auditace.tech

AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Diamond Dex

DATED : 14 FEB 23'

AUDIT SUMMARY

Project name – Diamond Dex

Date: 14 February, 2023

Scope of Audit- Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

Audit Status: **Passed**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	0	0	0	0
Acknowledged	0	0	0	0	0
Resolved	0	0	0	0	0

USED TOOLS

Tools:

1- Manual Review:
a line by line code review has been performed by audit ace team.

2- BSC Test Network:
all tests were done on BSC Test network, each test has its transaction has attached to it.

3- Slither : Static Analysis

Testnet Link: all tests were done using this contract, tests are done on BSC Testnet

https://testnet.bscscan.com/token/0xE8D91684D9EFC37D7Ffe53D5e0e249B03E1b1511

Token Information

Token Name : Diamond Dex

Token Symbol: DDX

Decimals: 12

Token Supply: 8,000,000,000,000,000

Token Address:
https://bscscan.com/token/0x2af0ee3cc75ec4d434a49826aab94fdd21a11a21

Checksum:
a2ea4c87e83eab70edc4f39c2e7077389c3dd010c20cadfb9c58d7278cc3deec

Owner:
https://bscscan.com/address/0x07555c580214c2a857e8f8e5b9ece55a3fa4f3a3

TOKEN OVERVIEW

Fees:

Buy Fees: can be up to 25%

Sell Fees: can be up to 25%

Transfer Fees: can be up to 25%

Fees Privilige: Owner

Ownership : Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Priviliges: Changing Fees

AUDIT METHODOLOGY

The auditing process will follow a routine as special considerations by Auditace:

- Review of the specifications, sources, and instructions provided to Auditace to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.

- Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.

- Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Auditace describe.

- Test coverage analysis determines whether the test cases are covering the code and how much code isexercised when we run the test cases.

- Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.

- Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.

auditace.tech

VULNERABILITY CHECKLIST

- Return values of low-level calls
- Private modifier
- Multiple Sends
- Using Suicide
- Gas Limitand Loops
- Address hardcoded
- Exception Disorder
- Using inline assembly
- Divide before multiply
- Missing Zero Address Validation
- Compiler version not fixed
- **Gasless Send**
- Using block.timestamp
- Re-entrancy
- Tautology or contradiction
- Timestamp Dependence
- Revert/require functions
- Use of tx.origin
- Integer overflow/underflow
- Dangerous strict equalities
- Using SHA3
- Using throw

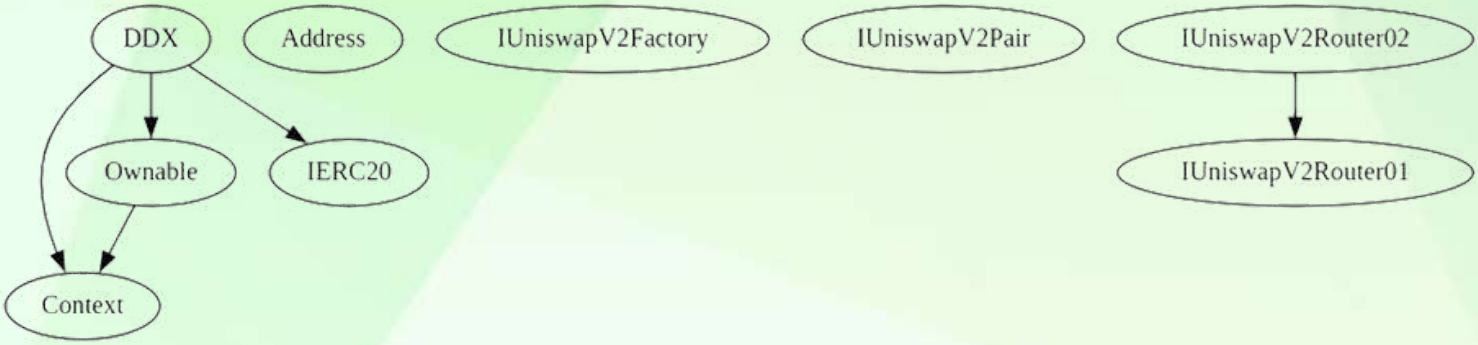
CLASSIFICATION OF RISK

Severity	Description
Critical	These vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
High-Risk	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.
Medium-Risk	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.
Low-Risk	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.
Gas Optimization /Suggestion	A vulnerability that has an informational character but is not affecting any of the code.

