auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Son Of Dragon

DATED : 31 Jan, 2024

AUDIT SUMMARY

Project name – Son Of Dragon

Date: 31 Jan, 2024

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	1	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/address/0x132d0729e974675cc6a3701482eb17e0184d0694#readContract

Token Information

Token Name : Son Of Dragon

Token Symbol: SOD

Decimals: 18

Token Supply: 420,000,000

Network: BscScan

Token Type: BEP-20

Token Address:
0x5367260923ce0e6aC1d4F8C3384897f8a537a7B8

Checksum:
f2032c616934aeb47e6039f76b20d2h5

Owner:
0x7e378453D71b54365A87d45a24690FBEfd247F29
(at time of writing the audit)

Deployer:
0x7e378453D71b54365A87d45a24690FBEfd247F29

TOKEN OVERVIEW

Fees:

Buy Fee: 5-10%

Sell Fee: 5-10%

Transfer Fee: 0-10%

Fees Privilege: Owner

Ownership: Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

swapBack: yes

AUDIT METHODOLOGY

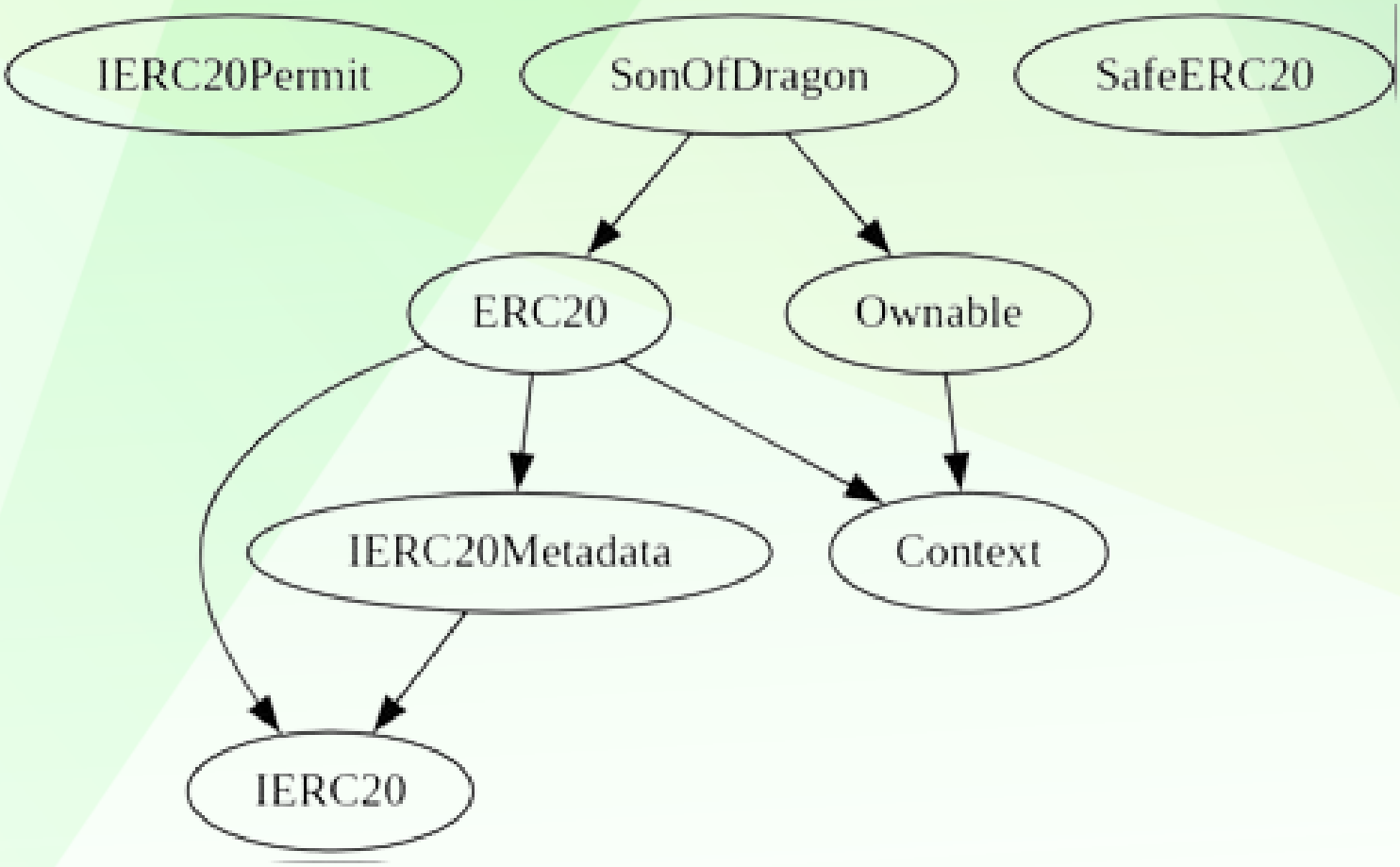
The auditing process will follow a routine as special considerations by Auditace:

- 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

INHERITANCE TREE



STATIC ANALYSIS

A static analysis of the code was performed using Slither.

