

# SMART CONTRACT

---

## Security Audit Report

Customer:	Sphynx Network
Website:	<a href="https://sphynx.network">https://sphynx.network</a>
Platform:	Binance Smart Chain
Language:	Solidity
Date:	June 14th, 2021

# Table of contents

Introduction .....	4
Project Background .....	4
Audit Scope .....	4
Claimed Smart Contract Features .....	5
Audit Summary .....	5
Technical Quick Stats .....	6
Code Quality .....	7
Documentation .....	7
Use of Dependencies .....	7
AS-IS overview .....	8
Severity Definitions .....	11
Audit Findings .....	11
Conclusion .....	16
Our Methodology .....	17
Disclaimers .....	19
Appendix	
• Code Flow Diagram .....	20
• Slither Report Log .....	21

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO PUBLIC AFTER ISSUES ARE RESOLVED.

# Introduction

We were contracted by the Sphynx Network team to perform the Security audit of the SphynxNetwork Token smart contract code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on June 14th, 2021.

**The purpose of this audit was to address the following:**

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

## Project Background

Sphynx Network is a disruptive project targeting to use the full potential of NFTs into the DeFi space. NFTs will make staking more dynamic and flexible by Implementing the transferable function that unties the LP Stake from the user's account.

## Audit scope

<b>Name</b>	<b>Code Review and Security Analysis Report for Sphynx Network Token (SPH) Smart Contract</b>
<b>Platform</b>	<b>BSC / Solidity</b>
<b>File</b>	SphynxNetwork.sol
<b>Smart Contract Online Code</b>	<a href="https://bscscan.com/address/0x8bAc6b4AF65C8c1967a0FBc27cd37FD6059daa00#code">https://bscscan.com/address/0x8bAc6b4AF65C8c1967a0FBc27cd37FD6059daa00#code</a>
<b>File MD5 Hash</b>	F4E5921C94CDC1D0E25BEA5DA4300C38
<b>Audit Date</b>	June 14th, 2021

## Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Name: Sphynx Network Token	<b>YES, This is valid.</b>
Symbol: SPH	<b>YES, This is valid.</b>
Decimals: 18	<b>YES, This is valid.</b>
Minting of tokens	<b>No additional minting possible.</b>
Tax fee: 5% Liquidity fee: 5% Burned Tokens: 10%	<b>YES, This is valid. Owner can change these fees.</b>
Max Transaction Amount: 5 Trillion	<b>YES, This is valid. Owner can change this value.</b>

## Audit Summary

According to the standard audit assessment, Customer's solidity smart contract is **secured**. These contracts also have owner functions (described in the centralization section below), which does not make everything 100% decentralized. Thus, the owner must execute those smart contract functions as per the business plan.

Insecure	Poor secured	Secure	Well-secured
----------	--------------	--------	--------------

You are here



We used various tools like MythX, Slither and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.

All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

**We found 0 critical, 0 high, 0 medium and 5 low and some very low level issues.**

## Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Moderated
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Moderated
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	Passed
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Other code specification issues	Passed
Gas Optimization	"Out of Gas" Issue	Passed
	High consumption 'for/while' loop	Moderated
	High consumption 'storage' storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	"Short Address" Attack	Passed
	"Double Spend" Attack	Passed

**Overall Audit Result: PASSED**

## Code Quality

This audit scope has 1 smart contract. This smart contract also contains Libraries, Smart contracts inherits and Interfaces. This is a compact and well written contract.

The libraries in the Sphynx Network Token are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the Sphynx Network Token.

The Sphynx Network team has **not** provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Some code parts are **not well** commented on smart contracts.

## Documentation

We were given Sphynx Network Token smart contract code in the form of a BscScan web link. The hashes of that code are mentioned above in the table.

As mentioned above, some code parts are **not well** commented. So it is difficult to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official website <https://sphynx.network/> which provided rich information about the project architecture and tokenomics.

## Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects. And their core code blocks are written well.

Apart from libraries, its functions are used in external smart contract calls.

# AS-IS overview

Sphynx Network token is a smart contract, having functionality like liquidity, staking, etc.

## SphynxNetwork.sol

### (1) Interface

- (a) IERC20
- (b) IUniswapV2Factory
- (c) IUniswapV2Pair
- (d) IUniswapV2Router01
- (e) IUniswapV2Router02

### (2) Inherited contracts

- (a) Context
- (b) Ownable
- (c) IERC20

### (3) Usages

- (a) using SafeMath for uint256;
- (b) using Address for address;

### (4) Events

- (a) event Transfer(address indexed from, address indexed to, uint256 value);
- (b) event Approval(address indexed owner, address indexed spender, uint256 value);
- (c) event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);
- (d) event PairCreated(address indexed token0, address indexed token1, address pair, uint);
- (e) event Approval(address indexed owner, address indexed spender, uint value);
- (f) event Transfer(address indexed from, address indexed to, uint value);
- (g) event Mint(address indexed sender, uint amount0, uint amount1);
- (h) event Burn(address indexed sender, uint amount0, uint amount1, address indexed to);



- (i) event Swap(address indexed sender, uint amount0In, uint amount1In, uint amount0Out, uint amount1Out, address indexed to);
- (j) event Sync(uint112 reserve0, uint112 reserve1);
- (k) event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
- (l) event SwapAndLiquifyEnabledUpdated(bool enabled);
- (m) event SwapAndLiquify(uint256 tokensSwapped, uint256 ethReceived, uint256 tokensIntoLiquidity);