Findings

Severity	Found
Critical	0
High-Risk	0
Medium-Risk	0
Low-Risk	0
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- Owner is not able to set buy/sell/transfer taxes over 25%
- Owner is not able to blacklist an arbitrary wallet
- Owner is not able to set max buy/sell/transfer amounts
- Owner is not able to disable trades
- Owner is not able to mint new tokens

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
IERC20	Interface			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
Address	Library			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	_functionCallWithValue	Private 🔐	🛑	
IUniswapV2Factory	Interface			
└	feeTo	External ❗		NO❗
└	feeToSetter	External ❗		NO❗
└	getPair	External ❗		NO❗
└	allPairs	External ❗		NO❗
└	allPairsLength	External ❗		NO❗
└	createPair	External ❗	🛑	NO❗
└	setFeeTo	External ❗	🛑	NO❗
└	setFeeToSetter	External ❗	🛑	NO❗
IUniswapV2Pair	Interface			

CONTRACT ASSESMENT

└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transfer	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
└	DOMAIN_SEPARATOR	External ❗		NO❗
└	PERMIT_TYPEHASH	External ❗		NO❗
└	nonces	External ❗		NO❗
└	permit	External ❗	🛑	NO❗
└	MINIMUM_LIQUIDITY	External ❗		NO❗
└	factory	External ❗		NO❗
└	token0	External ❗		NO❗
└	token1	External ❗		NO❗
└	getReserves	External ❗		NO❗
└	price0CumulativeLast	External ❗		NO❗
└	price1CumulativeLast	External ❗		NO❗
└	kLast	External ❗		NO❗
└	burn	External ❗	🛑	NO❗
└	swap	External ❗	🛑	NO❗
└	skim	External ❗	🛑	NO❗
└	sync	External ❗	🛑	NO❗
└	initialize	External ❗	🛑	NO❗
IUniswapV2Router01	Interface			
└	factory	External ❗		NO❗
└	WETH	External ❗		NO❗
└	addLiquidity	External ❗	🛑	NO❗
└	addLiquidityETH	External ❗	💵	NO❗
└	removeLiquidity	External ❗	🛑	NO❗
└	removeLiquidityETH	External ❗	🛑	NO❗
└	removeLiquidityWithPermit	External ❗	🛑	NO❗
└	removeLiquidityETHWithPermit	External ❗	🛑	NO❗
└	swapExactTokensForTokens	External ❗	🛑	NO❗
└	swapTokensForExactTokens	External ❗	🛑	NO❗
└	swapExactETHForTokens	External ❗	💵	NO❗
└	swapTokensForExactETH	External ❗	🛑	NO❗
└	swapExactTokensForETH	External ❗	🛑	NO❗

CONTRACT ASSESMENT

└	swapETHForExactTokens	External ❗	💵	NO❗
└	quote	External ❗		NO❗
└	getAmountOut	External ❗		NO❗
└	getAmountIn	External ❗		NO❗
└	getAmountsOut	External ❗		NO❗
└	getAmountsIn	External ❗		NO❗
IUniswapV2Router02	Interface	IUniswapV2Router01		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
DDX	Implementation	Context, IERC20, Ownable		
└	<Constructor>	Public ❗	🛑	NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	decimals	Public ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	transfer	Public ❗	🛑	NO❗
└	allowance	Public ❗		NO❗
└	approve	Public ❗	🛑	NO❗
└	transferFrom	Public ❗	🛑	NO❗
└	increaseAllowance	Public ❗	🛑	NO❗
└	decreaseAllowance	Public ❗	🛑	NO❗
└	isExcludedFromReward	Public ❗		NO❗
└	totalReflectionDistributed	Public ❗		NO❗
└	deliver	Public ❗	🛑	NO❗
└	reflectionFromToken	Public ❗		NO❗
└	tokenFromReflection	Public ❗		NO❗
└	excludeFromReward	Public ❗	🛑	onlyOwner
└	includeInReward	External ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💵	NO❗
└	claimStuckTokens	External ❗	🛑	onlyOwner
└	_reflectFee	Private 🔐	🛑	
└	_getValues	Private 🔐		
└	_getTValues	Private 🔐		
└	_getRValues	Private 🔐		
└	_getRate	Private 🔐		

