

Security Assessment: **Anydex Token**





April 9, 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	25%
 Sale Tax	25%
 Cannot Buy	Pass
 Cannot Sale	Pass
 Max Tax	30%
 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	
 Self Destruct?	Not Detected
 External Call?	Not Detected
 Other?	Not Detected
 Holders	0
 Auditor Confidence	Medium-High 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	
Name	Anydex
Token Tracker	Anydex (ANYDEX)
Decimals	18
Supply	10,000,000
Platform	ETHEREUM
compiler	v0.8.20+commit.a1b79de6
Contract Name	Anydex
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://basescan.org/address/0x338b050D138529CD6d76AE2702fFcB02490dd828#code
Payment Tx	Corporate

Main Contract Assessed Contract Name

Name	Contract	Live
Anydex		Yes

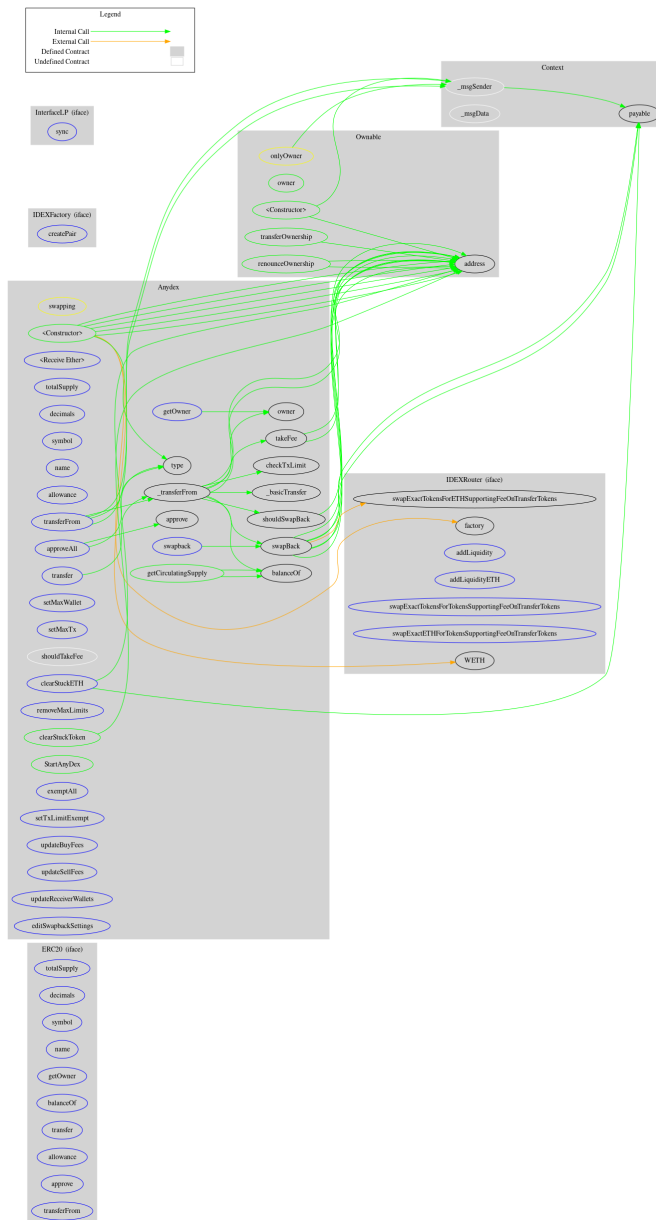
TestNet Contract was Not Assessed

Solidity Code Provided

SolidID	File Sha-1	FileName
Anydex	390b2d0c88c78b5e2a8ac6324782920a90afea65	anydex.sol
Anydex		
Anydex		
Anydex		
Anydex		
Anydex	undefined	

Call Graph

The contract for Anydex has the following call graph structure.



