auditace.tech

AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

JOEY OGGY

DATED : 9 May 23'

AUDIT SUMMARY

Project name – JOEY OGGY

Date: 9 May, 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 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/0xd7d6316E316987eAc1B7a0FC3eB25b9f1a7A5437

Token Information

Token Name : JOEY OGGY

Token Symbol: JOEY

Decimals: 9

Token Supply: 420,000,000,000,000

Token Address:
0x753979f1b0F6b26A6B0F9409492c0CC8099ed97e

Checksum:
035da10987d6b003b2ba595b9eeba594a03fc652

Owner:
0x97b0aA30AeDCa959C394A62Ac970Be168AF3Da04
(at time of writing the audit)

Deployer:
0x97b0aA30AeDCa959C394A62Ac970Be168AF3Da04

TOKEN OVERVIEW

Fees:

Buy Fees: 0%

Sell Fees: 0%

Transfer Fees: 0%

Fees Privilege: no

Ownership: no

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: no

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

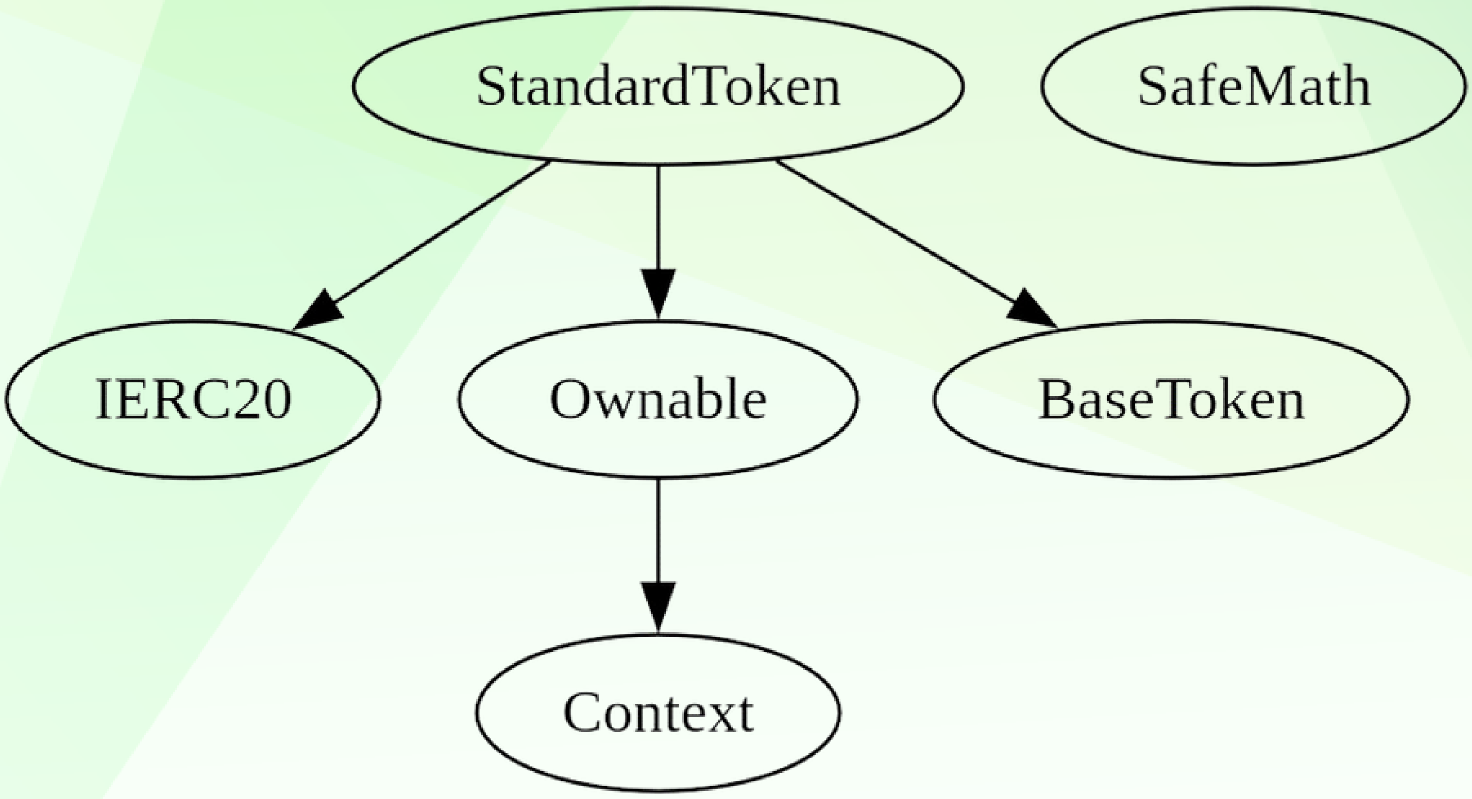
CLASSIFICATION OF RISK