## (5) Functions

Sl.	Functions	Type	Observation	Conclusion
1	lockTheSwap	modifier	Passed	No Issue
2	updateDex	read	Passed	No Issue
3	updateTokenLimit	write	access only Owner	No Issue
4	name	read	Passed	No Issue
5	symbol	read	Passed	No Issue
6	decimals	read	Passed	No Issue
7	totalSupply	read	Passed	No Issue
8	balanceOf	read	Passed	No Issue
9	transfer	write	Passed	No Issue
10	allowance	read	Passed	No Issue
11	approve	write	Passed	No Issue
12	transferFrom	write	Passed	No Issue
13	increaseAllowance	write	Passed	No Issue
14	decreaseAllowance	write	Passed	No Issue
15	isExcludedFromReward	read	Passed	No Issue
16	totalFees	read	Passed	No Issue
17	deliver	write	external instead of public	Refer Audit Findings
18	reflectionFromToken	read	external instead of public	Refer Audit Findings
19	tokenFromReflection	read	Passed	No Issue
20	excludeFromReward	write	access only Owner	No Issue
21	includeInReward	external	Infinite loop possibility	Refer Audit Findings
22	transferBothExcluded	write	Passed	No Issue
23	excludeFromFee	write	Missing Events	No Issue
24	includeInFee	write	Missing Events	No Issue
25	setTaxFeePercent	external	Missing Events	No Issue
26	setLiquidityFeePercent	external	Missing Events	No Issue
27	setMaxTxPercent	external	Missing Events	No Issue
28	setSwapAndLiquifyEnabled	write	Missing Events	No Issue

29	receive	write	Passed	No Issue
30	_reflectFee	write	Passed	No Issue
31	_getValues	read	Passed	No Issue
32	_getTVValues	read	Passed	No Issue
33	_getRVValues	write	Passed	No Issue
34	_getRate	read	Passed	No Issue
35	_getCurrentSupply	read	Passed	No Issue
36	_takeLiquidity	write	Passed	No Issue
37	calculateTaxFee	read	Passed	No Issue
38	calculateLiquidityFee	read	Passed	No Issue
39	removeAllFee	write	Passed	No Issue
40	restoreAllFee	write	Passed	No Issue
41	isExcludedFromFee	read	Passed	No Issue
42	_approve	write	Passed	No Issue
43	_transfer	write	Passed	No Issue
44	swapAndLiquify	write	Passed	No Issue
45	swapTokensForEth	write	Passed	No Issue
46	addLiquidity	write	Centralized risk in addLiquidity	Refer Audit Findings
47	_tokenTransfer	write	Passed	No Issue
48	_transferStandard	write	Passed	No Issue
49	_transferToExcluded	write	Passed	No Issue
50	_transferFromExcluded	write	Passed	No Issue
51	owner	read	Passed	No Issue
52	onlyOwner	modifier	Passed	No Issue
53	renounceOwnership	write	access only Owner	No Issue
54	transferOwnership	write	access only Owner	No Issue
55	geUnlockTime	read	Passed	No Issue
56	lock	write	access only Owner	No Issue
57	unlock	write	Ownership can be regained	Refer Audit Findings
58	_msgSender	read	Passed	No Issue
59	_msgData	read	Passed	No Issue

## Severity Definitions

Risk Level	Description
<b>Critical</b>	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
<b>High</b>	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
<b>Medium</b>	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
<b>Low</b>	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
<b>Lowest / Code Style / Best Practice</b>	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

## Audit Findings

### Critical

No critical severity vulnerabilities were found.

### High

No high severity vulnerabilities were found.

### Medium

No Medium severity vulnerabilities were found.

## Low

(1) Make variables constant:

```
string private _name = "Sphynx Network";  
string private _symbol = "SPH";  
uint8 private _decimals = 18;
```

Following variables will be unchanged. So, please make it constant. It will save some gas.

- name
- symbol
- decimals

**Resolution:** Declare those variables as constant. Just put a constant keyword.

(2) Possibility of high gas consumption.

```
function includeInReward(address account) external onlyOwner() {  
    require(!_isExcluded[account], "Account is already excluded");  
    for (uint256 i = 0; i < _excluded.length; i++) {  
        if (_excluded[i] == account) {  
            _excluded[i] = _excluded[_excluded.length - 1];  
            _tOwned[account] = 0;  
            _isExcluded[account] = false;  
            _excluded.pop();  
            break;  
        }  
    }  
}
```

If there are so many excluded wallets, then this logic will fail, as it might hit the block's gas limit. If there are very limited exceptions, then this will work, but will cost more gas.

**Resolution:** We suggest excluding limited wallets only.

(3) Centralized risk in addLiquidity:

```
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
    // approve token transfer to cover all possible scenarios
    _approve(address(this), address(uniswapV2Router), tokenAmount);

    // add the liquidity
    uniswapV2Router.addLiquidityETH{value: ethAmount}(
        address(this),
        tokenAmount,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        owner(),
        block.timestamp
    );
}
```

In addLiquidityETH function, owner() gets SPH Tokens from the Pool. At some time, The owner will accumulate significant SPH tokens. If the \_owner is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project.

**Resolution:** Ideally, owner() can be replaced by a contract address, who has governance functionality. Otherwise, the owner must acknowledge this issue and handle the private key of the owner's wallet very securely.

(4) Possible to gain ownership after renouncing the contract ownership. Owner can renounce ownership and make contract without owner but he can regain ownership by following the steps below:

- Owner calls the lock function in contract to set the current owner as \_previousOwner.
- Owner calls unlock to unlock contract and set \_owner = \_previousOwner.
- Owner called renounceOwnership to leave the contract without the owner.
- Owner calls unlock to regain ownership.

**Resolution:** We suggest removing these lock/unlock functions as this seems not serving a great purpose. Otherwise, always renounce ownership first before calling the lock function.

## (5) Missing Events:

Missing Events log for some functions:

- excludeFromFee
- excludeFromReward
- includeInFee
- includeInReward
- setLiquidityFeePercent
- setMaxTxPercent
- setTaxFeePercent
- setSwapAndLiquifyEnabled

## Very Low / Discussion / Best practices:

### (1) Use latest solidity version:

```
pragma solidity ^0.6.12;
```

Using the latest solidity will prevent any compiler level bugs.

**Resolution:** Please use 0.8.5 which is the latest version.

### (2) external instead of public:

```
function deliver(uint256 tAmount) public {  
    address sender = _msgSender();  
    require(!_isExcluded[sender], "Excluded addresses cannot call this function");  
    (uint256 rAmount,,,,) = _getValues(tAmount);  
    _rOwned[sender] = _rOwned[sender].sub(rAmount);  
    _rTotal = _rTotal.sub(rAmount);  
    _tFeeTotal = _tFeeTotal.add(tAmount);  
}  
  
function reflectionFromToken(uint256 tAmount, bool deductTransferFee) public view returns(uint256) {  
    require(tAmount <= _tTotal, "Amount must be less than supply");  
    if (!deductTransferFee) {  
        (uint256 rAmount,,,,) = _getValues(tAmount);  
        return rAmount;  
    } else {  
        (uint256 rTransferAmount,,,,) = _getValues(tAmount);  
        return rTransferAmount;  
    }  
}
```

If any function is not called from inside the smart contract, then it is better to declare it as external instead of public. As it saves some gas as well.