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	anydex.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	anydex.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	anydex.sol	L: 0 C: 0
SWC-103	Low	A floating pragma is set.	anydex.sol	L: 4 C: 0
SWC-104	Pass	Unchecked Call Return Value.	anydex.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	anydex.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	anydex.sol	L: 0 C: 0
SWC-107	Low	Read of persistent state following external call.	anydex.sol	L: 124-171 C: 12
SWC-108	Pass	State variable visibility is not set..	anydex.sol	L: 0 C: 0
SWC-109	Pass	Uninitialized Storage Pointer.	anydex.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	anydex.sol	L: 0 C: 0
SWC-111	Pass	Use of Deprecated Solidity Functions.	anydex.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	anydex.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	anydex.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	anydex.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-115	Pass	Authorization through tx.origin.	anydex.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	anydex.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	anydex.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	anydex.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	anydex.sol	L: 0 C: 0
SWC-120	Low	Potential use of block.number as source of randomness.	anydex.sol	L: 267-357 C: 23
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	anydex.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	anydex.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	anydex.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	anydex.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	anydex.sol	L: 0 C: 0
SWC-126	Pass	Insufficient Gas Griefing.	anydex.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	anydex.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	anydex.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	anydex.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U+202E).	anydex.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	anydex.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	anydex.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	anydex.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	anydex.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	anydex.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	anydex.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-103 - Floating Pragma.

CWE-664: Improper Control of a Resource Through its Lifetime.

References:

Description:

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Remediation:

Lock the pragma version and also consider known bugs (<https://github.com/ethereum/solidity/releases>) for the compiler version that is chosen.

Pragma statements can be allowed to float when a contract is intended for consumption by other developers, as in the case with contracts in a library or EthPM package. Otherwise, the developer would need to manually update the pragma in order to compile locally.

References:

Ethereum Smart Contract Best Practices - Lock pragmas to specific compiler version.

Smart Contract Vulnerability Details

SWC-107 - Reentrancy.

CWE-841: Improper Enforcement of Behavioral Workflow.

Description:

One of the major dangers of calling external contracts is that they can take over the control flow. In the reentrancy attack (a.k.a. recursive call attack), a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways.

Remediation:

The best practices to avoid Reentrancy weaknesses are: Make sure all internal state changes are performed before the call is executed. This is known as the Checks-Effects-Interactions pattern. Use a reentrancy lock.

References:

Ethereum Smart Contract Best Practices - Reentrancy

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?)

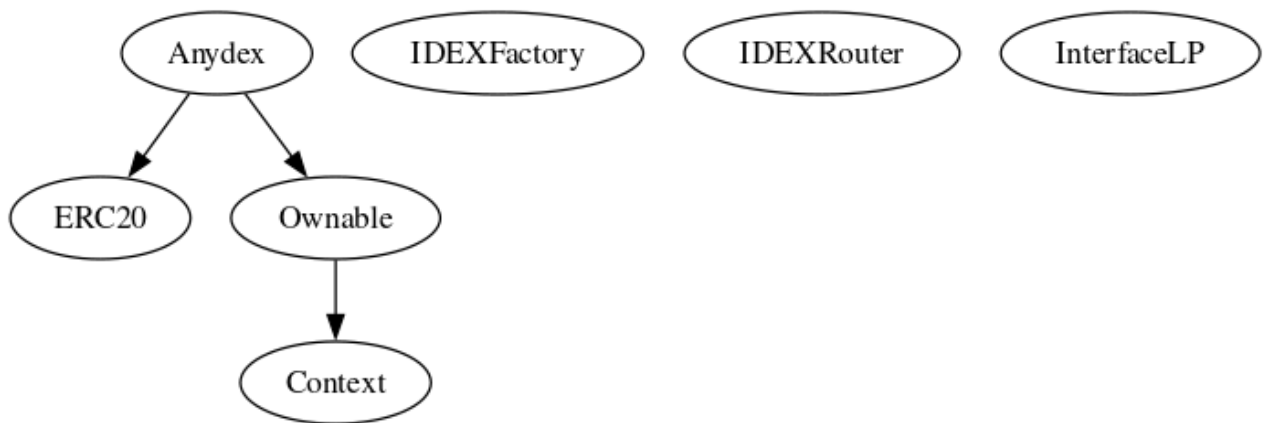
When can BLOCKHASH be safely used for a random number? When would it be unsafe?

The Run smart contract.

Inheritance

The contract for Anydex has the following inheritance structure.

