AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Trust Trump

DATED : 18 March, 2024

AUDIT SUMMARY

Project name – Trust Trump

Date: 18 March, 2024

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

Audit Status: **Passed with Medium Risk**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	0	3	2	2
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/0xc7bd9507d91bb7d663418cfcd6f74764d7f00a19#code

Token Information

Token Name : Trust Trump

Token Symbol: TRUMP

Decimals: 18

Token Supply: 1,000,000,000

Network: BscScan

Token Type: BEP-20

Token Address:
0x0174Cc5BBB0d3D2C3571271Eaf11a52c4B921Baa

Checksum:
A2032c616934aeb47e6039f76b20dfg1

Owner:
0x7a387DE18dC57Dd3A1B655E5DA5b9B539B707bBD
(at time of writing the audit)

Deployer:
0x7a387DE18dC57Dd3A1B655E5DA5b9B539B707bBD

TOKEN OVERVIEW

Fees:

Buy Fee: 3-20%

Sell Fee: 5-20%

Transfer Fee: 0%

Fees Privilege: Owner

Ownership: Owned

Minting: None

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

AUDIT METHODOLOGY

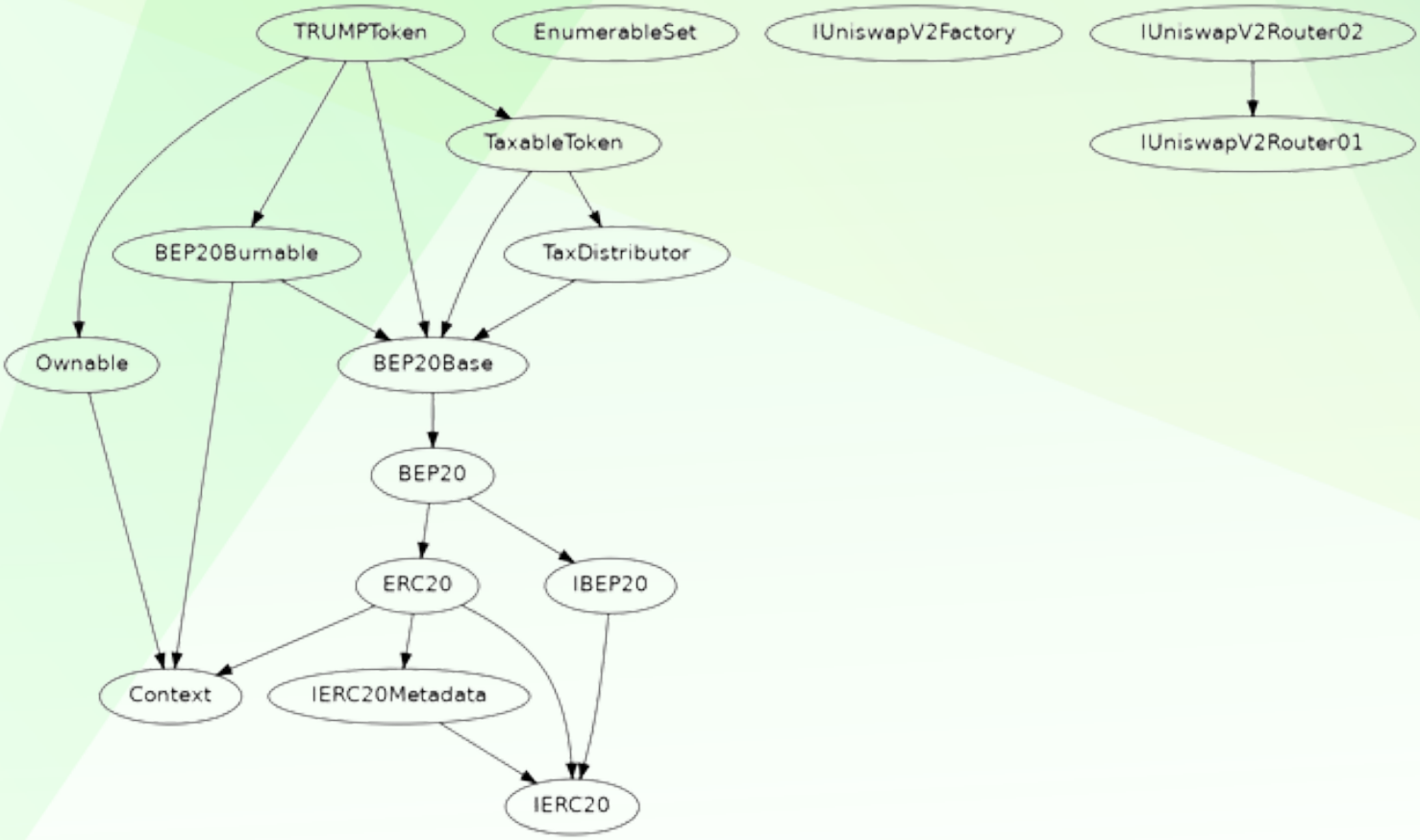
The auditing process will follow a routine as special considerations by Auditace:

