



Security Assessment: **CYBONK Token**

March 5, 2024

- Audit Status: **Pass**
- 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	0%
● Sale Tax	0%
● Cannot Sale	Pass
● Cannot Sale	Pass
● Max Tax	0%
● Modify Tax	No
● 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?	Pass
● Can Mint?	Pass
● Is Proxy?	Not Detected
● Can Take Ownership?	Not Detected
● Hidden Owner?	Not Detected
● Owner	no
● Self Destruct?	Not Detected
● External Call?	Not Detected
● Other?	Not Detected
● Holders	1
● Auditor Confidence	Low Risk
● KYC Present	No
● KYC URL	

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	0x9194337C06405623c0F374e63Fa1CC94e2788c58
Name	CYBONK
Token Tracker	CYBONK (CYBONK)
Decimals	18
Supply	100,000,000,000
Platform	ETHEREUM
compiler	v0.8.23+commit.f704f362
Contract Name	Cybonk
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://etherscan.io/ token/0x9194337c06405623c0f374e63fa1cc94e2788c58#code
Payment Tx	Corporate

Main Contract Assessed Contract Name

Name	Contract	Live
CYBONK	0x9194337C06405623c0F374e63Fa1CC94e2788c58	Yes

TestNet Contract was Not Assessed

Solidity Code Provided

Solid ID	File Sha-1	FileName
CYBONK	2ffe3004f7493ac379f9760146cca3509ee66d1a	contract.sol
CYBONK		
CYBONK		
CYBONK		
CYBONK	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	contract.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	contract.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	contract.sol	L: 0 C: 0
SWC-103	Pass	A floating pragma is set.	contract.sol	L: 0 C: 0
SWC-104	Pass	Unchecked Call Return Value.	contract.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	contract.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	contract.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	contract.sol	L: 0 C: 0
SWC-108	Pass	State variable visibility is not set..	contract.sol	L: 0 C: 0
SWC-109	Pass	Uninitialized Storage Pointer.	contract.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	contract.sol	L: 0 C: 0
SWC-111	Pass	Use of Deprecated Solidity Functions.	contract.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	contract.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	contract.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	contract.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-115	Pass	Authorization through tx.origin.	contract.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	contract.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	contract.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	contract.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	contract.sol	L: 0 C: 0
SWC-120	Pass	Potential use of block.number as source of randomness.	contract.sol	L: 0 C: 0
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	contract.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	contract.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	contract.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	contract.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	contract.sol	L: 0 C: 0
SWC-126	Pass	Insufficient Gas Griefing.	contract.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	contract.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	contract.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	contract.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U+202E).	contract.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	contract.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	contract.sol	L: 0 C: 0

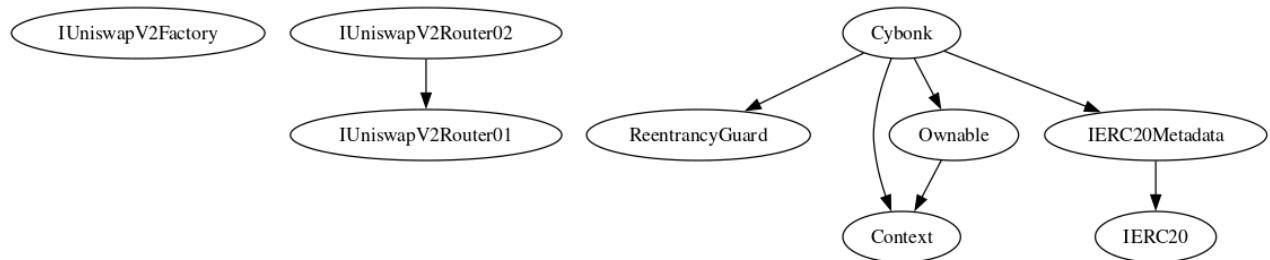
ID	Severity	Name	File	location
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	contract.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	contract.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	contract.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	contract.sol	L: 0 C: 0

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

Inheritance

The contract for CYBONK has the following inheritance structure.

The Project has a Total Supply of 100,000,000,000



Privileged Functions (onlyOwner)

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

Function Name	Parameters	Visibility
renounceOwnership		Public
transferOwnership	address newOwner	Public
setBuyFee	uint256 newBuyFee	Public
setSellFee	uint256 newSellFee	Public
updateWallet	address account, bool status	Public

CYBONK-01 | Potential Sandwich Attacks.

Category	Severity	Location	Status
Security	 Medium	contract.sol: L: 594, 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?.](#)

CYBONK-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	 Low	contract.sol: L: 631-659 C: 14	 Detected

Description

The given input is missing the check for the non-zero address.

The given input is missing the check for the missing required function.

Remediation

We advise the client to add the check for the passed-in values to prevent unexpected errors as below:

```
...
require(receiver != address(0), "Receiver is the zero address");
...
...
require(value X limitation, "Your not able to do this function");
...
```

We also recommend customer to review the following function that is missing a required validation. missing required function.

CYBONK-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	 Low	contract.sol: L: 631-659 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.

CYBONK-06 | Conformance with Solidity Naming Conventions.

Category	Severity	Location	Status
Coding Style	● Low	contract.sol: L: 505 C: 14	■ Detected

Description

Solidity defines a naming convention that should be followed. Rule exceptions: Allow constant variable name/symbol/decimals to be lowercase. Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

list

Remediation

Follow the Solidity naming convention.

<https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-convention>

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	0	0	0
● Medium	1	1	0
● Low	3	3	0
● Informational	0	0	0
Total	4	4	0

Social Media Checks

Social Media	URL	Result
Twitter	https://twitter.com/cybonktoken	Pass
Other		Fail
Website	https://cybonk.io	Pass
Telegram	https://t.me/CYBONKTOKEN	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	87/100
Auditor Score	90/100
Review by Section	Score
Manual Scan Score	22
SWC Scan Score	37
Advance Check Score	28

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 Passed



Assessment Results

Important Notes:

- Reentrancy: Properly addressed with ReentrancyGuard.
- Ownership: Centralized control; owner can change fees and exempt addresses. Ownership can be renounced.
- Fee Logic: No validation on fee percentage inputs; could be set to extreme values.
- Function Visibility: Correctly scoped, but some public functions could be external.
- Gas Efficiency: Could be improved by optimizing state variable visibility and event emissions.
- Renounce Ownership: The renounceOwnership function allows the owner to permanently renounce control. This was executed in TX Hash 0x42690da2d136871eee5b9d1da2691d7d9131fc4d025ac127aa54db18a587ef8c.
- Naming Conventions: The List variable should follow the naming convention, e.g., isExcludedFromFees.
- Constants: NOT_ENTERED and ENTERED can be declared as constant.
- Dead Code: No dead code detected in the provided snippet.
- Sandwich Attacks: Potential risk due to public fee settings and lack of anti-front-running mechanisms.

Auditor Score =90

Audit Passed



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 require statements 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 depreciation 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.

