



# Security Assessment: CygnusNetwork Token

February 2, 2024







- Audit Status: **Fail**
- Audit Edition: **Standard**
































# Risk Analysis

## Classifications of Manual Risk Results

Classification	Description
 Critical	Danger or Potential Problems.
 High	Be Careful or Fail test.
 Low	Pass, Not-Detected or Safe Item.
 Informational	Function Detected

## Manual Code Review Risk Results

Contract Privilege	Description
 Buy Tax	20%
 Sale Tax	20%
 Cannot Sale	Pass
 Cannot Sale	Pass
 Max Tax	20%
 Modify Tax	Yes
 Fee Check	Pass
 Is Honeypot?	Not Detected
 Trading Cooldown	Not Detected
 Can Pause Trade?	Pass
 Pause Transfer?	Not Detected
 Max Tx?	Pass
 Is Anti Whale?	Not Detected
 Is Anti Bot?	Not Detected

Contract Privilege	Description
 Is Blacklist?	Not Detected
 Blacklist Check	Pass
 is Whitelist?	Detected
 Can Mint?	Pass
 Is Proxy?	Not Detected
 Can Take Ownership?	Not Detected
 Hidden Owner?	Not Detected
 Owner	0x
 Self Destruct?	Not Detected
 External Call?	Detected
 Other?	Not Detected
 Holders	1
 Auditor Confidence	Medium
 KYC Present	No
 KYC URL	<a href="https://assuredefi.com/projects/fort-block-games/">https://assuredefi.com/projects/fort-block-games/</a>

The following quick summary it's added to the project overview; however, there are more details about the audit and its results. Please read every detail.

# Project Overview

## Token Summary

Parameter	Result
Address	
Name	CygnusNetwork
Token Tracker	CygnusNetwork (CygN)
Decimals	18
Supply	500,000,000
Platform	ETHEREUM
compiler	0.8.23
Contract Name	CygnusNetworkToken
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	
Payment Tx	Corporate

## Main Contract Assessed Contract Name

Name	Contract	Live
CygnusNetwork		No

## TestNet Contract Assessed Contract Name

Name	Contract	Live
CygnusNetwork	0xac96EE5cAb0d4d70178f06E49F635256bF4c9A43	No

## Solidity Code Provided

SolidID	File Sha-1	FileName
CYGNUS	6e090a485077fdd92382ecbeabfd3e52	cygnusnetworktoken.sol
CYGNUS		
CYGNUS		
CYGNUS		
CYGNUS		
CYGNUS	undefined	

# Smart Contract Vulnerability Checks

**The Smart Contract Weakness Classification Registry (SWC Registry) is an implementation of the weakness classification scheme proposed in EIP-1470. It is loosely aligned to the terminologies and structure used in the Common Weakness Enumeration (CWE) while overlaying a wide range of weakness variants that are specific to smart contracts.**

ID	Severity	Name	File	location
SWC-100	Pass	Function Default Visibility	cygnusnetworktoken.s ol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	cygnusnetworktoken.s ol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	cygnusnetworktoken.s ol	L: 0 C: 0
SWC-103	Pass	A floating pragma is set.	cygnusnetworktoken.s ol	L: 0 C: 0
SWC-104	Pass	Unchecked Call Return Value.	cygnusnetworktoken.s ol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	cygnusnetworktoken.s ol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	cygnusnetworktoken.s ol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	cygnusnetworktoken.s ol	L: 0 C: 0
SWC-108	Low	State variable visibility is not set..	cygnusnetworktoken.s ol	L: 82 C: 12
SWC-109	Pass	Uninitialized Storage Pointer.	cygnusnetworktoken.s ol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	cygnusnetworktoken.s ol	L: 0 C: 0
SWC-111	Pass	Use of Deprecated Solidity Functions.	cygnusnetworktoken.s ol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	cygnusnetworktoken.s ol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	cygnusnetworktoken.s ol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-114	Pass	Transaction Order Dependence.	cygnusnetworktoken.sol	L: 0 C: 0
SWC-115	Pass	Authorization through tx.origin.	cygnusnetworktoken.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	cygnusnetworktoken.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	cygnusnetworktoken.sol	L: 0 C: 0
SWC-118	Medium	Incorrect Constructor Name.	cygnusnetworktoken.sol	L: 513 C: 14
SWC-119	Pass	Shadowing State Variables.	cygnusnetworktoken.sol	L: 0 C: 0
SWC-120	Low	Potential use of block.number as source of randomness.	cygnusnetworktoken.sol	L: 3391 C: 42
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	cygnusnetworktoken.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	cygnusnetworktoken.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	cygnusnetworktoken.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	cygnusnetworktoken.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	cygnusnetworktoken.sol	L: 0 C: 0
SWC-126	Pass	Insufficient Gas Griefing.	cygnusnetworktoken.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	cygnusnetworktoken.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	cygnusnetworktoken.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	cygnusnetworktoken.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-130	Pass	Right-To-Left-Override control character (U+202E).	cygnusnetworktoken.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	cygnusnetworktoken.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	cygnusnetworktoken.sol	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	cygnusnetworktoken.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	cygnusnetworktoken.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	cygnusnetworktoken.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	cygnusnetworktoken.sol	L: 0 C: 0

We scan the contract for additional security issues using MYTHX and industry-standard security scanning tools.



