ASSURE DEFI®
THE VERIFICATION GOLD STANDARD

Security Assessment:

Anydex Token

April 12, 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?	Not 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

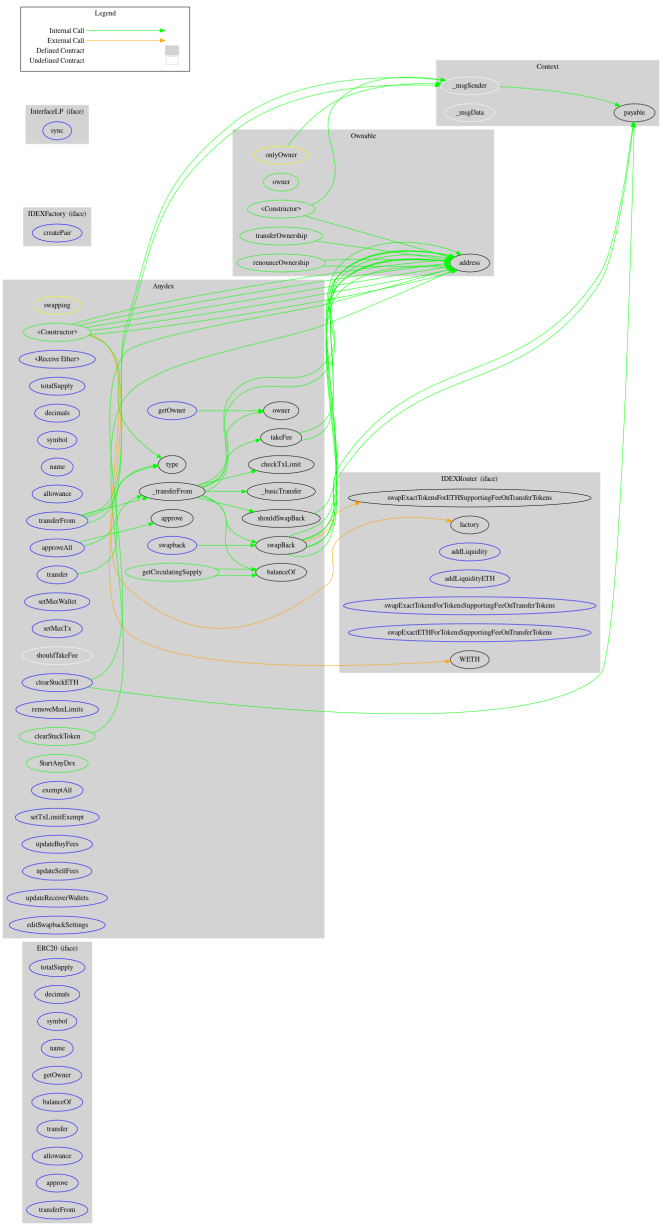
TestNet Contract was Not Assessed

Solidity Code Provided

SolID	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 randonmness.	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?)

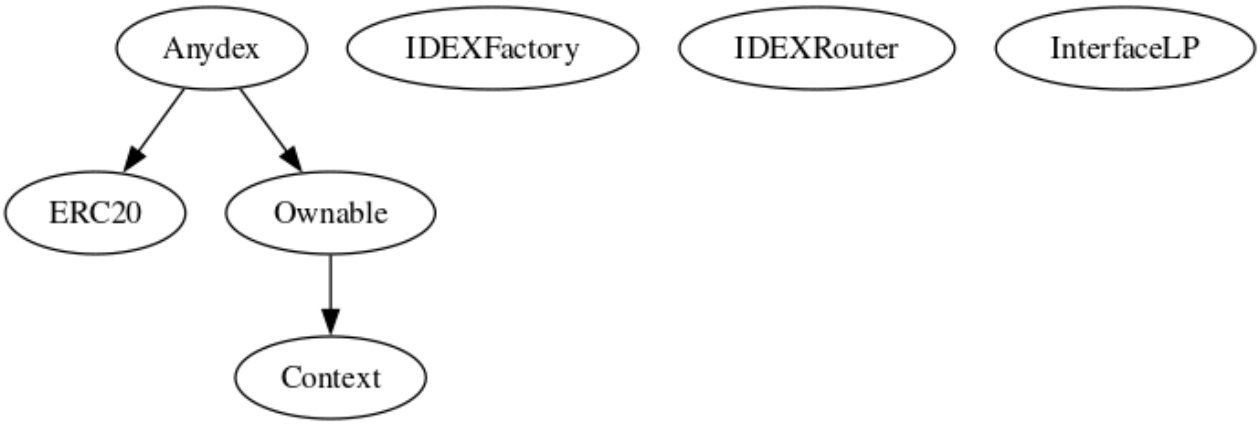
When can BLOCKHASH be safely used for a random number? When would it be unsafe?

