



*Security Assessment*

# Not Financial Advice nsurance

Verified on 03/10/2024

## SUMMARY

Project

Not Financial Advice insurance

CHAIN

Ethereum

METHODOLOGY

Manual &amp; Automatic Analysis

FILES

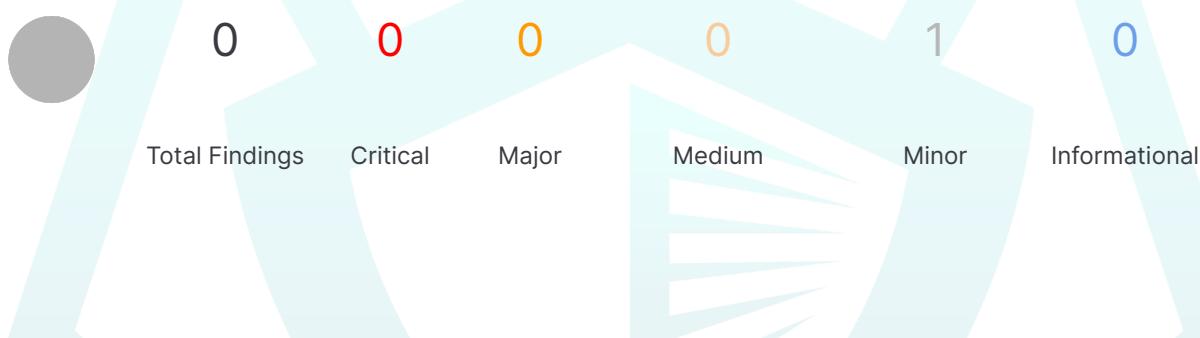
Single

DELIVERY

03/10/2024

TYPE

Standard Audit


■ 0 Critical

An exposure that can affect the contract functions in several events that can risk and disrupt the contract

■ 0 Major

An exposure that can affect the outcome when using the contract that can serve as an opening in manipulating the contract in an unwanted manner

■ 0 Medium

An opening that could affect the outcome in executing the contract in a specific situation

■ 1 Minor

An opening but doesn't have an impact on the functionality of the contract

■ 0 Informational

An opening that consists information but will not risk or affect the contract

**STATUS**
✓ **AUDIT PASSED**

## TABLE OF CONTENTS | Not Financial Advice nsurance

### | **Summary**

Project Summary  
Findings Summary  
Disclaimer  
Scope of Work  
Auditing Approach

### | **Project Information**

Token/Project Details  
Inheritance Graph  
Call Graph

### | **Findings**

Issues  
SWC Attacks  
CW Assessment  
Fixes & Recommendation  
Audit Comments

## DISCLAIMER | Not Financial Advice nsurance

**ContractWolf** audits and reports should not be considered as a form of project's "Advertisement" and does not cover any interaction and assessment from "Project Contract" to "External Contracts" such as PancakeSwap, UniSwap, SushiSwap or similar.

**ContractWolf** does not provide any warranty on its released report and should not be used as a decision to invest into audited projects.

**ContractWolf** provides a transparent report to all its "Clients" and to its "Clients Participants" and will not claim any guarantee of bug-free code within its **SMART CONTRACT**.

**ContractWolf's** presence is to analyze, audit and assess the Client's Smart Contract to find any underlying risk and to eliminate any logic and flow errors within its code.

*Each company or project should be liable to its security flaws and functionalities.*

## SCOPE OF WORK | Not Financial Advice nsurance

**Not Financial Advice** team has agreed and provided us with the files that need to be tested (*Github, BSCscan, Etherscan, Local files etc*). The scope of audit is the main contract.

The goal of this engagement is to identify if there is a possibility of security flaws in the implementation of smart contract and its systems.

ContractWolf will be focusing on contract issues and functionalities along with the project claims from smart contract to their website, whitepaper, repository which has been provided by **Not Financial Advice**.

## AUDITING APPROACH | Not Financial Advice nsurance

Every line of code along with its functionalities will undergo manual review to check for security issues, quality of logic and contract scope of inheritance. The manual review will be done by our team that will document any issues that they discovered.