- 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

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

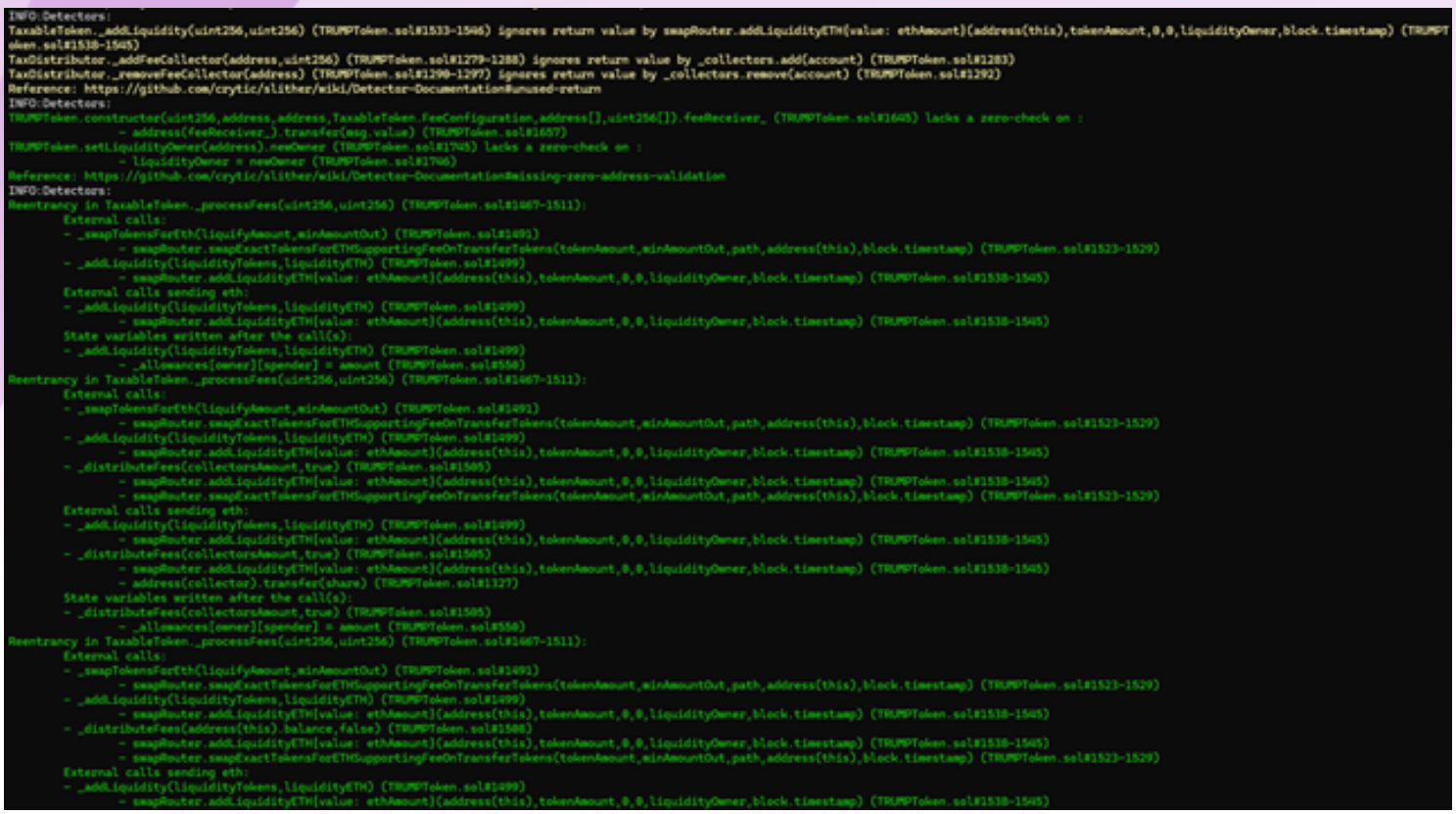
INHERITANCE TREE



STATIC ANALYSIS

A static analysis of the code was performed using Slither.

No issues were found.



STATIC ANALYSIS

```
INFO:Detectors:
EnumerableSet.values(EnumerableSet.Bytes32Set) (TRUMPToken.sol#1090-1100) uses assembly
        - INLINE ASM (TRUMPToken.sol#1095-1097)
EnumerableSet.values(EnumerableSet.AddressSet) (TRUMPToken.sol#1164-1174) uses assembly
        - INLINE ASM (TRUMPToken.sol#1169-1171)
EnumerableSet.values(EnumerableSet.UintSet) (TRUMPToken.sol#1238-1248) uses assembly
        - INLINE ASM (TRUMPToken.sol#1243-1245)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Context._msgData() (TRUMPToken.sol#30-32) is never used and should be removed
EnumerableSet._values(EnumerableSet.Set) (TRUMPToken.sol#1024-1026) is never used and should be removed
EnumerableSet.add(EnumerableSet.Bytes32Set,bytes32) (TRUMPToken.sol#1040-1042) is never used and should be removed
EnumerableSet.add(EnumerableSet.UintSet,uint256) (TRUMPToken.sol#1188-1190) is never used and should be removed
EnumerableSet.at(EnumerableSet.Bytes32Set,uint256) (TRUMPToken.sol#1078-1080) is never used and should be removed
EnumerableSet.at(EnumerableSet.UintSet,uint256) (TRUMPToken.sol#1226-1228) is never used and should be removed
EnumerableSet.contains(EnumerableSet.Bytes32Set,bytes32) (TRUMPToken.sol#1057-1059) is never used and should be removed
EnumerableSet.contains(EnumerableSet.UintSet,uint256) (TRUMPToken.sol#1205-1207) is never used and should be removed
EnumerableSet.length(EnumerableSet.Bytes32Set) (TRUMPToken.sol#1064-1066) is never used and should be removed
EnumerableSet.length(EnumerableSet.UintSet) (TRUMPToken.sol#1212-1214) is never used and should be removed
EnumerableSet.remove(EnumerableSet.Bytes32Set,bytes32) (TRUMPToken.sol#1050-1052) is never used and should be removed
EnumerableSet.remove(EnumerableSet.UintSet,uint256) (TRUMPToken.sol#1198-1200) is never used and should be removed
EnumerableSet.values(EnumerableSet.AddressSet) (TRUMPToken.sol#1164-1174) is never used and should be removed
EnumerableSet.values(EnumerableSet.Bytes32Set) (TRUMPToken.sol#1090-1100) is never used and should be removed
