AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

WWD

DATED : 26 June 23'

AUDIT SUMMARY

Project name – WWD

Date: 26 June, 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	1	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 conducted on the BSC Test network, and each test has a corresponding transaction attached to it. These tests can be found in the "Functional Tests" section of the report.

3- Slither :
The code has undergone static analysis using Slither.

Testnet version:
The tests were performed using the contract deployed on the BSC Testnet, which can be found at the following address:
https://testnet.bscscan.com/token/0xAC359AaDe930096bdAad3976813C4572330c134E

Token Information

Token Name : WorldWarDoge

Token Symbol: WWD

Decimals: 18

Token Supply: 1,000,000,000

Token Address:
0x8De5f00225728d4872FA903Fa3DC9AB4C2241eAC

Checksum:
039415dc98eee91d346c1ba2e74a2f67f9376c5d

Owner:
0xB89020d6FCc5c97F6C4112f08d197e5363F918E8
(at time of writing the audit)

Deployer:
0xb89020d6fcc5c97f6c4112f08d197e5363f918e8

TOKEN OVERVIEW

Fees:

Buy Fees: 0%

Sell Fees: 2%

Transfer Fees: 0%

Fees Privilege: Owner

Ownership: Owned

Minting: none

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: - - including in fees

- excluding from fees

- initial distribution of the tokens

- enabling trades

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.

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

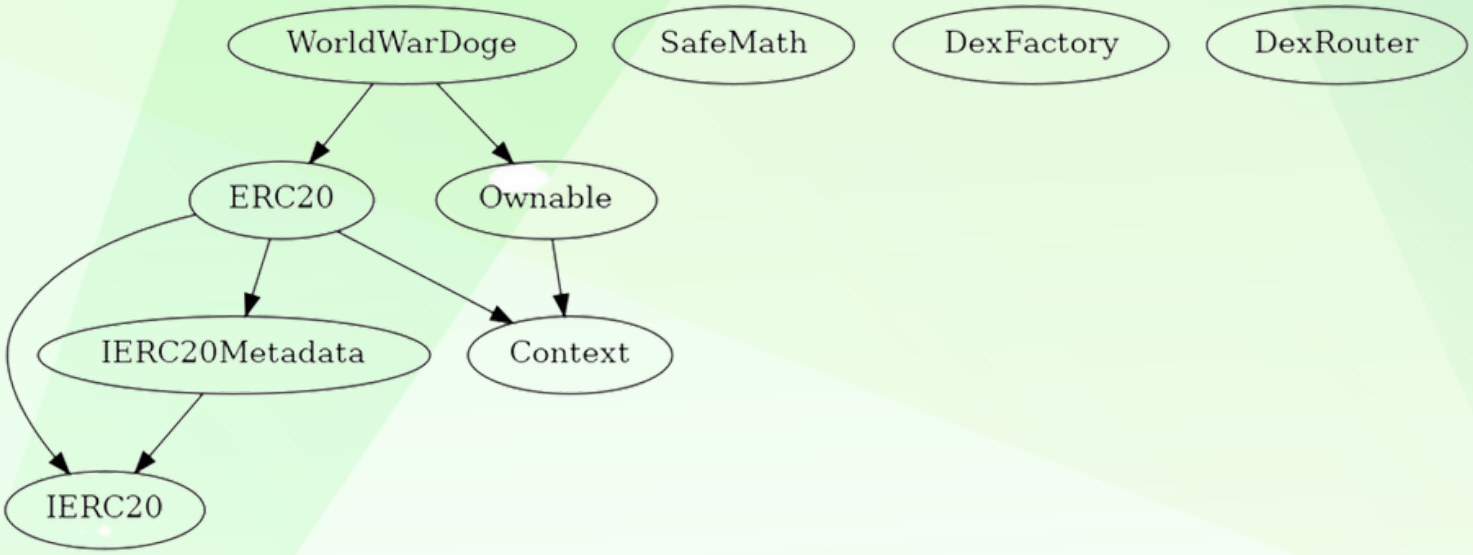
CLASSIFICATION OF RISK

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	1
Medium-Risk	0
Low-Risk	0
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- Owner is not able to change sell fees 2%
- Owner is not able to set fee on buy or transfers
- Owner is not able to blacklist an arbitrary address.
- Owner is not able to disable trades
- Owner is not able to set max buy/sell/transfer/hold amount to 0
- Owner is not able to mint new tokens
- Owner must enable trades manually

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
IERC20Metadata	Interface	IERC20		
└	name	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	decimals	External ❗		NO ❗
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
ERC20	Implementation	Context, IERC20, IERC20Metadata		
└	<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 ❗
└	_transfer	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
└	_afterTokenTransfer	Internal 🔒	🛑	
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO ❗
└	owner	Public ❗		NO ❗