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	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 (0% all)
- Owner is not able to set a max buy/transfer/wallet/sell amount
- Owner is able to blacklist an arbitrary wallet
- Owner is able to disable trades
- Owner is not able to mint new tokens

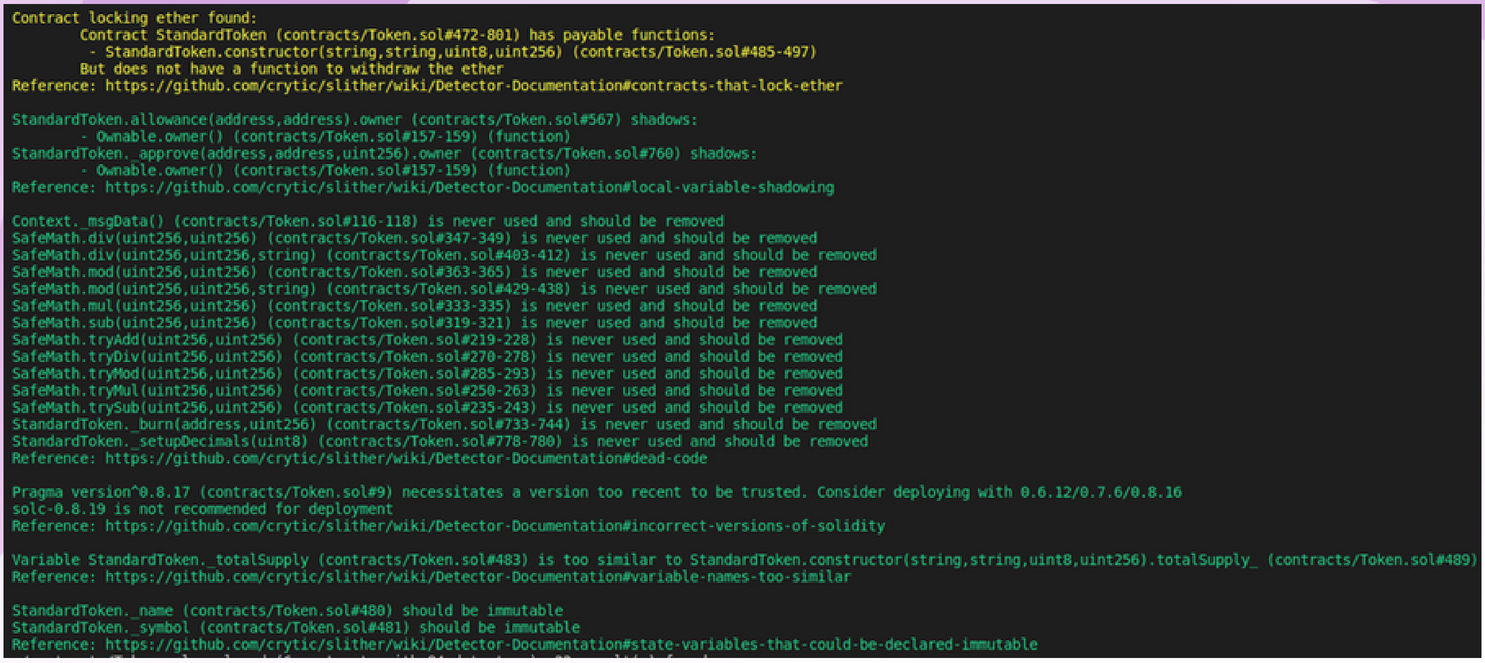
CONTRACT ASSESMENT

```
| 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❗️ |
||||||
| **Context** | Implementation |  |||
| └ | _msgSender | Internal 🔒 |   | |
| └ | _msgData | Internal 🔒 |   | |
||||||
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Public ❗️ | 🛑  |NO❗️ |
| └ | owner | Public ❗️ |   |NO❗️ |
| └ | 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 🔒 |   | |
||||||
| **BaseToken** | Implementation |  |||
||||||
| **StandardToken** | Implementation | IERC20, Ownable, BaseToken |||
| └ | <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 🔒 | 🛑 | |
| └ | _setupDecimals | Internal 🔒 | 🛑 | |
| └ | _beforeTokenTransfer | Internal 🔒 | 🛑 | |
```