No issues were found.

```
INFO:Detectors:
Address._revert(bytes,string) (test/testStake.sol#402-415) uses assembly
        - INLINE ASM (test/testStake.sol#408-411)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.functionCall(address,bytes) (test/testStake.sol#263-274) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (test/testStake.sol#284-296) is never used and should be removed
Address.functionDelegateCall(address,bytes) (test/testStake.sol#347-357) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (test/testStake.sol#359-372) is never used and should be removed
Address.functionStaticCall(address,bytes) (test/testStake.sol#320-330) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (test/testStake.sol#332-345) is never used and should be removed
Address.sendValue(address,uint256) (test/testStake.sol#250-261) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (test/testStake.sol#390-400) is never used and should be removed
Context._msgData() (test/testStake.sol#614-616) is never used and should be removed
ERC20._burn(address,uint256) (test/testStake.sol#758-774) is never used and should be removed
SafeERC20._callOptionalReturnBool(IERC20,bytes) (test/testStake.sol#589-598) is never used and should be removed
SafeERC20.forceApprove(IERC20,address,uint256) (test/testStake.sol#539-557) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (test/testStake.sol#486-499) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (test/testStake.sol#517-537) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (test/testStake.sol#501-515) is never used and should be removed
SafeERC20.safePermit(IERC20Permit,address,address,uint256,uint256,uint8,bytes32,bytes32) (test/testStake.sol#559-576) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (test/testStake.sol#474-484) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.15 (test/testStake.sol#16) allows old versions
solc-0.8.15 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (test/testStake.sol#250-261):
        - (success) = recipient.call{value: amount}() (test/testStake.sol#256)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (test/testStake.sol#298-310):
        - (success,returndata) = target.call{value: value}(data) (test/testStake.sol#308-310)
Low level call in Address.functionStaticCall(address,bytes,string) (test/testStake.sol#332-345):
        - (success,returndata) = target.staticcall(data) (test/testStake.sol#337)
Low level call in Address.functionDelegateCall(address,bytes,string) (test/testStake.sol#359-372):
        - (success,returndata) = target.delegatecall(data) (test/testStake.sol#364)
Low level call in SafeERC20._callOptionalReturnBool(IERC20,bytes) (test/testStake.sol#589-598):
        - (success,returndata) = address(token).call(data) (test/testStake.sol#593)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IUniswapV2Router01.WETH() (test/testStake.sol#21) is not in mixedCase
Function IERC20Permit.DOMAIN_SEPARATOR() (test/testStake.sol#432) is not in mixedCase
Parameter SonOfDragon.setBuyTax(uint256)._marketingTaxBuy (test/testStake.sol#961) is not in mixedCase
Parameter SonOfDragon.setSellTax(uint256)._marketingTaxSell (test/testStake.sol#973) is not in mixedCase
Parameter SonOfDragon.setTransferTax(uint256)._marketingTaxTransfer (test/testStake.sol#985) is not in mixedCase
Parameter SonOfDragon.setMarketingWallet(address)._marketingWallet (test/testStake.sol#1000) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (test/testStake.sol#26) is too similar to IUniswapV2Router01.ad
dLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (test/testStake.sol#27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
SonOfDragon.setSwapTokensAtAmount(uint256) (test/testStake.sol#1018-1031) uses literals with too many digits:
        - require(bool,string)(amount >= totalSupply() / 1_000_000,Amount must be equal or greater than 0.000001% of Total Supply) (test/testStake.sol#1023-1026)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Slither:./test/testStake.sol analyzed (12 contracts with 88 detectors), 35 result(s) found
```

FUNCTIONAL TESTING

1- Approve (passed):

https://testnet.bscscan.com/tx/0x912b27f48f9483c72d7b2faa3d916f32bfc14e197dfc3ad45972c85a2a5d2f26

2- Set buy tax(passed):

https://testnet.bscscan.com/tx/0xc914be169934bb7d1387f8f495f04e8769f7b1cb74836a0dc6bcf9eae20e7ba1

3- Set sell tax(passed):

https://testnet.bscscan.com/tx/0x76a290d7b1f798fccfbb1c072ae0567baa91ebe0fac21986c5eb46794fb0a441

3- transfer (passed):

https://testnet.bscscan.com/tx/0x417dabdf99621979e083552509c931c5b10d132fa294eaa53320e0354d0a405c

4- claim stuck tokens(passed):

https://testnet.bscscan.com/tx/0x2d5ab93ca2bd03d522cfe626cab13cfb41e3e22baf56f27ccb7fb6a55588fe58

POINTS TO NOTE

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can set the fees not more than 10%.
- The owner can change the marketing wallet.
- The owner can exclude any addresses for fees.
- The owner can claim stuck tokens.

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	1
Low-Risk	0
Gas Optimization / Suggestions	0

MANUAL TESTING

Centralization – The owner can claim struck tokens<=

Severity: Medium

Subject: claim struck tokens

Status: Open

Overview:

```
function claimStuckTokens(address token) external onlyOwner {
    require(token != address(this), "Owner cannot claim native tokens");

    if (token == address(0x0)) {
payable(msg.sender).transfer(address(this).balance);
        return;
    }
    IERC20 ERC20token = IERC20(token);
    uint256 balance = ERC20token.balanceOf(address(this));
    ERC20token.safeTransfer(msg.sender, balance);
}
```

if owner set token value to "0x0000000000000000000000000000000000000000", This function works.

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