### METHODOLOGY

The auditing process follows a routine series of steps :

1. Code review that includes the following :
  - Review of the specifications, sources and instructions provided to ContractWolf to make sure we understand the size, scope and functionality of the smart contract.
  - Manual review of code. Our team will have a process of reading the code line-by-line with the intention of identifying potential vulnerabilities, underlying and hidden security flaws.
2. Testing and automated analysis that includes :
  - Testing the smart contract function with common test cases and scenarios to ensure that it returns the expected results.
3. Best practices and ethical review. The team will review the contract with the aim to improve efficiency, effectiveness, clarifications, maintainability, security and control within the smart contract.
4. Recommendations to help the project take steps to eliminate or minimize threats and secure the smart contract.

## TOKEN DETAILS | Not Financial Advice insurance



NFAi is a decentralized Hedge Fund experiment that aims to use the power of neural net technology to make leverage trading decisions on decentralized leverage trading platforms.

|            |   |         |                   |          |
|------------|---|---------|-------------------|----------|
| Token Name | n | Decimal | Total Supply      | Chain    |
| insurance  |   | 18      | 1,000,000,000,000 | Ethereum |

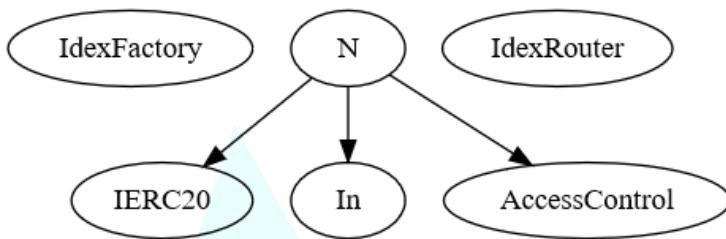
## SOURCE

Source *Sent Via local-files*

## INHERITANCE GRAPH

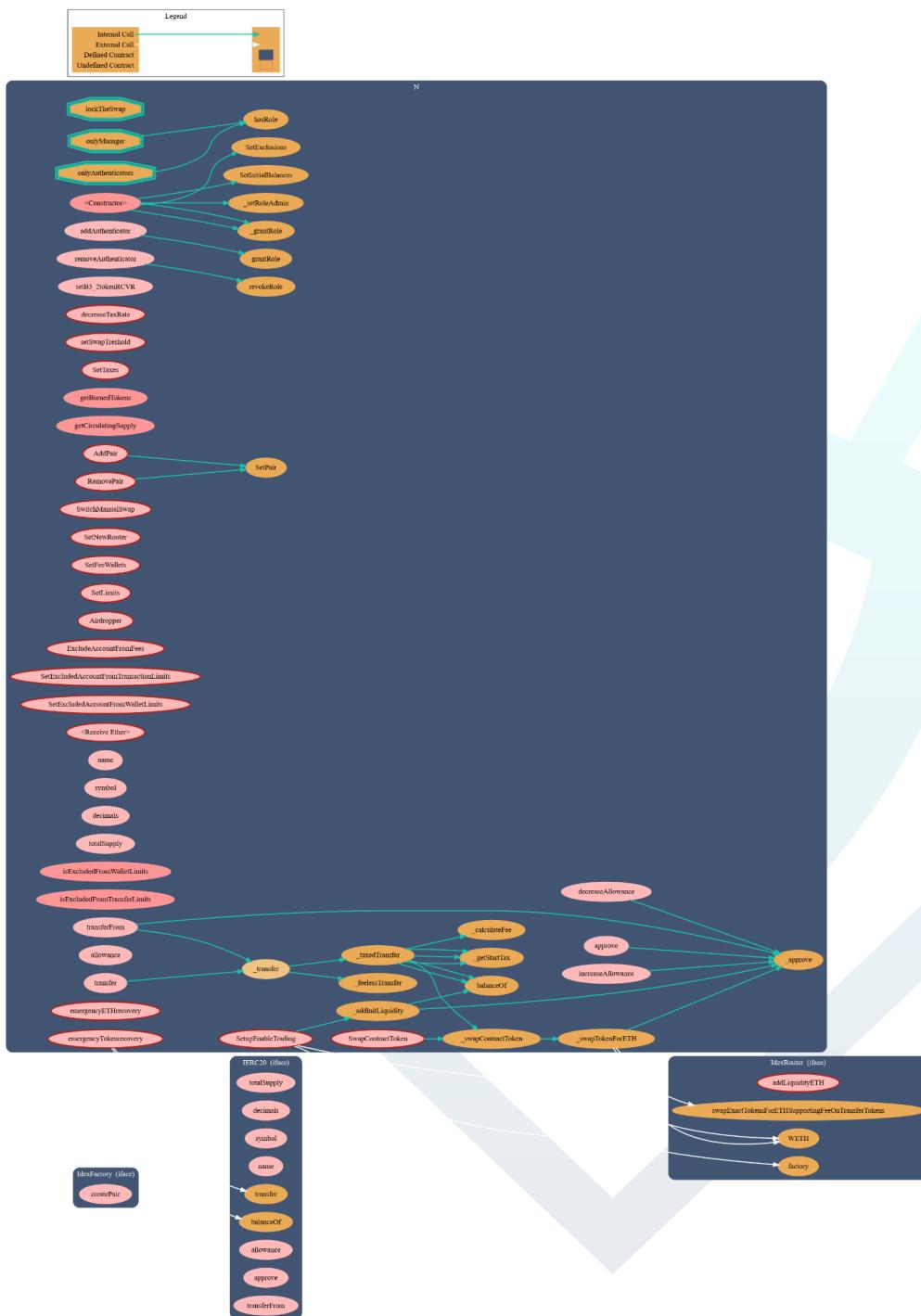
Not Financial Advice nsurance

Inheritance Graph of Contract Functions



# CALL GRAPH

Call Graph of Contract Functions



## FINDINGS

Not Financial Advice nsurance



0

0

0

0

1

0

Total Findings

Critical

Major

Medium

Minor

Informational

This report has been prepared to state the issues and vulnerabilities for Not Financial Advice nsurance through this audit. The goal of this report findings is to identify specifically and fix any underlying issues and errors