EnumerableSet.values(EnumerableSet.UintSet) (TRUMPToken.sol#1238-1248) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.10 (TRUMPToken.sol#3) allows old versions
solc-0.8.24 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Variable BEP20Base.TOKEN_CODE (TRUMPToken.sol#642) is not in mixedCase
Function IUniswapV2Router01.WETH() (TRUMPToken.sol#735) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Reentrancy in TaxDistributor._distributeFees(uint256,bool) (TRUMPToken.sol#1312-1335):
        External calls:
        - address(collector).transfer(share) (TRUMPToken.sol#1327)
        State variables written after the call(s):
        - _transfer(address(this),collector,share) (TRUMPToken.sol#1325)
                - _balances[from] = fromBalance - amount (TRUMPToken.sol#465)
                - _balances[to] += amount (TRUMPToken.sol#468)
        Event emitted after the call(s):
        - FeeCollected(collector,share) (TRUMPToken.sol#1329)
        - Transfer(from,to,amount) (TRUMPToken.sol#471)
                - _transfer(address(this),collector,share) (TRUMPToken.sol#1325)
Reentrancy in TaxableToken._transfer(address,address,uint256) (TRUMPToken.sol#1567-1620):
        External calls:
        - _processFees(numTokensToSwap,0) (TRUMPToken.sol#1596)
                - address(collector).transfer(share) (TRUMPToken.sol#1327)
        External calls sending eth:
        - _processFees(numTokensToSwap,0) (TRUMPToken.sol#1596)
                - swapRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,liquidityOwner,block.timestamp) (TRUMPToken.sol#1538-1545)
                - address(collector).transfer(share) (TRUMPToken.sol#1327)
        State variables written after the call(s):
        - super._transfer(from,BURN_ADDRESS,burnAmount) (TRUMPToken.sol#1607)
                - _balances[from] = fromBalance - amount (TRUMPToken.sol#465)
```

```
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (TRUMPToken.sol#740) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (TRUMPToken.sol#741)
Variable TaxDistributor._collectors (TRUMPToken.sol#1255) is too similar to TaxDistributor.constructor(address[],uint256[]).collectors_ (TRUMPToken.sol#1264)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Slither:TRUMPToken.sol analyzed (16 contracts with 93 detectors), 47 result(s) found
```