<https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices>

**Resolution:** Make all public to external, which are not called from inside the contract.

## Centralization

This smart contract has some functions which can be executed by Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

- updateDex: can be updated DexAddress by Owner.
- updateTokenLimit: can be allowed to change the 'numTokensSellToAddToLiquidity'.
- excludeFromReward: Owner can check if the account is excluded.
- includeInReward: Owner can include rewards.
- excludeFromFee: Owner can check exclude Fee.
- includeInFee: Owner can check include Fee.
- setTaxFeePercent: Owner can set tax Fee Percentage.
- setLiquidityFeePercent: Owner can set Liquidity Fee Percentage.
- setMaxTxPercent: Owner can set maximum TxAmount.
- setSwapAndLiquifyEnabled: Owner can set swap and Liquify enable.

## Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We observed some issues in the smart contracts and those are fixed/acknowledged in the smart contracts. **So it is good to go for the production.**

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high level description of functionality was presented in As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is **"Secured"**.



# Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

## **Manual Code Review:**

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

## **Vulnerability Analysis:**

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

## **Documenting Results:**

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

## **Suggested Solutions:**

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

# Disclaimers

## EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

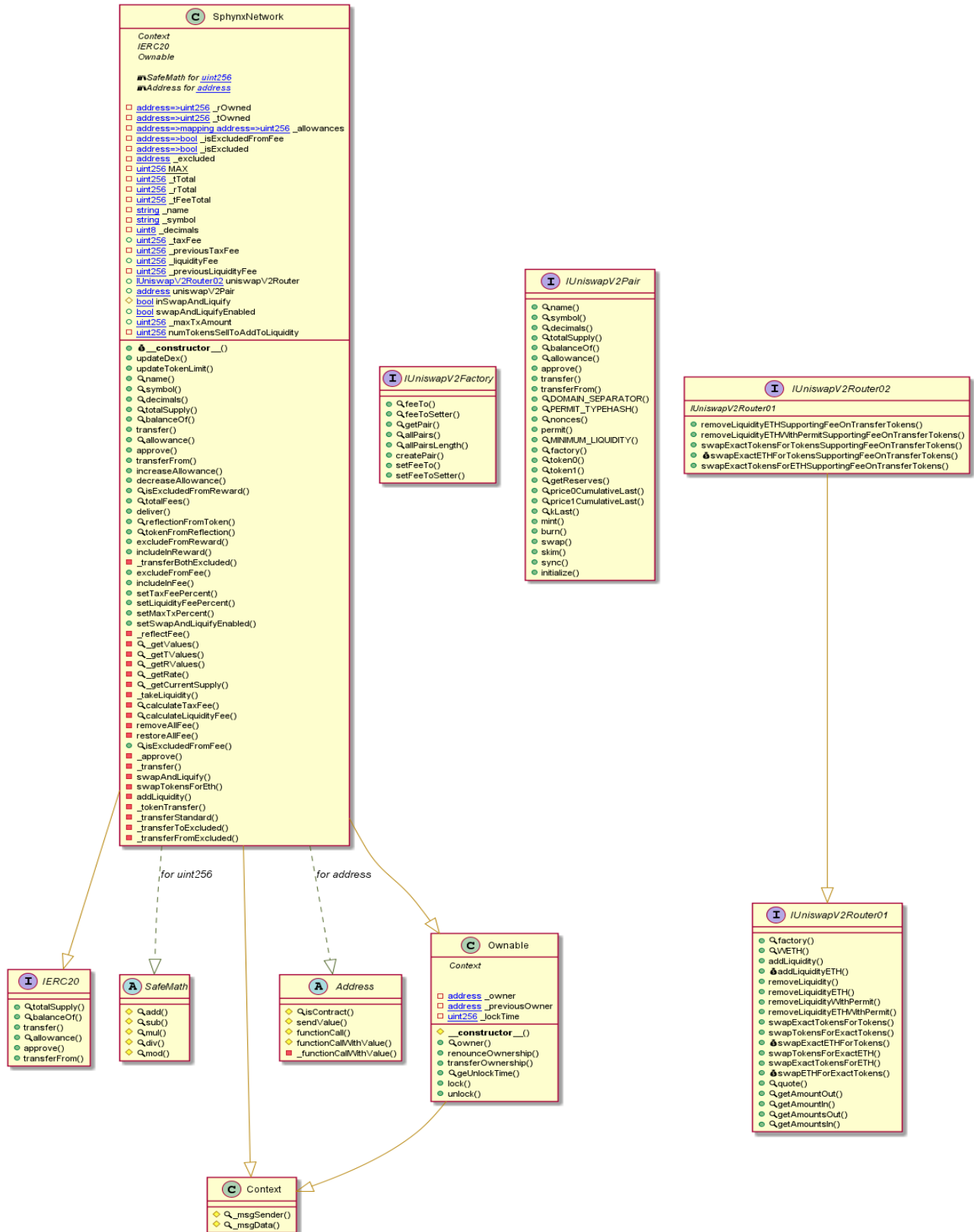
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

## Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can't guarantee explicit security of the audited smart contracts.