CONTRACT ASSESMENT

└	_getCurrentSupply	Private 🔐		
└	_takeLiquidity	Private 🔐	🛑	
└	_takeMarketing	Private 🔐	🛑	
└	calculateTaxFee	Private 🔐		
└	calculateLiquidityFee	Private 🔐		
└	calculateMarketingFee	Private 🔐		
└	removeAllFee	Private 🔐	🛑	
└	setBuyFee	Private 🔐	🛑	
└	setSellFee	Private 🔐	🛑	
└	isExcludedFromFee	Public ❗		NO❗
└	_approve	Private 🔐	🛑	
└	tradeEnable	External ❗	🛑	onlyOwner
└	_transfer	Private 🔐	🛑	
└	swapAndLiquify	Private 🔐	🛑	
└	swapAndSendMarketing	Private 🔐	🛑	
└	setSwapTokensAtAmount	External ❗	🛑	onlyOwner
└	setSwapEnabled	External ❗	🛑	onlyOwner
└	_tokenTransfer	Private 🔐	🛑	
└	_transferStandard	Private 🔐	🛑	
└	_transferToExcluded	Private 🔐	🛑	
└	_transferFromExcluded	Private 🔐	🛑	
└	_transferBothExcluded	Private 🔐	🛑	
└	excludeFromFees	External ❗	🛑	onlyOwner
└	changeMarketingWallet	External ❗	🛑	onlyOwner
└	setBuyFeePercentages	External ❗	🛑	onlyOwner
└	setSellFeePercentages	External ❗	🛑	onlyOwner

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
Variable DDX._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (contracts/TestToken.sol#864) is too similar to DDX._getValues(uint256).tTransferAmount (contracts/TestToken
.sol#820)
Variable DDX._transferStandard(address,address,uint256).rTransferAmount (contracts/TestToken.sol#1124) is too similar to DDX._transferToExcluded(address,address,uint256).tTransferAmount (con
tracts/TestToken.sol#1148)
Variable DDX._transferBothExcluded(address,address,uint256).rTransferAmount (contracts/TestToken.sol#1192) is too similar to DDX._transferStandard(address,address,uint256).tTransferAmount (c
ontracts/TestToken.sol#1126)
Variable DDX.reflectionFromToken(uint256,bool).rTransferAmount (contracts/TestToken.sol#754) is too similar to DDX._transferToExcluded(address,address,uint256).tTransferAmount (contracts/Tes
tToken.sol#1148)
Variable DDX.reflectionFromToken(uint256,bool).rTransferAmount (contracts/TestToken.sol#754) is too similar to DDX._transferBothExcluded(address,address,uint256).tTransferAmount (contracts/T
estToken.sol#1194)
Variable DDX.reflectionFromToken(uint256,bool).rTransferAmount (contracts/TestToken.sol#754) is too similar to DDX._transferFromExcluded(address,address,uint256).tTransferAmount (contracts/T
estToken.sol#1171)
Variable DDX._transferBothExcluded(address,address,uint256).rTransferAmount (contracts/TestToken.sol#1192) is too similar to DDX._transferFromExcluded(address,address,uint256).tTransferAmoun
t (contracts/TestToken.sol#1171)
Variable DDX._getValues(uint256).rTransferAmount (contracts/TestToken.sol#825) is too similar to DDX._getValues(uint256).tTransferAmount (contracts/TestToken.sol#820)
Variable DDX._transferStandard(address,address,uint256).rTransferAmount (contracts/TestToken.sol#1124) is too similar to DDX._transferBothExcluded(address,address,uint256).tTransferAmount (c
ontracts/TestToken.sol#1194)
Variable DDX._transferToExcluded(address,address,uint256).rTransferAmount (contracts/TestToken.sol#1146) is too similar to DDX._transferStandard(address,address,uint256).tTransferAmount (con
tracts/TestToken.sol#1126)
Variable DDX._transferToExcluded(address,address,uint256).rTransferAmount (contracts/TestToken.sol#1146) is too similar to DDX._getValues(uint256).tTransferAmount (contracts/TestToken.sol#82
0)
Variable DDX._getValues(uint256).rTransferAmount (contracts/TestToken.sol#825) is too similar to DDX._transferStandard(address,address,uint256).tTransferAmount (contracts/TestToken.sol#1126)
Variable DDX._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (contracts/TestToken.sol#864) is too similar to DDX._transferBothExcluded(address,address,uint256).tTransfer
Amount (contracts/TestToken.sol#1194)
Variable DDX._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (contracts/TestToken.sol#864) is too similar to DDX._transferFromExcluded(address,address,uint256).tTransfer
Amount (contracts/TestToken.sol#1171)
Variable DDX._transferStandard(address,address,uint256).rTransferAmount (contracts/TestToken.sol#1124) is too similar to DDX._transferFromExcluded(address,address,uint256).tTransferAmount (c
ontracts/TestToken.sol#1171)
Variable DDX._transferToExcluded(address,address,uint256).rTransferAmount (contracts/TestToken.sol#1146) is too similar to DDX._getTValues(uint256).tTransferAmount (contracts/TestToken.sol#8
49)
Variable DDX._getValues(uint256).rTransferAmount (contracts/TestToken.sol#825) is too similar to DDX._getTValues(uint256).tTransferAmount (contracts/TestToken.sol#849)
Variable DDX._transferToExcluded(address,address,uint256).rTransferAmount (contracts/TestToken.sol#1146) is too similar to DDX._transferBothExcluded(address,address,uint256).tTransferAmount
(contracts/TestToken.sol#1194)
Variable DDX._getValues(uint256).rTransferAmount (contracts/TestToken.sol#825) is too similar to DDX._transferToExcluded(address,address,uint256).tTransferAmount (contracts/TestToken.sol#114
8)
Variable DDX._getValues(uint256).rTransferAmount (contracts/TestToken.sol#825) is too similar to DDX._transferBothExcluded(address,address,uint256).tTransferAmount (contracts/TestToken.sol#1
194)
Variable DDX._getValues(uint256).rTransferAmount (contracts/TestToken.sol#825) is too similar to DDX._transferFromExcluded(address,address,uint256).tTransferAmount (contracts/TestToken.sol#1
171)
Variable DDX.reflectionFromToken(uint256,bool).rTransferAmount (contracts/TestToken.sol#754) is too similar to DDX._getValues(uint256).tTransferAmount (contracts/TestToken.sol#820)
Variable DDX._transferToExcluded(address,address,uint256).rTransferAmount (contracts/TestToken.sol#1146) is too similar to DDX._transferFromExcluded(address,address,uint256).tTransferAmount
(contracts/TestToken.sol#1171)
Variable DDX.reflectionFromToken(uint256,bool).rTransferAmount (contracts/TestToken.sol#754) is too similar to DDX._transferStandard(address,address,uint256).tTransferAmount (contracts/TestT
oken.sol#1126)
Variable DDX.reflectionFromToken(uint256,bool).rTransferAmount (contracts/TestToken.sol#754) is too similar to DDX._getTValues(uint256).tTransferAmount (contracts/TestToken.sol#849)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