FUNCTIONAL TESTING

1- Approve (passed):
https://testnet.bscscan.com/tx/0x55d8e077cc2ed9d9e322d2eba6317b83b64b1161ad3bbec773522ea7b518bfdd

2- Add Fee Collector (passed):
https://testnet.bscscan.com/tx/0x1e9d4b8386e1fe0b92fd3240ac39d6a690c3b6cd8326cda5cb71640fe9e48612

3- Decrease Allowance (passed):
https://testnet.bscscan.com/tx/0x9c9a0c27a7994be930ca3b80f4c0f8b1a76c0b4eae38dad2b7d27a43e0e38fbb

4- Set Liquidity Owner (passed):
https://testnet.bscscan.com/tx/0xed9751d75ce2bfe5b5ca4b4a932b3072959c5b632c5ece227e94aa2b4ae02601

5- Set Fee Configuration (passed):
https://testnet.bscscan.com/tx/0x55cd70c05b081ae4b02f9c0dbbe1b0c716bb316f65fa383e520e225b43b31d4c

6- Burn (passed):
https://testnet.bscscan.com/tx/0xc288c3087ba355343b5f0324344dc56b3a6278f03c960c9962982a2c9d5f45d5

POINTS TO NOTE

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can burn tokens.
- The owner can set auto-process fees.
- The owner can add a fee collector.
- The owner sets distribution fees.
- The owner sets the process fees.
- The owner can remove the fee collector.
- The owner set liquidity address.
- The owner can set the fee configuration.
- The owner can set swap router

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	3
Low-Risk	2
Gas Optimization / Suggestions	2