CONTRACT ASSESMENT

└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_setOwner	Private 🔐	🛑	
SafeMath	Library		
└	tryAdd	Internal 🔒		
└	trySub	Internal 🔒		
└	tryMul	Internal 🔒		
└	tryDiv	Internal 🔒		
└	tryMod	Internal 🔒		
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	sub	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
DexFactory	Interface		
└	createPair	External ❗	🛑	NO ❗
DexRouter	Interface		
└	factory	External ❗		NO ❗
└	WETH	External ❗		NO ❗
└	addLiquidityETH	External ❗	💵	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
WorldWarDoge	Implementation	ERC20, Ownable	
└	<Constructor>	Public ❗	🛑	ERC20
└	enableTrading	External ❗	🛑	onlyOwner
└	setWhitelistStatus	External ❗	🛑	onlyOwner
└	checkWhitelist	External ❗		NO ❗
└	_takeTax	Internal 🔒	🛑	
└	_transfer	Internal 🔒	🛑	
└	withdrawStuckETH	External ❗	🛑	onlyOwner
└	withdrawStuckTokens	External ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💵	NO ❗

CONTRACT ASSESMENT

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
Context._msgData() (contracts/Token.sol#31-33) is never used and should be removed
ERC20._burn(address,uint256) (contracts/Token.sol#139-154) is never used and should be removed
SafeMath.add(uint256,uint256) (contracts/Token.sol#249-251) is never used and should be removed
SafeMath.div(uint256,uint256) (contracts/Token.sol#261-263) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/Token.sol#276-281) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/Token.sol#265-267) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/Token.sol#283-288) is never used and should be removed
SafeMath.mul(uint256,uint256) (contracts/Token.sol#257-259) is never used and should be removed
SafeMath.sub(uint256,uint256) (contracts/Token.sol#253-255) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (contracts/Token.sol#269-274) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (contracts/Token.sol#208-214) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (contracts/Token.sol#235-240) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (contracts/Token.sol#242-247) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (contracts/Token.sol#223-233) is never used and should be removed
SafeMath.trySub(uint256,uint256) (contracts/Token.sol#216-221) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.17 (contracts/Token.sol#6) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.20 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in WorldWarDoge.withdrawStuckETH() (contracts/Token.sol#402-408):
        - (success) = address(msg.sender).call{value: balance}() (contracts/Token.sol#406)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function DexRouter.WETH() (contracts/Token.sol#298) is not in mixedCase
Parameter WorldWarDoge.setWhitelistStatus(address,bool)._wallet (contracts/Token.sol#360) is not in mixedCase
Parameter WorldWarDoge.setWhitelistStatus(address,bool)._status (contracts/Token.sol#360) is not in mixedCase
Parameter WorldWarDoge.checkWhitelist(address)._wallet (contracts/Token.sol#365) is not in mixedCase
Parameter WorldWarDoge.withdrawStuckTokens(address).ERC20_token (contracts/Token.sol#410) is not in mixedCase
Constant WorldWarDoge._totalSupply (contracts/Token.sol#319) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

WorldWarDoge.burnAddress (contracts/Token.sol#336) should be constant
WorldWarDoge.sellTaxes (contracts/Token.sol#326) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

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):
https://testnet.bscscan.com/tx/0x00334c01765920eb6e98d2b7a6608768689f5fa5403cadcfdd6f0fc6d50e9e97

2- Buying when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x43b5b860fbd2c1a57ee220e497202de72e4644ae99ccbb087eb50acc87f40e25

3- Selling when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x951f9548f01668694f25767988702efd13e5f4fb53b37b752cfee26c20784f8e

4- Transferring when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0x3e2797d35d5141230584a5fd03181657d48f42f4b251101d87205c09a54bcfc8

5- Buying when not excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0x0753929a51e54477514dc4aa02232b6eded4560a7560b92827fd8a83469a8b42

6- Selling when not excluded from fees (2% tax) (passed):
https://testnet.bscscan.com/tx/0xefc1cb807f77705df2c489efdb1e0a3a3b411ffc3a96036b960c9a8c347a4f86

FUNCTIONAL TESTING

7- Transferring when not excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0x02630ea5f07bb9777c7271fad145c4875b63f252a22fc5bf1cb525edf5b5699c

8- Burns (passed):

https://testnet.bscscan.com/tx/0xefc1cb807f77705df2c489efdb1e0a3a3b411ffc3a96036b960c9a8c347a4f86

FUNCTIONAL TESTING

Centralization – Trades must be enabled

Severity: High

function: enableTrading

Status: Resolved

Overview:

The smart contract owner must enable trades for holders. If trading remain disabled, no one would be able to buy/sell/transfer tokens.

```
function enableTrading() external onlyOwner {
    require(!tradingEnabled, "Trading is already enabled");
    tradingEnabled = true;
    startTradingBlock = block.number;
}
```

Suggestion

To mitigate this centralization issue, we propose the following options:

1. Renounce Ownership: Consider relinquishing control of the smart contract by renouncing ownership. This would remove the ability for a single entity to manipulate the router, reducing centralization risks.
2. Multi-signature Wallet: Transfer ownership to a multi-signature wallet. This would require multiple approvals for any changes to the mainRouter, adding an additional layer of security and reducing the centralization risk.
3. Transfer ownership to a trusted and valid 3rd party in order to guarantee enabling of the trades

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