The Project has a Total Supply of 10,000,000





Privileged Functions (onlyOwner)

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

Function Name	Parameters	Visibility
transferOwnership	address newOwner	Public
renounceOwnership		Public
setMaxWallet	uint256 maxWalletPercent	Public
setMaxTx	uint256 maxTXPercent	Public
swapback		Public
removeMaxLimits		Public
clearStuckToken	address tokenAddress, uint256 tokens	Public
StartAnyDex		Public
exemptAll	address holder, bool exempt	Public
setTxLimitExempt	address holder, bool exempt	External
updateBuyFees	uint256 _liquidityFee, uint256 _teamFee, uint256 _marketingFee	External

Function Name	Parameters	Visibility
updateSellFees	uint256 _liquidityFee, uint256 _teamFee, uint256 _marketingFee	External
updateReceiverWallets	address _autoLiquidit yReceiver, address _ marketingFeeReceive r, address _teamFeeReceiver	External
editSwapbackSettings	bool _enabled, uint256 _amount	Public
clearStuckETH	uint256 amountPercentage	External

ANYDEX-01 | Potential Sandwich Attacks.

Category	Severity	Location	Status
Security	 Medium	anydex.sol: L: 360, 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?.

ANYDEX-02 | Function Visibility Optimization.

Category	Severity	Location	Status
Gas Optimization	 Informational	anydex.sol: L: 54-59 C: 14, L: 345 C: 14, L: 354 C: 14, L: 354 C: 14	 Detected

Description

The following functions are declared as public and are not invoked in any of the contracts contained within the projects scope:

Function Name	Parameters	Visibility
transferOwnership	address newOwner	Public
renounceOwnership		Public
clearStuckToken	address tokenAddress, uint256 tokens	Public
StartAnyDex		Public

The functions that are never called internally within the contract should have external visibility



Remediation

We advise that the function's visibility specifiers are set to external, and the array-based arguments change their data location from memory to calldata, optimizing the gas cost of the function.

References:

external vs public best practices.

ANYDEX-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	 Low	anydex.sol: L: 407-412 C: 14, L: 433-439 C: 14, L: 532 C: 14, L: 532-593 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.

ANYDEX-04 | Centralized Risk In addLiquidity.

Category	Severity	Location	Status
Coding Style	 High	anydex.sol: L:400, 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 ANYDEX-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 autoLiquidityReceiver

ANYDEX-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	 Low	anydex.sol: L: 407-439 C: 14, L: 339 C: 14, L: 235-241 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.

ANYDEX-06 | Conformance with Solidity Naming Conventions.

Category	Severity	Location	Status
Coding Style	 Low	anydex.sol: L: 354 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.



StartAnyDex

Remediation

Follow the Solidity naming convention.

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

ANYDEX-07 | State Variables could be Declared Constant.

Category	Severity	Location	Status
Coding Style	 Low	anydex.sol: L: 125-126 C: 14	 Detected

Description

Constant state variables should be declared constant to save gas.



```
DEAD  
ZERO
```

Remediation

Add the constant attribute to state variables that never changes.

<https://docs.soliditylang.org/en/latest/contracts.html#constant-state-variables>

ANYDEX-08 | Dead Code Elimination.

Category	Severity	Location	Status
Coding Style	 Low	anydex.sol: L: 28 C:14	 Detected

Description

Functions that are not used in the contract, and make the code s size bigger.



```
_msgData()
```

Remediation

Remove unused functions. dead-code elimination (also known as DCE, dead-code removal, dead-code stripping, or dead-code strip) is a compiler optimization to remove code which does not affect the program results. Removing such code has several benefits: it shrinks program size, an important consideration in some contexts, and it allows the running program to avoid executing irrelevant operations, which reduces its running time. It can also enable further optimizations by simplifying program structure.

<https://docs.soliditylang.org/en/latest/cheatsheet.html>

ANYDEX-10 | Initial Token Distribution.

Category	Severity	Location	Status
Centralization / Privilege	 High	anydex.sol: L: 195 C: 14	 Detected