# Smart Contract Vulnerability Details

## SWC-108 - State Variable Default Visibility

### CWE-710: Improper Adherence to Coding Standards

#### Description:

Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.

#### Remediation:

Variables can be specified as being public, internal or private. Explicitly define visibility for all state variables.

#### References:

Ethereum Smart Contract Best Practices - Explicitly mark visibility in functions and state variables

# Smart Contract Vulnerability Details

## SWC-118 - Incorrect Constructor Name

### CWE-665: Improper Initialization

#### Description:

Constructors are special functions that are called only once during the contract creation. They often perform critical, privileged actions such as setting the owner of the contract. Before Solidity version 0.4.22, the only way of defining a constructor was to create a function with the same name as the contract class containing it. A function meant to become a constructor becomes a normal, callable function if its name doesn't exactly match the contract name. This behavior sometimes leads to security issues, in particular when smart contract code is re-used with a different name but the name of the constructor function is not changed accordingly..

#### Remediation:

Solidity version 0.4.22 introduces a new constructor keyword that make a constructor definitions clearer. It is therefore recommended to upgrade the contract to a recent version of the Solidity compiler and change to the new constructor declaration.

#### References:

SigmaPrime - Constructors with Care

# Smart Contract Vulnerability Details

## SWC-120 - Weak Sources of Randomness from Chain Attributes

### CWE-330: Use of Insufficiently Random Values

#### Description:

Solidity allows for ambiguous naming of state variables when inheritance is used. Contract A with a variable `x` could inherit contract B that also has a state variable `x` defined. This would result in two separate versions of `x`, one of them being accessed from contract A and the other one from contract B. In more complex contract systems this condition could go unnoticed and subsequently lead to security issues.

Shadowing state variables can also occur within a single contract when there are multiple definitions on the contract and function level.

#### Remediation:

Using commitment scheme, e.g. RANDAO. Using external sources of randomness via oracles, e.g. Oraclize. Note that this approach requires trusting in oracle, thus it may be reasonable to use multiple oracles. Using Bitcoin block hashes, as they are more expensive to mine.

#### References:

How can I securely generate a random number in my smart contract?)

When can BLOCKHASH be safely used for a random number? When would it be unsafe?

The Run smart contract.





## Privileged Functions (onlyOwner)

Please Note if the contract is Renounced none of this functions can be executed.

Function Name	Parameters	Visibility
renounceOwnership		External
transferOwnership	address newOwner	External
includeInRewards		External
setTaxEnabled		External
setMiniBeforeLiquify		External
setExcludedFromFee		External
setPair		External
setLiquidityHolder		External
setSellTax		External
setBuyTax		External
recoverLostTokens		External

## CygN-01 | Potential Sandwich Attacks.

Category	Severity	Location	Status
Security	 Low	cygnusnetworktoken.sol: L: 5250, C: 14	 Detected

### Description

A sandwich attack might happen when an attacker observes a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by frontrunning (before the transaction being attacked) a transaction to purchase one of the assets and make profits by back running (after the transaction being attacked) a transaction to sell the asset. The following functions are called without setting restrictions on slippage or minimum output amount, so transactions triggering these functions are vulnerable to sandwich attacks, especially when the input amount is large:

- swapExactTokensForETHSupportingFeeOnTransferTokens()
- addLiquidityETH()



### Remediation

We recommend setting reasonable minimum output amounts, instead of 0, based on token prices when calling the aforementioned functions.

### References:

What Are Sandwich Attacks in DeFi — and How Can You Avoid Them?.

## CygN-04 | Centralized Risk In addLiquidity.

Category	Severity	Location	Status
Coding Style	 High	cygnusnetworktoken.sol: L: 5268, C: 14	 Detected

### Description

```
uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this), tokenAmount, 0, 0, owner(), block.timestamp);
```

The addLiquidity function calls the uniswapV2Router.addLiquidityETH function with the to address specified as owner() for acquiring the generated LP tokens from the CygN-WBNB pool.