| ID | Title | File & Line # | Severity | Status |
|----|-------|---------------|----------|--------|
|----|-------|---------------|----------|--------|

CW-007

Deprecated keywords

L : 276, 278

Minor

● Resolved

# SWC ATTACKS

Not Financial Advice nsurance

Smart Contract Weakness Classification and Test Cases

| ID      | Description   | Status |
|---------|---|--------|
| SWC-100 | Function Default Visibility                         | Passed |
| SWC-101 | Integer Overflow and Underflow                      | Passed |
| SWC-102 | Outdated Compiler Version                           | Passed |
| SWC-103 | FloatingPragma                                      | Passed |
| SWC-104 | Unchecked Call Return Value                         | Passed |
| SWC-105 | Unprotected Ether Withdrawal                        | Passed |
| SWC-106 | Unprotected SELF DESTRUCT Instruction               | Passed |
| SWC-107 | Reentrancy  | Passed |
| SWC-108 | State Variable Default Visibility                   | Passed |
| SWC-109 | Uninitialized Storage Pointer                       | Passed |
| SWC-110 | Assert Violation                                    | Passed |
| SWC-111 | Use of Deprecated Solidity Functions                | Passed |
| SWC-112 | Delegatecall to Untrusted Callee                    | Passed |
| SWC-113 | DoS with Failed Call                                | Passed |
| SWC-114 | Transaction Order Dependence                        | Passed |
| SWC-115 | Authorization through tx.origin                     | Passed |
| SWC-116 | Block values as a proxy for time                    | Passed |
| SWC-117 | Signature Malleability                              | Passed |
| SWC-118 | Incorrect Constructor Name                          | Passed |
| SWC-119 | Shadowing State Variables                           | Passed |
| SWC-120 | Weak Sources of Randomness from Chain Attributes    | Passed |
| SWC-121 | Missing Protection against Signature Replay Attacks | Passed |
| SWC-122 | Lack of Proper Signature Verification               | Passed |