# Appendix

## Code Flow Diagram - SphynxNetwork (SPH) Token



This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: [audit@EtherAuthority.io](mailto:audit@EtherAuthority.io)

# Slither Results Log

## Slither log >> SphynxNetwork.sol

INFO:Detectors:

SphynxNetwork.addLiquidity(uint256,uint256) (SphynxNetwork.sol#1095-1108) sends eth to arbitrary user

Dangerous calls:

- uniswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (SphynxNetwork.sol#1100-1107)

Reference:

<https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations>

INFO:Detectors:

Reentrancy in SphynxNetwork.\_transfer(address,address,uint256) (SphynxNetwork.sol#1008-1052):

External calls:

- swapAndLiquify(contractTokenBalance) (SphynxNetwork.sol#1039)

- uniswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (SphynxNetwork.sol#1100-1107)

-

uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (SphynxNetwork.sol#1086-1092)

External calls sending eth:

- swapAndLiquify(contractTokenBalance) (SphynxNetwork.sol#1039)

- uniswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (SphynxNetwork.sol#1100-1107)

State variables written after the call(s):

- \_tokenTransfer(from,to,amount,takeFee) (SphynxNetwork.sol#1051)

- \_rOwned[address(this)] = \_rOwned[address(this)].add(rLiquidity) (SphynxNetwork.sol#964)

- \_rOwned[sender] = \_rOwned[sender].sub(rAmount) (SphynxNetwork.sol#1133)

- \_rOwned[sender] = \_rOwned[sender].sub(rAmount) (SphynxNetwork.sol#1142)

- \_rOwned[sender] = \_rOwned[sender].sub(rAmount) (SphynxNetwork.sol#880)

- \_rOwned[recipient] = \_rOwned[recipient].add(rTransferAmount) (SphynxNetwork.sol#1134)

- \_rOwned[sender] = \_rOwned[sender].sub(rAmount) (SphynxNetwork.sol#1153)

- \_rOwned[recipient] = \_rOwned[recipient].add(rTransferAmount) (SphynxNetwork.sol#1154)

- \_rOwned[recipient] = \_rOwned[recipient].add(rTransferAmount) (SphynxNetwork.sol#1144)

- \_rOwned[recipient] = \_rOwned[recipient].add(rTransferAmount) (SphynxNetwork.sol#882)

- \_tokenTransfer(from,to,amount,takeFee) (SphynxNetwork.sol#1051)

- \_rTotal = \_rTotal.sub(rFee) (SphynxNetwork.sol#919)

- \_tokenTransfer(from,to,amount,takeFee) (SphynxNetwork.sol#1051)

- \_tOwned[address(this)] = \_tOwned[address(this)].add(tLiquidity) (SphynxNetwork.sol#966)

- \_tOwned[sender] = \_tOwned[sender].sub(tAmount) (SphynxNetwork.sol#1152)

- \_tOwned[sender] = \_tOwned[sender].sub(tAmount) (SphynxNetwork.sol#879)

- \_tOwned[recipient] = \_tOwned[recipient].add(tTransferAmount) (SphynxNetwork.sol#1143)

- \_tOwned[recipient] = \_tOwned[recipient].add(tTransferAmount) (SphynxNetwork.sol#881)

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

INFO:Detectors:

SphynxNetwork.addLiquidity(uint256,uint256) (SphynxNetwork.sol#1095-1108) ignores return value by

uniswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (SphynxNetwork.sol#1100-1107)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return>

INFO:Detectors:

SphynxNetwork.allowance(address,address).owner (SphynxNetwork.sol#796) shadows:

- Ownable.owner() (SphynxNetwork.sol#414-416) (function)

SphynxNetwork.\_approve(address,address,uint256).owner (SphynxNetwork.sol#1000) shadows:

- Ownable.owner() (SphynxNetwork.sol#414-416) (function)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing>

INFO:Detectors:

Reentrancy in SphynxNetwork.\_transfer(address,address,uint256) (SphynxNetwork.sol#1008-1052):

External calls:

- swapAndLiquify(contractTokenBalance) (SphynxNetwork.sol#1039)
- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (SphynxNetwork.sol#1100-1107)
- 
- uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (SphynxNetwork.sol#1086-1092)
- External calls sending eth:
  - swapAndLiquify(contractTokenBalance) (SphynxNetwork.sol#1039)
  - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (SphynxNetwork.sol#1100-1107)
- State variables written after the call(s):
  - \_tokenTransfer(from,to,amount,takeFee) (SphynxNetwork.sol#1051)
    - \_liquidityFee = \_previousLiquidityFee (SphynxNetwork.sol#993)
    - \_liquidityFee = 0 (SphynxNetwork.sol#988)
  - \_tokenTransfer(from,to,amount,takeFee) (SphynxNetwork.sol#1051)
    - \_previousLiquidityFee = \_liquidityFee (SphynxNetwork.sol#985)
  - \_tokenTransfer(from,to,amount,takeFee) (SphynxNetwork.sol#1051)
    - \_previousTaxFee = \_taxFee (SphynxNetwork.sol#984)
  - \_tokenTransfer(from,to,amount,takeFee) (SphynxNetwork.sol#1051)
    - \_tFeeTotal = \_tFeeTotal.add(tFee) (SphynxNetwork.sol#920)
  - \_tokenTransfer(from,to,amount,takeFee) (SphynxNetwork.sol#1051)
    - \_taxFee = \_previousTaxFee (SphynxNetwork.sol#992)
    - \_taxFee = 0 (SphynxNetwork.sol#987)
- Reentrancy in SphynxNetwork.constructor() (SphynxNetwork.sol#736-752):
  - External calls:
    - uniswapV2Pair = IUniswapV2Factory(\_uniswapV2Router.factory()).createPair(address(this),\_uniswapV2Router.WETH()) (SphynxNetwork.sol#741-742)
  - State variables written after the call(s):
    - \_isExcludedFromFee[owner()] = true (SphynxNetwork.sol#748)
    - \_isExcludedFromFee[address(this)] = true (SphynxNetwork.sol#749)
    - uniswapV2Router = \_uniswapV2Router (SphynxNetwork.sol#745)
- Reentrancy in SphynxNetwork.swapAndLiquify(uint256) (SphynxNetwork.sol#1054-1075):
  - External calls:
    - swapTokensForEth(half) (SphynxNetwork.sol#1066)
    -
  - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (SphynxNetwork.sol#1086-1092)
    - addLiquidity(otherHalf,newBalance) (SphynxNetwork.sol#1072)
      - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (SphynxNetwork.sol#1100-1107)
  - External calls sending eth:
    - addLiquidity(otherHalf,newBalance) (SphynxNetwork.sol#1072)
      - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (SphynxNetwork.sol#1100-1107)
  - State variables written after the call(s):
    - addLiquidity(otherHalf,newBalance) (SphynxNetwork.sol#1072)
      - \_allowances[owner][spender] = amount (SphynxNetwork.sol#1004)
- Reentrancy in SphynxNetwork.transferFrom(address,address,uint256) (SphynxNetwork.sol#805-809):
  - External calls:
    - \_transfer(sender,recipient,amount) (SphynxNetwork.sol#806)
      - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (SphynxNetwork.sol#1100-1107)
    -
  - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (SphynxNetwork.sol#1086-1092)
    - External calls sending eth:
      - \_transfer(sender,recipient,amount) (SphynxNetwork.sol#806)
        - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (SphynxNetwork.sol#1100-1107)
    - State variables written after the call(s):
      - \_approve(sender,\_msgSender(),\_allowances[sender][\_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (SphynxNetwork.sol#807)
        - \_allowances[owner][spender] = amount (SphynxNetwork.sol#1004)
- Reentrancy in SphynxNetwork.updateDex(address) (SphynxNetwork.sol#754-762):