Description

All of the Anydex 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), msg.sender, _totalSupply);
```


ANYDEX-11 | Redundant swapback Wrapper Function.

Category	Severity	Location	Status
Optimization	 Low	anydex.sol: L: 334 C: 14	 Detected

Description



The contract contains a public swapback function that simply calls an internal swapBack function. This wrapper is redundant since the internal function already has the necessary swapping modifier and can only be called by the contract itself. Remediation: Consider removing the external swapback function if it does not serve a distinct purpose from the internal swapBack function. If the intention is to provide an external interface to trigger swaps, ensure it is properly secured with onlyOwner.

Remediation

The presence of the swapback function does not pose a direct security risk since it is properly protected by the onlyOwner modifier. However, if it is redundant, removing it can simplify the contract's interface.

Project Action

ANYDEX-18 | Stop Transactions by using Enable Trade.

Category	Severity	Location	Status
Logical Issue	 Critical	anydex.sol: L: 354 C: 14	 Detected

Description



Enable Trade is present on the following contract and when combined with Exclude from fees it can be considered a whitelist process, this will allow anyone to trade before others and can represent an issue for the holders.

Remediation

We recommend the project owner to carefully review this function and avoid problems when performing both actions.

Project Action

ANYDEX-19 | Arbitrary setMaxWallet and setMaxTx.

Category	Severity	Location	Status
Access Control	 Medium	anydex.sol: L: 500 C: 14	 Detected

Description



The owner can arbitrarily change the maximum wallet and transaction limits to very high values, effectively removing these protections.

Remediation

Implement a maximum cap for these values or require community consensus for changes.

Project Action

ANYDEX-20 | Fee Receiver Wallet Update Risk.

Category	Severity	Location	Status
Logical	 Critical	anydex.sol: L: 433 C: 14	 Detected

Description

The updateReceiverWallets function allows the owner to change the destination wallets for fees, which could be misused.






Remediation

Implement a transparent process for updating fee receiver wallets, possibly involving multi-sig or community approval, also their missing zero address check and can make the contract into a honeypot.






Project Action

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	2	2	0
 High	2	2	0
 Medium	2	2	0
 Low	6	6	0
 Informational	1	1	0
Total	13	13	0

Social Media Checks

Social Media	URL	Result
Twitter	https://x.com/anydexofficial	Pass
Other	no	N/A
Website	https://anydex.org	Pass
Telegram	https://t.me/anydexofficial	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	48/100
Auditor Score	65/100
Review by Section	Score
Manual Scan Score	24
SWC Scan Score	31
Advance Check Score	-7

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:

- Privileged Functions Without Timelock - Medium risk, abrupt changes without notice.␣
- Renounce Ownership Risk - High risk, irreversible loss of control.␣
- Lack of Slippage Protection in swapBack - Medium risk, unfavorable swap rates.␣
- External Contract Dependencies - Medium risk, dependency vulnerabilities.␣
- High Transaction Fees - Low risk, user deterrence.␣
- Lack of approve Front-Running Protection - Medium risk, front-running attacks.␣
- clearStuckETH and clearStuckToken Exploitability - High risk, unauthorized withdrawals.␣
- Redundant swapback Wrapper Function - Low risk, code redundancy.␣
- Toggling TradingOpen Flag - Medium risk, abrupt trading disruption.␣
- exemptAll Privileged Trading Potential - Medium risk, unfair advantages.␣
- Arbitrary setMaxWallet and setMaxTx Changes - Medium risk, whale manipulation.␣

- Removal of Max Limits Function - High risk, whale manipulation.↓
- Fee Receiver Wallet Update Risk - Medium risk, fund misdirection.↓
- Swapback Settings Modification - Medium risk, changes without consensus.↓
- Hardcoded Router and Pair Addresses - Medium risk, lack of upgradability.↓
- Trading Toggle Without Constraints - Medium risk, market manipulation.↓
- Non-Conformance with Solidity Naming Conventions - Low risk, code quality.↓
- State Variables Not Declared as Constant - Low risk, gas optimization.↓
- Dead Code Presence - Low risk, code quality.↓
- Missing Input Validations in onlyOwner Functions - Medium risk, unintended behavior.↓
- Missing Event Emissions in onlyOwner Functions - Low risk, transparency.↓
- Function Naming Convention Non-Adherence - Low risk, code quality.↓
- Incorrect Access Control on swapback Function - High risk, unauthorized access.↓
- Overall Risk - Medium-High↓
- Score - 65/100↓

- Final Conclusion - The contract has several areas that require attention, including access control, transparency, and adherence to best practices. While some functions have proper security measures, the overall contract could benefit from additional input validations, event emissions, and naming convention adherence. The high severity issues like potential loss of control and unauthorized withdrawals should be addressed immediately. The score remains below the passing threshold, indicating the need for remediation before the contract is considered secure.

Auditor Score =65
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.

