AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Goge Token

DATED : 18 JAN 23'

AUDIT SUMMARY

Project name – Goge

TimeLine- 18 January , 2023

Method- Manual Review ,Functional Testing, Automated Testing etc.

Scope of Audit- Audit Ace was consulted to conduct the smart contract audit of the solidity source codes. The audit scope of work is strictly limited to mentioned solidity file(s) only:
Goge.*sol*

Audit Status: Failed (Mint function and bad code practices which failes the audit report.)

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	2	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- Goerli:
all tests were done on Goerli network, each test has its transaction has attached to it.

3- Slither : Static Analysis

TESTNET LINKS

1- Deployment:

https://goerli.etherscan.io/token/0x7742d14c90d9a0d49e01e8ddff41f84aaa12e892#code

Token Address:
0xb2AF03a018bFD19d9F784909bD1C6D6DD42950b1

Checksum:
f0e4c2f76c58916ec258f246851bea091d14d4247a2fc3e18694461b1816e13b

Deployer:
0x6334BAE02114C080F05E6D58b65A1d7926FbbeBc

Owner:
0x97114295b2262ce2dd65b74f94c0198523a60d96

TOKEN OVERVIEW

Fees:

Buy Fees: 0%

Sell Fees: 0%

Transfer Fees: 0%

Fees Privilige: Owner

Ownership : Owned

Minting: Yes mint function

Max Tx Amount/ Max Wallet Amount:NO

Blacklist: no

Other Priviliges: none

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

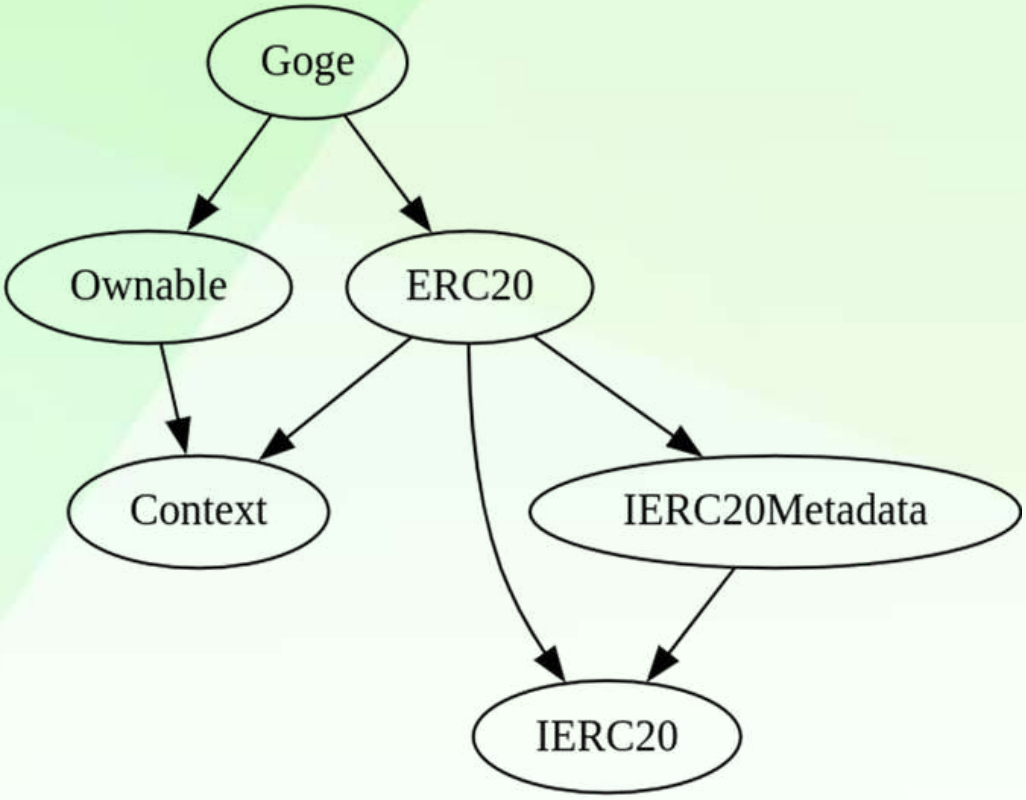
CLASSIFICATION OF RISK

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	2
High-Risk	0
Medium-Risk	0
Low-Risk	0
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- Owner is not able to change taxes (0% tax)
- 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 able to mint new tokens