External calls:

- `uniswapV2Pair =`

`IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH())` (SphynxNetwork.sol#757-758)

State variables written after the call(s):

- `uniswapV2Router = _uniswapV2Router` (SphynxNetwork.sol#761)

Reference: <https://github.com/cryptic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2>

INFO:Detectors:

Reentrancy in `SphynxNetwork._transfer(address,address,uint256)` (SphynxNetwork.sol#1008-1052):

External calls:

- `swapAndLiquify(contractTokenBalance)` (SphynxNetwork.sol#1039)
- `uniswapV2Router.addLiquidityETH{value:`

`ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp)` (SphynxNetwork.sol#1100-1107)

-

`uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(t`  
`his),block.timestamp)` (SphynxNetwork.sol#1086-1092)

External calls sending eth:

- `swapAndLiquify(contractTokenBalance)` (SphynxNetwork.sol#1039)
- `uniswapV2Router.addLiquidityETH{value:`

`ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp)` (SphynxNetwork.sol#1100-1107)

Event emitted after the call(s):

- `Transfer(sender,recipient,tTransferAmount)` (SphynxNetwork.sol#1137)
- `_tokenTransfer(from,to,amount,takeFee)` (SphynxNetwork.sol#1051)
- `Transfer(sender,recipient,tTransferAmount)` (SphynxNetwork.sol#1147)
- `_tokenTransfer(from,to,amount,takeFee)` (SphynxNetwork.sol#1051)
- `Transfer(sender,recipient,tTransferAmount)` (SphynxNetwork.sol#1157)
- `_tokenTransfer(from,to,amount,takeFee)` (SphynxNetwork.sol#1051)
- `Transfer(sender,recipient,tTransferAmount)` (SphynxNetwork.sol#885)
- `_tokenTransfer(from,to,amount,takeFee)` (SphynxNetwork.sol#1051)

Reentrancy in `SphynxNetwork.constructor()` (SphynxNetwork.sol#736-752):

External calls:

- `uniswapV2Pair =`

`IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH())` (SphynxNetwork.sol#741-742)

Event emitted after the call(s):

- `Transfer(address(0),_msgSender(),_tTotal)` (SphynxNetwork.sol#751)

Reentrancy in `SphynxNetwork.swapAndLiquify(uint256)` (SphynxNetwork.sol#1054-1075):

External calls:

- `swapTokensForEth(half)` (SphynxNetwork.sol#1066)

-

`uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(t`  
`his),block.timestamp)` (SphynxNetwork.sol#1086-1092)

- `addLiquidity(otherHalf,newBalance)` (SphynxNetwork.sol#1072)
- `uniswapV2Router.addLiquidityETH{value:`

`ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp)` (SphynxNetwork.sol#1100-1107)

External calls sending eth:

- `addLiquidity(otherHalf,newBalance)` (SphynxNetwork.sol#1072)
- `uniswapV2Router.addLiquidityETH{value:`

`ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp)` (SphynxNetwork.sol#1100-1107)

Event emitted after the call(s):

- `Approval(owner,spender,amount)` (SphynxNetwork.sol#1005)
- `addLiquidity(otherHalf,newBalance)` (SphynxNetwork.sol#1072)
- `SwapAndLiquify(half,newBalance,otherHalf)` (SphynxNetwork.sol#1074)

Reentrancy in `SphynxNetwork.transferFrom(address,address,uint256)` (SphynxNetwork.sol#805-809):

External calls:

- `_transfer(sender,recipient,amount)` (SphynxNetwork.sol#806)
- `uniswapV2Router.addLiquidityETH{value:`

`ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp)` (SphynxNetwork.sol#1100-1107)

-

`uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(t`  
`his),block.timestamp)` (SphynxNetwork.sol#1086-1092)

External calls sending eth:

- `_transfer(sender,recipient,amount)` (SphynxNetwork.sol#806)
- `uniswapV2Router.addLiquidityETH{value:`

`ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp)` (SphynxNetwork.sol#1100-1107)

Event emitted after the call(s):

- Approval(owner,spender,amount) (SphynxNetwork.sol#1005)
  - \_approve(sender,\_msgSender(),\_allowances[sender][\_msgSender()].sub(amount,ERC20:transfer amount exceeds allowance)) (SphynxNetwork.sol#807)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3>

INFO:Detectors:

Ownable.unlock() (SphynxNetwork.sol#461-466) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(now > \_lockTime,Contract is locked until 7 days) (SphynxNetwork.sol#463)

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

INFO:Detectors:

Address.isContract(address) (SphynxNetwork.sol#266-275) uses assembly

- INLINE ASM (SphynxNetwork.sol#273)

Address.\_functionCallWithValue(address,bytes,uint256,string) (SphynxNetwork.sol#359-380) uses assembly

- INLINE ASM (SphynxNetwork.sol#372-375)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage>

INFO:Detectors:

Address.\_functionCallWithValue(address,bytes,uint256,string) (SphynxNetwork.sol#359-380) is never used and should be removed

Address.functionCall(address,bytes) (SphynxNetwork.sol#319-321) is never used and should be removed

Address.functionCall(address,bytes,string) (SphynxNetwork.sol#329-331) is never used and should be removed

Address.functionCallWithValue(address,bytes,uint256) (SphynxNetwork.sol#344-346) is never used and should be removed

Address.functionCallWithValue(address,bytes,uint256,string) (SphynxNetwork.sol#354-357) is never used and should be removed

Address.isContract(address) (SphynxNetwork.sol#266-275) is never used and should be removed