| ID      | Description                                      | Status   |
|---------|--|----------|
| SWC-123 | Requirement Violation                            | ● Passed |
| SWC-124 | Write to Arbitrary Storage Location              | ● Passed |
| SWC-125 | Incorrect Inheritance Order                      | ● Passed |
| SWC-126 | Insufficient Gas Griefing                        | ● Passed |
| SWC-127 | Arbitrary Jump with Function Type Variable       | ● Passed |
| SWC-128 | DoS With Block Gas Limit                         | ● Passed |
| SWC-129 | Typographical Error                              | ● Passed |
| SWC-130 | Right-To-Left-Override control character(U+202E) | ● Passed |
| SWC-131 | Presence of unused variables                     | ● Passed |
| SWC-132 | Unexpected Ether balance                         | ● Passed |
| SWC-133 | Hash Collisions With Multiple Variable Arguments | ● Passed |
| SWC-134 | Message call with hardcoded gas amount           | ● Passed |
| SWC-135 | Code With No Effects                             | ● Passed |
| SWC-136 | Unencrypted Private Data On-Chain                | ● Passed |

## CW ASSESSMENT

Not Financial Advice nsurance

ContractWolf Vulnerability and Security Tests

| ID     | Name                     | Description  | Status |
|--------|--------------------------|--|--------|
| CW-001 | Multiple Version         | Presence of multiple compiler version across all contracts   | ✓      |
| CW-002 | Incorrect Access Control | Additional checks for critical logic and flow  | ✓      |
| CW-003 | Payable Contract         | A function to withdraw ether should exist otherwise the ether will be trapped  | ✓      |
| CW-004 | Custom Modifier          | major recheck for custom modifier logic  | ✓      |
| CW-005 | Divide Before Multiply   | Performing multiplication before division is generally better to avoid loss of precision   | ✓      |
| CW-006 | Multiple Calls           | Functions with multiple internal calls   | ✓      |
| CW-007 | Deprecated Keywords      | Use of deprecated functions/operators such as block.blockhash() for blockhash(), msg.gas for gasleft(), throw for revert(), sha3() for keccak256(), callcode() for delegatecall(), suicide() for selfdestruct(), constant for view or var for actual type name should be avoided to prevent unintended errors with newer compiler versions | ✗      |
| CW-008 | Unused Contract          | Presence of an unused, unimported or uncalled contract   | ✓      |
| CW-009 | Assembly Usage           | Use of EVM assembly is error-prone and should be avoided or double-checked for correctness   | ✓      |
| CW-010 | Similar Variable Names   | Variables with similar names could be confused for each other and therefore should be avoided  | ✓      |
| CW-011 | Commented Code           | Removal of commented/unused code lines   | ✓      |
| CW-012 | SafeMath Override        | SafeMath is no longer needed starting with Solidity v0.8+. The compiler now has built-in overflow checking.  | ✓      |

## FIXES & RECOMMENDATION

### CW-007 | Deprecated Keywords

Code from L 276 uses a deprecated keyword

```
_setupRole(MANAGER_ROLE, msg.sender); // Setting up the manager role  
  
_setRoleAdmin(AUTHENTICATOR_ROLE, MANAGER_ROLE); // Manager can manage authenticators  
  
grantRole(AUTHENTICATOR_ROLE, msg.sender); // Adding the deployer as an authenticator
```

`_setupRole` keyword has been deprecated in [OpenZeppelin v5.0.0](#)

Use `_grantRole` instead

Recommended fix

```
_grantRole(MANAGER_ROLE, msg.sender); // Setting up the manager role  
  
_setRoleAdmin(AUTHENTICATOR_ROLE, MANAGER_ROLE); // Manager can manage authenticators  
  
_grantRole(AUTHENTICATOR_ROLE, msg.sender); // Adding the deployer as an authenticator
```

## AUDIT COMMENTS

Not Financial Advice insurance

Smart Contract audit comment for a non-technical perspective

- Owner can set taxes up to 10% only
- Owner can grant roles(manager & authenticator)
- Owner can set a new router
- Owner can add pairs
- Contract cannot be paused
- Owner cannot renounce and transfer ownership
- Owner cannot burn tokens
- Owner cannot mint after initial deployment
- Owner can set max transaction limit not lower than 0.1%
- Owner cannot block users



# CONTRACTWOLF

Blockchain Security - Smart Contract Audits