DDX.DEAD (contracts/TestToken.sol#568) should be constant
DDX._burnFee (contracts/TestToken.sol#561) should be constant
DDX._decimals (contracts/TestToken.sol#539) should be constant
DDX._name (contracts/TestToken.sol#537) should be constant
DDX._symbol (contracts/TestToken.sol#538) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

DDX._tTotal (contracts/TestToken.sol#542) should be immutable
DDX.uniswapV2Pair (contracts/TestToken.sol#571) should be immutable
DDX.uniswapV2Router (contracts/TestToken.sol#570) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

Result => A static analysis of contract's source code has been performed using slither,

No major issues were found in the output

FUNCTIONAL TESTING

Router (PCS V2):
0xD99D1c33F9fC3444f8101754aBC46c52416550D1

1- Adding Liquidity (Passed):
liquidity added on Pancakeswap V2:

https://testnet.bscscan.com/tx/0xc00f5aeec0654eddfb122169009 9b8c5bb6a930c9510df9801de27febf192b75

no issue were found on adding liquidity.

2- Buying (liquidity, marketing, buy and burn fees = 25% max) (Passed):

https://testnet.bscscan.com/tx/0x7d80c70510f2c9aec6a97ce593 8e3ac58aff72e6068b3e18cb7a270d623dafe0

3- Selling (liquidity, marketing, buy and burn fees = 25% max) (Passed):

https://testnet.bscscan.com/tx/0x8a7d24cbbcb64efc99a6b2f28f4 00597f320c97ac64d0b31588456dadbfebf18

4-Transferring (sell fees apply for transfer fees except burning) (Passed):

https://testnet.bscscan.com/tx/0x9d66ebbb55d139a4ea897a920a 1cd05926800912a6cd23b7024143b5e0c1c8c5

FUNCTIONAL TESTING

5-Auto Liquidity(Passed):

https://testnet.bscscan.com/token/0x11859cdc6aa8e441a40af3dad9eb347c16260891?a=0x000000000000000000000000000000000000dead

6-**Internal Swap(Passed):**

https://testnet.bscscan.com/address/0x4151d3697d9745d75a02dfcefce63cb5232d2ae8#internaltx

MANUAL TESTING

NO ISSUES FOUND

Social Media Overview

Here are the Social Media Accounts of

Diamond Dex

https://t.me/DiamondDex

https://twitter.com/DiamondDexnft

https://www.diamonddextoken.com/

DISCLAIMER

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed. The Auditace team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Auditace receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token. The Auditace team disclaims any liability for the resulting losses.

ABOUT AUDITACE

We specializes in providing thorough and reliable audits for Web3 projects. With a team of experienced professionals, we use cutting-edge technology and rigorous methodologies to evaluate the security and integrity of blockchain systems. We are committed to helping our clients ensure the safety and transparency of their digital assets and transactions.

https://auditace.tech/

https://t.me/Audit_Ace

https://twitter.com/auditace_

https://github.com/Audit-Ace