Address.sendValue(address,uint256) (SphynxNetwork.sol#293-299) is never used and should be removed

Context.\_msgData() (SphynxNetwork.sol#238-241) is never used and should be removed

SafeMath.mod(uint256,uint256) (SphynxNetwork.sol#211-213) is never used and should be removed

SafeMath.mod(uint256,uint256,string) (SphynxNetwork.sol#227-230) is never used and should be removed

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code>

INFO:Detectors:

SphynxNetwork.\_rTotal (SphynxNetwork.sol#698) is set pre-construction with a non-constant function or state variable:

- (MAX - (MAX % \_tTotal))

SphynxNetwork.\_previousTaxFee (SphynxNetwork.sol#706) is set pre-construction with a non-constant function or state variable:

- \_taxFee

SphynxNetwork.\_previousLiquidityFee (SphynxNetwork.sol#709) is set pre-construction with a non-constant function or state variable:

- \_liquidityFee

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state-variables>

INFO:Detectors:

Low level call in Address.sendValue(address,uint256) (SphynxNetwork.sol#293-299):

- (success) = recipient.call{value: amount}() (SphynxNetwork.sol#297)

Low level call in Address.\_functionCallWithValue(address,bytes,uint256,string) (SphynxNetwork.sol#359-380):

- (success,returndata) = target.call{value: weiValue}(data) (SphynxNetwork.sol#363)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls>

INFO:Detectors:

Function IUniswapV2Pair.DOMAIN\_SEPARATOR() (SphynxNetwork.sol#505) is not in mixedCase

Function IUniswapV2Pair.PERMIT\_TYPEHASH() (SphynxNetwork.sol#506) is not in mixedCase

Function IUniswapV2Pair.MINIMUM\_LIQUIDITY() (SphynxNetwork.sol#523) is not in mixedCase

Function IUniswapV2Router01.WETH() (SphynxNetwork.sol#545) is not in mixedCase

Parameter SphynxNetwork.updateDex(address).\_newDexAddress (SphynxNetwork.sol#754) is not in mixedCase

Parameter SphynxNetwork.setSwapAndLiquifyEnabled(bool).\_enabled (SphynxNetwork.sol#910) is not in mixedCase

Parameter SphynxNetwork.calculateTaxFee(uint256).\_amount (SphynxNetwork.sol#969) is not in mixedCase

Parameter SphynxNetwork.calculateLiquidityFee(uint256).\_amount (SphynxNetwork.sol#975) is not in mixedCase

Variable SphynxNetwork.\_taxFee (SphynxNetwork.sol#705) is not in mixedCase

Variable SphynxNetwork.\_liquidityFee (SphynxNetwork.sol#708) is not in mixedCase



Variable SphynxNetwork.\_maxTxAmount (SphynxNetwork.sol#717) is not in mixedCase  
Reference:  
<https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions>  
INFO:Detectors:  
Redundant expression "this (SphynxNetwork.sol#239)" inContext (SphynxNetwork.sol#233-242)  
Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements>  
INFO:Detectors:  
Variable  
IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amount  
tADesired (SphynxNetwork.sol#550) is too similar to  
IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amount  
tBDesired (SphynxNetwork.sol#551)  
Variable SphynxNetwork.\_transferStandard(address,address,uint256).rTransferAmount  
(SphynxNetwork.sol#1132) is too similar to  
SphynxNetwork.\_transferStandard(address,address,uint256).tTransferAmount (SphynxNetwork.sol#1132)  
Variable SphynxNetwork.\_transferFromExcluded(address,address,uint256).rTransferAmount  
(SphynxNetwork.sol#1151) is too similar to  
SphynxNetwork.\_transferFromExcluded(address,address,uint256).tTransferAmount  
(SphynxNetwork.sol#1151)  
Variable SphynxNetwork.\_transferFromExcluded(address,address,uint256).rTransferAmount  
(SphynxNetwork.sol#1151) is too similar to  
SphynxNetwork.\_transferBothExcluded(address,address,uint256).tTransferAmount  
(SphynxNetwork.sol#878)  
Variable SphynxNetwork.\_transferToExcluded(address,address,uint256).rTransferAmount  
(SphynxNetwork.sol#1141) is too similar to  
SphynxNetwork.\_transferFromExcluded(address,address,uint256).tTransferAmount  
(SphynxNetwork.sol#1151)  
Variable SphynxNetwork.\_transferStandard(address,address,uint256).rTransferAmount  
(SphynxNetwork.sol#1132) is too similar to  
SphynxNetwork.\_transferFromExcluded(address,address,uint256).tTransferAmount  
(SphynxNetwork.sol#1151)  
Variable SphynxNetwork.\_transferBothExcluded(address,address,uint256).rTransferAmount  
(SphynxNetwork.sol#878) is too similar to  
SphynxNetwork.\_transferBothExcluded(address,address,uint256).tTransferAmount  
(SphynxNetwork.sol#878)  
Variable SphynxNetwork.\_transferStandard(address,address,uint256).rTransferAmount  
(SphynxNetwork.sol#1132) is too similar to  
SphynxNetwork.\_transferBothExcluded(address,address,uint256).tTransferAmount  
(SphynxNetwork.sol#878)  
Variable SphynxNetwork.reflectionFromToken(uint256,bool).rTransferAmount (SphynxNetwork.sol#844) is  
too similar to SphynxNetwork.\_transferFromExcluded(address,address,uint256).tTransferAmount  
(SphynxNetwork.sol#1151)  
Variable SphynxNetwork.\_transferToExcluded(address,address,uint256).rTransferAmount  
(SphynxNetwork.sol#1141) is too similar to  
SphynxNetwork.\_transferStandard(address,address,uint256).tTransferAmount (SphynxNetwork.sol#1132)  
Variable SphynxNetwork.\_transferToExcluded(address,address,uint256).rTransferAmount  
(SphynxNetwork.sol#1141) is too similar to SphynxNetwork.\_getTValues(uint256).tTransferAmount  
(SphynxNetwork.sol#932)  
Variable SphynxNetwork.\_transferToExcluded(address,address,uint256).rTransferAmount  
(SphynxNetwork.sol#1141) is too similar to  
SphynxNetwork.\_transferToExcluded(address,address,uint256).tTransferAmount (SphynxNetwork.sol#1141)  
Variable SphynxNetwork.\_getRValues(uint256,uint256,uint256,uint256).rTransferAmount  
(SphynxNetwork.sol#940) is too similar to  
SphynxNetwork.\_transferFromExcluded(address,address,uint256).tTransferAmount  
(SphynxNetwork.sol#1151)  
Variable SphynxNetwork.\_getValues(uint256).rTransferAmount (SphynxNetwork.sol#925) is too similar to  
SphynxNetwork.\_transferStandard(address,address,uint256).tTransferAmount (SphynxNetwork.sol#1132)  
Variable SphynxNetwork.\_transferToExcluded(address,address,uint256).rTransferAmount  
(SphynxNetwork.sol#1141) is too similar to SphynxNetwork.\_getValues(uint256).tTransferAmount  
(SphynxNetwork.sol#924)  
Variable SphynxNetwork.\_getRValues(uint256,uint256,uint256,uint256).rTransferAmount  
(SphynxNetwork.sol#940) is too similar to  
SphynxNetwork.\_transferBothExcluded(address,address,uint256).tTransferAmount  
(SphynxNetwork.sol#878)