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		External
setMaxWallet	uint256 maxWalletPercent	External
setMaxTx	uint256 maxTxPercent	External
removeMaxLimits		External
clearStuckToken	address tokenAddress, uint256 tokens	External
startAnyDex		External
exemptAll	address holder, bool exempt	External
setTxLimitExempt	address holder, bool exempt	External
updateBuyFees	uint256 _teamFee, uint256 _marketingFee	External
updateSellFees	uint256 _teamFee, uint256 _marketingFee	External

Function Name	Parameters	Visibility
updateReceiverWallets	address _marketingFeeReceiver, address _teamFeeReceiver	External
editSwapbackSettings	bool _enabled, uint256 _amount	External
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.

Referrences:

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

ANYDEX-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	Low	anydex.sol: L: 419 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-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	Low	anydex.sol: L: 346 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-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-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 presend 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 and 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-21 | clearStuckETH and clearStuckToken Exploitability.

Category	Severity	Location	Status
Access Control / Funds Extraction	Critical	anydex.sol: L: 337 C: 14	Detected

Description

The clearStuckETH and clearStuckToken functions are intended to allow the contract owner to remove ETH or tokens that are accidentally sent to the contract. However, these functions could be exploited if the exempted addresses are compromised, leading to unauthorized withdrawals of funds.

Remediation

Implement additional safeguards to prevent unauthorized access to these functions. This could include multi-signature verification, timelocks, or restricting access to only a subset of trusted addresses with additional checks. It is also recommended to have a clear and transparent policy regarding the handling of stuck funds to ensure community trust..

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	1	1	1
High	2	2	1
Medium	1	1	1
Low	2	2	4
Informational	0	0	1
Total	6	6	8

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	79/100
Auditor Score	80/100
Review by Section	**Score**
Manual Scan Score	24
SWC Scan Score	31
Advance Check Score	24

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 most 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: Unresolved. No timelock implemented.ı
- Renounce Ownership Risk: Resolved. Function present to renounce ownership.ı
- Lack of Slippage Protection in swapBack: Unresolved. No slippage protection.ı
- External Contract Dependencies: Unresolved. Dependencies on external contracts.ı
- High Transaction Fees: Unresolved. Fees can be high, deterring users.ı
- Lack of approve Front-Running Protection: Unresolved. No front-running protection.ı
- clearStuckETH and clearStuckToken Exploitability: additional safeguards are recommended.ı
- Toggling TradingOpen Flag: Unresolved. Can be toggled without constraints.ı
- exemptAll Privileged Trading Potential: Unresolved. Can give unfair advantages.ı
- Arbitrary setMaxWallet and setMaxTx Changes: Unresolved. Can be changed by owner without constraints.ı
- Removal of Max Limits Function: Unresolved. Owner can remove limits without constraints.ı

- Fee Receiver Wallet Update Risk: Resolved. Validation added to prevent setting to zero address.ı
- Overall Risk: Medium. Some high-severity issues have been addressed, but others remain that could impact the contract's integrity and security.ı
- Score: 75/100. The score reflects the resolution of some issues, but unresolved concerns continue to impact the overall security assessment.ı
- Conclusion: The contract has seen improvements, with certain high-severity issues being addressed. However, further attention is still required to resolve outstanding issues related to access control, transparency, and adherence to best practices. Additional input validations, event emissions, and adherence to best practices are recommended. The contract's security posture has improved, but further remediation is advised to ensure the contract is secure and functions as intended.ı

Auditor Score =80
Audit Fail

Appendix

Finding Categories

Centralization / Privilege

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

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimalEVM 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 howblock.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functionsbeing 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 mayresult in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to makethe 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 setterfunction.

Coding Best Practices

ERC 20 Conding Standards are a set of rules that each developer should follow to ensure the code meet a set of creterias 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 advocation 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.