Legend

```
| Symbol | Meaning |
|:--------:|-----------|
| 🛑 | Function can modify state |
| 💵 | Function is payable |
```

STATIC ANALYSIS



```
Contract locking ether found:
        Contract StandardToken (contracts/Token.sol#472-801) has payable functions:
         - StandardToken.constructor(string,string,uint8,uint256) (contracts/Token.sol#485-497)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether

StandardToken.allowance(address,address).owner (contracts/Token.sol#567) shadows:
        - Ownable.owner() (contracts/Token.sol#157-159) (function)
StandardToken._approve(address,address,uint256).owner (contracts/Token.sol#760) shadows:
        - Ownable.owner() (contracts/Token.sol#157-159) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Context._msgData() (contracts/Token.sol#116-118) is never used and should be removed
SafeMath.div(uint256,uint256) (contracts/Token.sol#347-349) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/Token.sol#403-412) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/Token.sol#363-365) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/Token.sol#429-438) is never used and should be removed
SafeMath.mul(uint256,uint256) (contracts/Token.sol#333-335) is never used and should be removed
SafeMath.sub(uint256,uint256) (contracts/Token.sol#319-321) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (contracts/Token.sol#219-228) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (contracts/Token.sol#270-278) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (contracts/Token.sol#285-293) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (contracts/Token.sol#250-263) is never used and should be removed
SafeMath.trySub(uint256,uint256) (contracts/Token.sol#235-243) is never used and should be removed
StandardToken._burn(address,uint256) (contracts/Token.sol#733-744) is never used and should be removed
StandardToken._setupDecimals(uint8) (contracts/Token.sol#778-780) 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#9) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Variable StandardToken._totalSupply (contracts/Token.sol#483) is too similar to StandardToken.constructor(string,string,uint8,uint256).totalSupply_ (contracts/Token.sol#489)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

StandardToken._name (contracts/Token.sol#480) should be immutable
StandardToken._symbol (contracts/Token.sol#481) 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

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0x632a972d7053c155c56cac513376
0d88b402453f1e3c81a93a95640d34447100

2- Buying when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x2af3952665b84a47b1b60f3717c8
2c09618c72b21d5fa04bc3c7e4cfe6bbf380

3- Selling when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x7e877ec064f99c8e90f1f580ec825
ee0c4f1de87b6e1b4649fd71fdb224525ea

4- Transferring when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0x96247a933f18da8a35623c237baa
081c4fbd793cb73ded7f2fb1028ea31e8814

5- Buying when not excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0x79d4ddd3f57bb4f12257ce8f59ad0
670ef7305d1ab9057cfd6830ae19814a59f

6- Selling when not excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0xce21383c091dd4d316637f8f68a18
387af18047b0ffc9925dd898b374f3af466

FUNCTIONAL TESTING

7- Transferring when not excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0xd4704f6c23d412beb1c828f6a348cb45e0081e880bbe815d5ab4daaba63eefdf

DISCLAIMER

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