Variable SphynxNetwork.reflectionFromToken(uint256,bool).rTransferAmount (SphynxNetwork.sol#844) is too similar to SphynxNetwork.\_transferStandard(address,address,uint256).tTransferAmount (SphynxNetwork.sol#1132)

Variable SphynxNetwork.\_transferToExcluded(address,address,uint256).rTransferAmount (SphynxNetwork.sol#1141) is too similar to SphynxNetwork.\_transferBothExcluded(address,address,uint256).tTransferAmount (SphynxNetwork.sol#878)

Variable SphynxNetwork.reflectionFromToken(uint256,bool).rTransferAmount (SphynxNetwork.sol#844) is too similar to SphynxNetwork.\_transferBothExcluded(address,address,uint256).tTransferAmount (SphynxNetwork.sol#878)

Variable SphynxNetwork.\_getRValues(uint256,uint256,uint256,uint256).rTransferAmount (SphynxNetwork.sol#940) is too similar to SphynxNetwork.\_getTValues(uint256).tTransferAmount (SphynxNetwork.sol#932)

Variable SphynxNetwork.\_getRValues(uint256,uint256,uint256,uint256).rTransferAmount (SphynxNetwork.sol#940) is too similar to SphynxNetwork.\_transferStandard(address,address,uint256).tTransferAmount (SphynxNetwork.sol#1132)

Variable SphynxNetwork.\_getValues(uint256).rTransferAmount (SphynxNetwork.sol#925) is too similar to SphynxNetwork.\_transferFromExcluded(address,address,uint256).tTransferAmount (SphynxNetwork.sol#1151)

Variable SphynxNetwork.\_transferBothExcluded(address,address,uint256).rTransferAmount (SphynxNetwork.sol#878) is too similar to SphynxNetwork.\_transferFromExcluded(address,address,uint256).tTransferAmount (SphynxNetwork.sol#1151)

Variable SphynxNetwork.\_getValues(uint256).rTransferAmount (SphynxNetwork.sol#925) is too similar to SphynxNetwork.\_transferBothExcluded(address,address,uint256).tTransferAmount (SphynxNetwork.sol#878)

Variable SphynxNetwork.\_getRValues(uint256,uint256,uint256,uint256).rTransferAmount (SphynxNetwork.sol#940) is too similar to SphynxNetwork.\_getValues(uint256).tTransferAmount (SphynxNetwork.sol#924)

Variable SphynxNetwork.reflectionFromToken(uint256,bool).rTransferAmount (SphynxNetwork.sol#844) is too similar to SphynxNetwork.\_getValues(uint256).tTransferAmount (SphynxNetwork.sol#924)

Variable SphynxNetwork.\_transferBothExcluded(address,address,uint256).rTransferAmount (SphynxNetwork.sol#878) is too similar to SphynxNetwork.\_getValues(uint256).tTransferAmount (SphynxNetwork.sol#924)

Variable SphynxNetwork.\_getValues(uint256).rTransferAmount (SphynxNetwork.sol#925) is too similar to SphynxNetwork.\_getTValues(uint256).tTransferAmount (SphynxNetwork.sol#932)

Variable SphynxNetwork.\_transferFromExcluded(address,address,uint256).rTransferAmount (SphynxNetwork.sol#1151) is too similar to SphynxNetwork.\_getValues(uint256).tTransferAmount (SphynxNetwork.sol#924)

Variable SphynxNetwork.\_getRValues(uint256,uint256,uint256,uint256).rTransferAmount (SphynxNetwork.sol#940) is too similar to SphynxNetwork.\_transferToExcluded(address,address,uint256).tTransferAmount (SphynxNetwork.sol#1141)

Variable SphynxNetwork.\_transferFromExcluded(address,address,uint256).rTransferAmount (SphynxNetwork.sol#1151) is too similar to SphynxNetwork.\_getTValues(uint256).tTransferAmount (SphynxNetwork.sol#932)

Variable SphynxNetwork.\_transferFromExcluded(address,address,uint256).rTransferAmount (SphynxNetwork.sol#1151) is too similar to SphynxNetwork.\_transferStandard(address,address,uint256).tTransferAmount (SphynxNetwork.sol#1132)

Variable SphynxNetwork.\_getValues(uint256).rTransferAmount (SphynxNetwork.sol#925) is too similar to SphynxNetwork.\_getValues(uint256).tTransferAmount (SphynxNetwork.sol#924)

Variable SphynxNetwork.\_transferBothExcluded(address,address,uint256).rTransferAmount (SphynxNetwork.sol#878) is too similar to SphynxNetwork.\_getTValues(uint256).tTransferAmount (SphynxNetwork.sol#932)

Variable SphynxNetwork.\_transferBothExcluded(address,address,uint256).rTransferAmount (SphynxNetwork.sol#878) is too similar to SphynxNetwork.\_transferStandard(address,address,uint256).tTransferAmount (SphynxNetwork.sol#1132)

Variable SphynxNetwork.reflectionFromToken(uint256,bool).rTransferAmount (SphynxNetwork.sol#844) is too similar to SphynxNetwork.\_getTValues(uint256).tTransferAmount (SphynxNetwork.sol#932)

