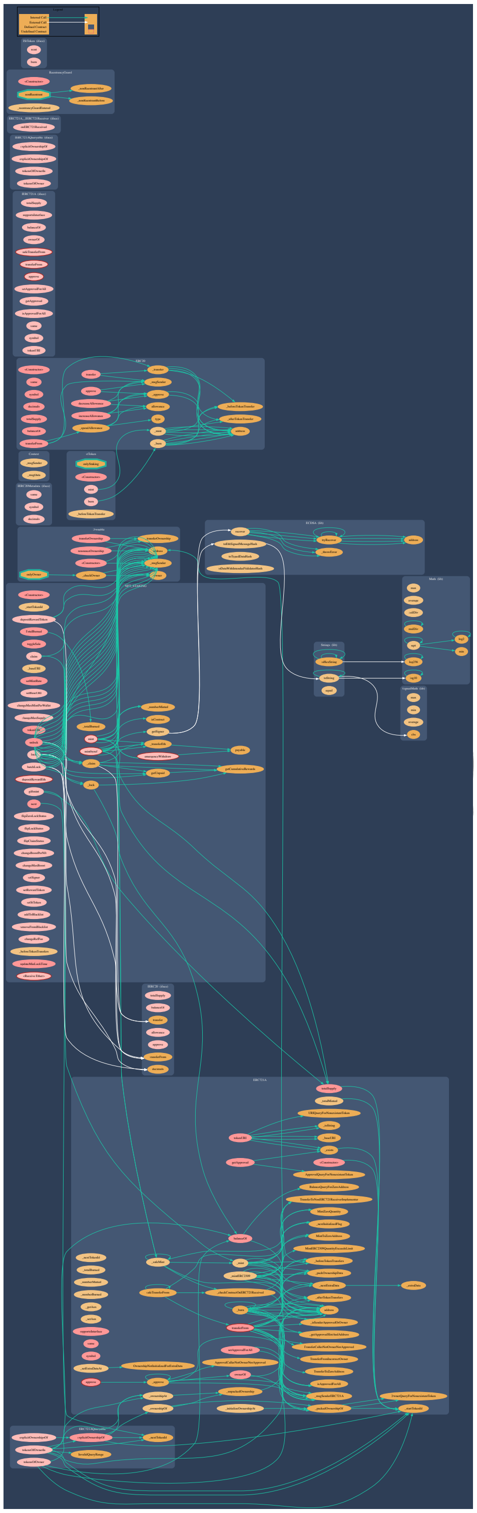
Contract	Type	Bases		
└	depositRewardToken	External ❗	🛑	onlyOwner
└	getCumulativeRewards	Internal 🔒		
└	getUnpaid	Public ❗		NO ❗
└	claim	External ❗	🛑	nonReentrant
└	_claim	Internal 🔒	🛑	
└	flipZeroLockStatus	External ❗	🛑	onlyOwner
└	flipLockStatus	External ❗	🛑	onlyOwner
└	flipClaimStatus	External ❗	🛑	onlyOwner
└	changeBoostPerNft	External ❗	🛑	onlyOwner
└	changeMaxBoost	External ❗	🛑	onlyOwner
└	setSigner	External ❗	🛑	onlyOwner
└	setRewardToken	External ❗	🛑	onlyOwner
└	setStToken	External ❗	🛑	onlyOwner
└	addToBlacklist	External ❗	🛑	onlyOwner
└	removeFromBlacklist	External ❗	🛑	onlyOwner
└	changeRefFee	External ❗	🛑	onlyOwner
└	_transferEth	Internal 🔒	🛑	

Contract	Type	Bases		
└	isContract	Internal 🔒		
└	_beforeTokenTransfers	Internal 🔒	🛑	
└	updateMinLockTime	Public ❗	🛑	onlyOwner
└		External ❗	💵	NO ❗