MANUAL TESTING

Centralization – Liquidity is added to EOA

Severity: Medium

Function: _addLiquidity

Status: Open

Overview:

Liquidity is added to EOA. It may be drained by the liquidityOwner.

```
function _addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
        // approve token transfer to cover all possible scenarios
        _approve(address(this), address(swapRouter), tokenAmount);

        // add the liquidity
        swapRouter.addLiquidityETH{value: ethAmount}(
            address(this),
            tokenAmount,
            0, // slippage is unavoidable
            0, // slippage is unavoidable
            liquidityOwner,
            block.timestamp
        );
```

Suggestion:

It is suggested that the address should be a contract address or a dead address.

MANUAL TESTING

Centralization – Owner can Burn Tokens

Severity: Medium

Function: BurnFrom

Status: Open

Overview:

The owner can burn tokens without approval from any wallet.

```
function burnFrom(address account, uint256 amount) external override onlyOwner {
        _burnFrom(account, amount);
    }
```

Suggestion:

There should not be any burning without any allowance from the user.

MANUAL TESTING

Centralization – Missing Require Check

Severity: Medium

Function: setLiquidityOwner

Status: Open

Overview:

The owner can set any arbitrary address excluding zero address as this is not recommended because if the owner sets the address to the contract address, then the ETH will not be sent to that address and the transaction will fail and this will lead to a potential honeypot in the contract.

```
function setLiquidityOwner(address newOwner) external override onlyOwner {
        liquidityOwner = newOwner;
    }
```

Suggestion:

It is recommended that the address should not be able to be set as a contract address.

MANUAL TESTING

Centralization – Missing Zero Address

Severity: Low

Function: setLiquidityOwner

Status: Open

Overview:

Functions can take a zero address as a parameter (0x00000...). If a function parameter of address type is not properly validated by checking for zero addresses, there could be serious consequences for the contract's functionality.

```
function setLiquidityOwner(address newOwner) external override onlyOwner {
        liquidityOwner = newOwner;
    }
```

Suggestion:

It is suggested that the address should not be zero or dead.

MANUAL TESTING

Centralization – Missing Events

Severity: Low

Function: Missing Events

Status: Open

Overview:
They serve as a mechanism for emitting and recording data onto the blockchain, making it transparent and easily accessible.

```
function setLiquidityOwner(address newOwner) external override onlyOwner {
        liquidityOwner = newOwner;
    }
```

Suggestion:
Emit an event for critical changes.

MANUAL TESTING

Optimization

Severity: Informational

Subject: Floating Pragma

Status: Open

Overview:

It is considered best practice to pick one compiler version and stick with it. With a floating pragma, contracts may accidentally be deployed using an outdated.

```
pragma solidity ^0.8.10;
```

Suggestion:

Adding the latest constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.

MANUAL TESTING

Optimization

Severity: Optimization

Subject: Remove unused code

Status: Open

Overview:

Unused variables are allowed in Solidity, and they do. not pose a direct security issue. It is the best practice though to avoid them

```
function _msgData() internal view virtual returns (bytes calldata) {
        return msg.data;
    }
function add(Bytes32Set storage set, bytes32 value) internal returns (bool) {
        return _add(set._inner, value);
    }
function remove(Bytes32Set storage set, bytes32 value) internal returns (bool) {
        return _remove(set._inner, value);
    }
function contains(Bytes32Set storage set, bytes32 value) internal view returns
(bool) {
        return _contains(set._inner, value);
    }
function length(Bytes32Set storage set) internal view returns (uint256) {
        return _length(set._inner);
    }
function at(Bytes32Set storage set, uint256 index) internal view returns (bytes32)
{
        return _at(set._inner, index);
    }
function values(Bytes32Set storage set) internal view returns (bytes32[] memory) {
        bytes32[] memory store = _values(set._inner);
        bytes32[] memory result;

        /// @solidity memory-safe-assembly
        assembly {
            result := store
        }

        return result;
    }
```

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