As a result, over time the \_owner address will accumulate a significant portion of LP tokens. If the \_owner is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

### Remediation

We advise the to address of the uniswapV2Router.addLiquidityETH function call to be replaced by the contract itself, i.e. address(this) , and to restrict the management of the LP tokens within the scope of the contract's business logic. This will also protect the LP tokens from being stolen if the \_owner account is compromised. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.



1. Indicatively, here are some feasible solutions that would also mitigate the potential risk:
2. Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
3. Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;

Introduction of a DAO / governance / voting module to increase transparency and user involvement

### Project Action

liquidity is set to liquidityHolder

## CygN-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	 Low	cygnusnetworktoken.sol: L: 5171, C: 14	 Detected

### Description



Detected missing events for critical arithmetic parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes. The linked code does not create an event for the transfer.

### Remediation

Emit an event for critical parameter changes. It is recommended emitting events for the sensitive functions that are controlled by centralization roles.



## CygN-10 | Initial Token Distribution.

Category	Severity	Location	Status
Centralization / Privilege	 High	cygnusnetworktoken.sol: L: 811 C: 14	 Detected

### Description

All of the CygnusNetwork tokens are sent to the contract deployer when deploying the contract. This could be a centralization risk as the deployer can distribute tokens without obtaining the consensus of the community.

### Remediation






We recommend the team to be transparent regarding the initial token distribution process, and the team shall make enough efforts to restrict the access of the private key.

### Project Action






```
emit Transfer(address(0), _msgSender(), _tTotal);
```

# Technical Findings Summary

## Classification of Risk

Severity	Description
 Critical	Risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
 High	Risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
 Medium	Risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform
 Low	Risks can be any of the above but on a smaller scale. They generally do not compromise the overall integrity of the Project, but they may be less efficient than other solutions.
 Informational	Errors are often recommended to improve the code's style or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

## Findings

Severity	Found	Pending	Resolved
 Critical	0	0	0
 High	2	2	0
 Medium	1	0	0
 Low	1	2	0
 Informational	0	0	0
Total	4	4	0

# Social Media Checks

Social Media	URL	Result
Twitter		N/A
Other	<a href="https://medium.com/@cygnusnetwork">https://medium.com/@cygnusnetwork</a>	Pass
Website	<a href="https://cygnus.network">https://cygnus.network</a>	Pass
Telegram	<a href="https://t.me/CygnusNetworkOfficial">https://t.me/CygnusNetworkOfficial</a>	Pass

We recommend to have 3 or more social media sources including a completed working websites.

**Social Media Information Notes:**

**Auditor Notes:** undefined

**Project Owner Notes:**



# Assessment Results

## Score Results

Review	Score
Overall Score	82/100
Auditor Score	84/100
Review by Section	Score
Manual Scan Score	22
SWC Scan Score	31
Advance Check Score	29

The Following Score System Has been Added to this page to help understand the value of the audit, the maximum score is 100, however to attain that value the project must pass and provide all the data needed for the assessment. Our Passing Score has been changed to 84 Points for a higher standard, if a project does not attain 85% is an automatic failure. Read our notes and final assessment below.

## Audit Fail



## **Assessment Results**

### **Important Notes:**

- The following contract is a reward-based contract.␣
- The developer documented the code very well.␣
- Liquidity goes to an external wallet.␣
- We need token Distribution Information and ownership plans.␣
- Always DYOR.

**Auditor Score =84**  
**Audit Fail**



# Appendix

## Finding Categories

### Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

### Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

### Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

### Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

### Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

### Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

### Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different requirements on the input variables than a setter function.

### Coding Best Practices

ERC 20 Coding Standards are a set of rules that each developer should follow to ensure the code meets a set of criteria and is readable by all the developers.

# Disclaimer

Assure Defi has conducted an independent security assessment to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the reviewed code for the scope of this assessment. This report does not constitute agreement, acceptance, or advocacy for the Project, and users relying on this report should not consider this as having any merit for financial advice in any shape, form, or nature. The contracts audited do not account for any economic developments that the Project in question may pursue, and the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude, and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are entirely free of exploits, bugs, vulnerabilities or deprecation of technologies.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence, regardless of the findings presented. Information is provided 'as is, and Assure Defi is under no covenant to audited completeness, accuracy, or solidity of the contracts. In no event will Assure Defi or its partners, employees, agents, or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions or actions with regards to the information provided in this audit report.

The assessment services provided by Assure Defi are subject to dependencies and are under continuing development. You agree that your access or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies with high levels of technical risk and uncertainty. The assessment reports could include false positives, negatives, and unpredictable results. The services may access, and depend upon, multiple layers of third parties.