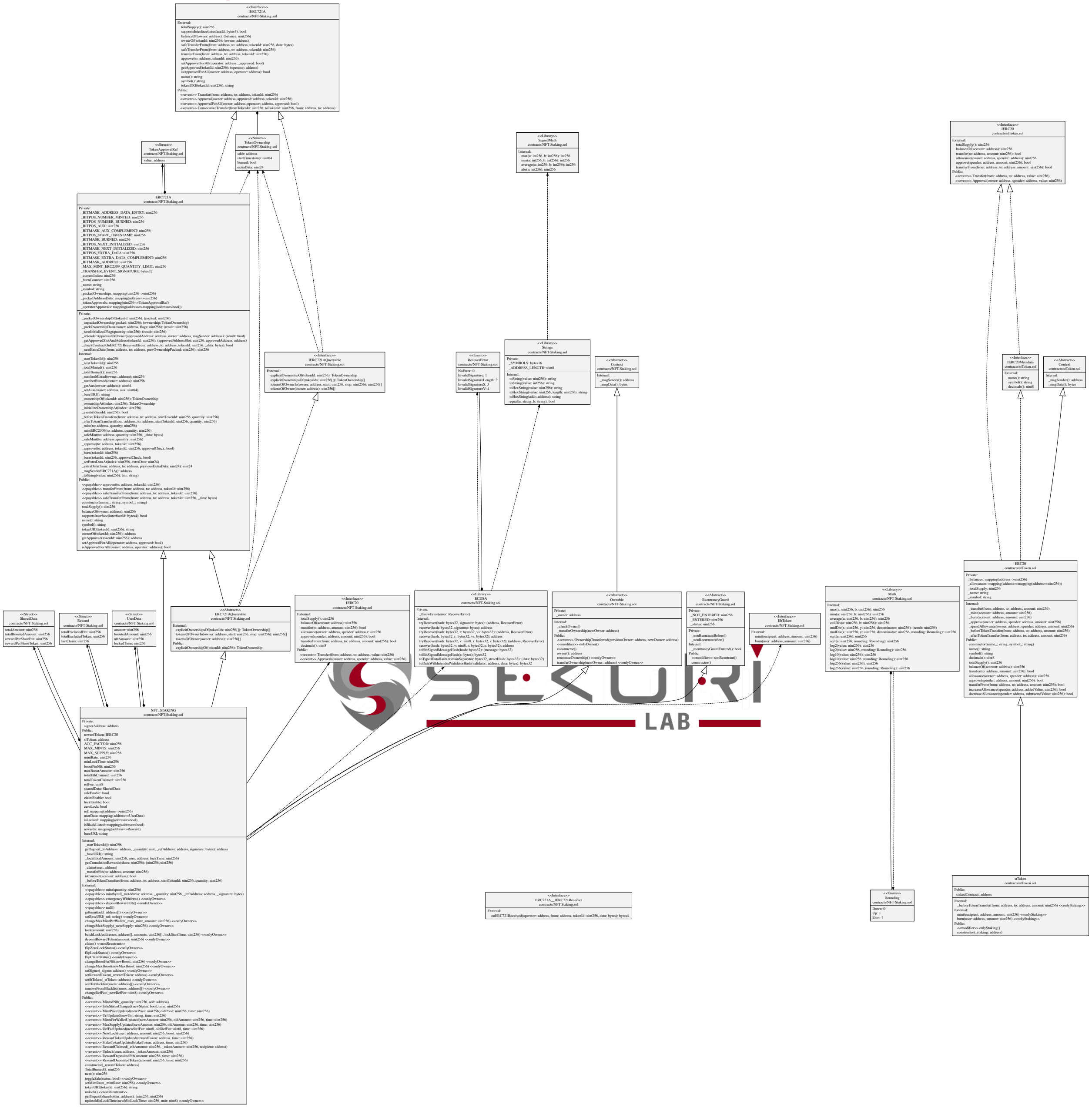
Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Inheritate Function Relation Graph



UML Class Diagram


<<Interface>> IERC721A contracts/NFT-Staking.sol
<<Struct>> TokenApprovalRef contracts/NFT-Staking.sol
<<Struct>> TokenOwnership contracts/NFT-Staking.sol
ERC721A contracts/NFT-Staking.sol
<<Library>> SignedMath contracts/NFT-Staking.sol
<<Interface>> IERC20 contracts/xfToken.sol
<<Interface>> IERC721AQueryable contracts/NFT-Staking.sol
<<Enum>> RecoverError contracts/NFT-Staking.sol
<<Library>> Strings contracts/NFT-Staking.sol
<<Abstract>> Context contracts/NFT-Staking.sol
<<Interface>> IERC20Metadata contracts/xfToken.sol
<<Abstract>> Context contracts/xfToken.sol
<<Struct>> ShareData contracts/NFT-Staking.sol
<<Struct>> Reward contracts/NFT-Staking.sol
<<Struct>> UserData contracts/NFT-Staking.sol
<<Abstract>> ERC721AQueryable contracts/NFT-Staking.sol
<<Interface>> IERC20 contracts/NFT-Staking.sol
<<Library>> ECDSA contracts/NFT-Staking.sol
<<Abstract>> Ownable contracts/NFT-Staking.sol
<<Abstract>> ReentrancyGuard contracts/NFT-Staking.sol
<<Interface>> IfxToken contracts/NFT-Staking.sol
<<Library>> Math contracts/NFT-Staking.sol
ERC20 contracts/xfToken.sol
NFT_STAKING contracts/NFT-Staking.sol
<<Interface>> ERC721A__IERC721Receiver contracts/NFT-Staking.sol
<<Enum>> Rounding contracts/NFT-Staking.sol
xfToken contracts/xfToken.sol

About SECURI LAB

SECURI LAB is a group of cyber security experts providing cyber security consulting, smart contract security audits, and KYC services.

Follow Us On:

Website	https://securi-lab.com/
Twitter	https://twitter.com/SECURI_LAB
Telegram	https://t.me/securi_lab
Medium	https://medium.com/@securi