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Goge	Implementation	ERC20, Ownable		
└	<Constructor>	Public !	🛑	ERC20
└	mint	Public !	🛑	onlyOwner
└	setTaxWallet	Public !	🛑	onlyOwner
└	setTaxRate	Public !	🛑	onlyOwner
Ownable	Implementation	Context		
└	<Constructor>	Public !	🛑	NO !
└	owner	Public !		NO !
└	_checkOwner	Internal 🔒		
└	renounceOwnership	Public !	🛑	onlyOwner
└	transferOwnership	Public !	🛑	onlyOwner
└	_transferOwnership	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 🔒	🛑	
└	_spendAllowance	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
└	_afterTokenTransfer	Internal 🔒	🛑	
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
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 ❗

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
ERC20.taxWallet (contracts/ERC20.sol#44) is never initialized. It is used in:
        - ERC20.transfer(address,uint256) (contracts/ERC20.sol#115-123)
        - ERC20.transferFrom(address,address,uint256) (contracts/ERC20.sol#164-178)
ERC20.taxRate (contracts/ERC20.sol#45) is never initialized. It is used in:
        - ERC20.transfer(address,uint256) (contracts/ERC20.sol#115-123)
        - ERC20.transferFrom(address,address,uint256) (contracts/ERC20.sol#164-178)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.9']
        - ^0.8.0 (contracts/Context.sol#4)
        - ^0.8.0 (contracts/ERC20.sol#4)
        - ^0.8.0 (contracts/IERC20.sol#4)
        - ^0.8.0 (contracts/IERC20Metadata.sol#4)
        - ^0.8.0 (contracts/Ownable.sol#4)
        - ^0.8.9 (contracts/token.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Context._msgData() (contracts/Context.sol#21-23) is never used and should be removed
ERC20._burn(address,uint256) (contracts/ERC20.sol#296-312) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (contracts/Context.sol#4) allows old versions
Pragma version^0.8.0 (contracts/ERC20.sol#4) allows old versions
Pragma version^0.8.0 (contracts/IERC20.sol#4) allows old versions
Pragma version^0.8.0 (contracts/IERC20Metadata.sol#4) allows old versions
Pragma version^0.8.0 (contracts/Ownable.sol#4) allows old versions
Pragma version^0.8.9 (contracts/token.sol#2) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter Goge.setTaxWallet(address)._newTaxWallet (contracts/token.sol#18) is not in mixedCase
Parameter Goge.setTaxRate(uint256)._newTaxRate (contracts/token.sol#22) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Goge.constructor(uint256) (contracts/token.sol#8-12) uses literals with too many digits:
        - _mint(msg.sender,100000000 * 10 ** decimals()) (contracts/token.sol#9)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

ERC20.taxRate (contracts/ERC20.sol#45) should be constant
ERC20.taxWallet (contracts/ERC20.sol#44) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

Result => as described in the report, taxWallet and taxRate are not initialized, this is because of a bad practice which is mentioned in logical issues.

MANUAL TESTING

Critical Risk Findings:

Logical –setTaxRate and setTaxWallet are not changing contract's storage, they are only changing properties on inherited class which wont have any impact on contract actual states and hence there wont be desired taxes for transfers/buys/sells

```
function setTaxWallet(address _newTaxWallet) public onlyOwner {
        ERC20.taxWallet = _newTaxWallet;
}
function setTaxRate(uint256 _newTaxRate) public onlyOwner {
        ERC20.taxRate = _newTaxRate;
}
```

Suggestions:

change the code to this

```
function setTaxWallet(address _newTaxWallet) public onlyOwner {
        taxWallet = _newTaxWallet;
}
function setTaxRate(uint256 _newTaxRate) public onlyOwner {
       taxRate = _newTaxRate;
}
```

==========

Critical Risk Findings:

Centralization –Owner is able to mint unlimited new tokens, there is not any limits for minting amount

```
function mint(address to, uint256 amount) public onlyOwner {
_mint(to, amount);
}
```

Suggestions:

there are several ways to resolve this issue:

- remove mint function
- renounce ownership to address dead

Testnet Link:

minted a large amount of tokens

https://goerli.etherscan.io/tx/0xc4dddd9415f144c4c2c6369952b172a32d38d7ff3e320efbeb1264fdb699ddbf

==========

Social Media Overview

Here are the Social Media Accounts of

Goge Labs

https://t.me/Gogelabs

https://Twitter.com/gogelabs

https://Goge.io

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