Variable SphynxNetwork.\_transferBothExcluded(address,address,uint256).rTransferAmount (SphynxNetwork.sol#878) is too similar to SphynxNetwork.\_transferToExcluded(address,address,uint256).tTransferAmount (SphynxNetwork.sol#1141)

Variable SphynxNetwork.\_transferStandard(address,address,uint256).rTransferAmount (SphynxNetwork.sol#1132) is too similar to SphynxNetwork.\_transferToExcluded(address,address,uint256).tTransferAmount (SphynxNetwork.sol#1141)

Variable SphynxNetwork.\_transferStandard(address,address,uint256).rTransferAmount (SphynxNetwork.sol#1132) is too similar to SphynxNetwork.\_getTValues(uint256).tTransferAmount (SphynxNetwork.sol#932)

Variable SphynxNetwork.reflectionFromToken(uint256,bool).rTransferAmount (SphynxNetwork.sol#844) is too similar to SphynxNetwork.\_transferToExcluded(address,address,uint256).tTransferAmount (SphynxNetwork.sol#1141)

Variable SphynxNetwork.\_transferFromExcluded(address,address,uint256).rTransferAmount (SphynxNetwork.sol#1151) is too similar to SphynxNetwork.\_transferToExcluded(address,address,uint256).tTransferAmount (SphynxNetwork.sol#1141)

Variable SphynxNetwork.\_transferStandard(address,address,uint256).rTransferAmount (SphynxNetwork.sol#1132) is too similar to SphynxNetwork.\_getTValues(uint256).tTransferAmount (SphynxNetwork.sol#924)

Variable SphynxNetwork.\_getTValues(uint256).rTransferAmount (SphynxNetwork.sol#925) is too similar to SphynxNetwork.\_transferToExcluded(address,address,uint256).tTransferAmount (SphynxNetwork.sol#1141)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar>

INFO:Detectors:

SphynxNetwork.slitherConstructorVariables() (SphynxNetwork.sol#683-1161) uses literals with too many digits:

- \_tTotal = 1000000 \* 10 \*\* 6 \* 10 \*\* 18 (SphynxNetwork.sol#697)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits>

INFO:Detectors:

SphynxNetwork.\_decimals (SphynxNetwork.sol#703) should be constant

SphynxNetwork.\_name (SphynxNetwork.sol#701) should be constant

SphynxNetwork.\_symbol (SphynxNetwork.sol#702) should be constant

SphynxNetwork.\_tTotal (SphynxNetwork.sol#697) should be constant

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant>

INFO:Detectors:

renounceOwnership() should be declared external:

- Ownable.renounceOwnership() (SphynxNetwork.sol#433-436)

transferOwnership(address) should be declared external:

- Ownable.transferOwnership(address) (SphynxNetwork.sol#442-446)

geUnlockTime() should be declared external:

- Ownable.geUnlockTime() (SphynxNetwork.sol#448-450)

lock(uint256) should be declared external:

- Ownable.lock(uint256) (SphynxNetwork.sol#453-458)

unlock() should be declared external:

- Ownable.unlock() (SphynxNetwork.sol#461-466)

updateTokenLimit(uint256) should be declared external:

- SphynxNetwork.updateTokenLimit(uint256) (SphynxNetwork.sol#765-768)

name() should be declared external:

- SphynxNetwork.name() (SphynxNetwork.sol#770-772)

symbol() should be declared external:

- SphynxNetwork.symbol() (SphynxNetwork.sol#774-776)

decimals() should be declared external:

- SphynxNetwork.decimals() (SphynxNetwork.sol#778-780)

totalSupply() should be declared external:

- SphynxNetwork.totalSupply() (SphynxNetwork.sol#782-784)

transfer(address,uint256) should be declared external:

- SphynxNetwork.transfer(address,uint256) (SphynxNetwork.sol#791-794)

allowance(address,address) should be declared external:

- SphynxNetwork.allowance(address,address) (SphynxNetwork.sol#796-798)

approve(address,uint256) should be declared external:

- SphynxNetwork.approve(address,uint256) (SphynxNetwork.sol#800-803)

transferFrom(address,address,uint256) should be declared external:

- SphynxNetwork.transferFrom(address,address,uint256) (SphynxNetwork.sol#805-809)

increaseAllowance(address,uint256) should be declared external:

- SphynxNetwork.increaseAllowance(address,uint256) (SphynxNetwork.sol#811-814)

decreaseAllowance(address,uint256) should be declared external:

- SphynxNetwork.decreaseAllowance(address,uint256) (SphynxNetwork.sol#816-819)

isExcludedFromReward(address) should be declared external:

- SphynxNetwork.isExcludedFromReward(address) (SphynxNetwork.sol#821-823)

totalFees() should be declared external:

- SphynxNetwork.totalFees() (SphynxNetwork.sol#825-827)

deliver(uint256) should be declared external:

- SphynxNetwork.deliver(uint256) (SphynxNetwork.sol#829-836)  
reflectionFromToken(uint256,bool) should be declared external:
- SphynxNetwork.reflectionFromToken(uint256,bool) (SphynxNetwork.sol#838-847)  
excludeFromReward(address) should be declared external:
- SphynxNetwork.excludeFromReward(address) (SphynxNetwork.sol#855-863)  
excludeFromFee(address) should be declared external:
- SphynxNetwork.excludeFromFee(address) (SphynxNetwork.sol#888-890)  
includeInFee(address) should be declared external:
- SphynxNetwork.includeInFee(address) (SphynxNetwork.sol#892-894)  
setSwapAndLiquifyEnabled(bool) should be declared external:
- SphynxNetwork.setSwapAndLiquifyEnabled(bool) (SphynxNetwork.sol#910-913)  
isExcludedFromFee(address) should be declared external:
- SphynxNetwork.isExcludedFromFee(address) (SphynxNetwork.sol#996-998)

Reference:

<https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external>

INFO:Slither:SphynxNetwork.sol analyzed (10 contracts with 75 detectors), 117 result(s) found

INFO:Slither:Use <https://crytic.io/> to get access to additional detectors and Github integration



This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

**Email: [audit@EtherAuthority.io](mailto:audit@EtherAuthority.io)**