auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Token2

DATED : 06 November 23'

MANUAL TESTING

Centralization – Enabling Trades
Severity: High
function: EnableTrading
Status: Open

Overview:
The EnableTrading function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

```
function EnableTrading() external onlyOwner {
 require(!tradingEnabled, "Cannot re-enable trading");
 tradingEnabled = true;
 providingLiquidity = true;
 genesis_block = block.number;
}
```

Suggestion
To reduce centralization and potential manipulation, consider one of the following approaches:
1.Automatically enable trading after a specified condition, such as the completion of a presale, is met.
2.If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can provide investors with more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad faith actions by the original owner

AUDIT SUMMARY

Project name – Token2

Date: 06 November 2023

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

Audit Status: **Passed with High risk**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	1	0	2	1
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/0xb44108701f21ec5d8bcdcc2679bf49411a178d3a

Token Information

Token Address :
0xdA561125107BF4C406086C1b49B4Bc0834696879

Name: Token2

Symbol: Token 2.0

Decimals: 18

Network: Binance smart chain

Token Type: ERC20

Owner: 0x44F9528dfD94B24c2d5c549614d809323Ee59523

Deployer: 0x44f9528dfd94b24c2d5c549614d809323ee59523

Token Supply: 1,000,000

Testnet version:
The tests were performed using the contract deployed on the Binance smart chain Testnet, which can be found at the following address:
https://testnet.bscscan.com/address/0xb44108701f21ec5d8bcdcc2679bf49411a178d3a

TOKEN OVERVIEW

buy fee: 0-5%

Sell fee: 0-15%

transfer fee: 0%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max Tx: No

Blacklist: No

Other Privileges:

- Initial distribution of the tokens
- Modifying fees
- Enabling trades
-bulk exempts fee
-claim stuck tokens.
— Update deadline

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

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.

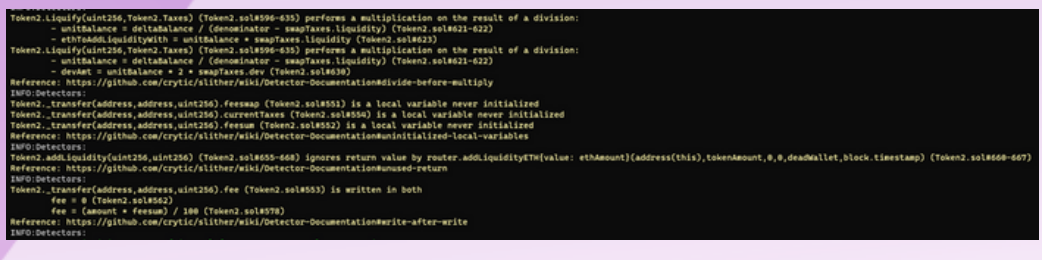
Findings

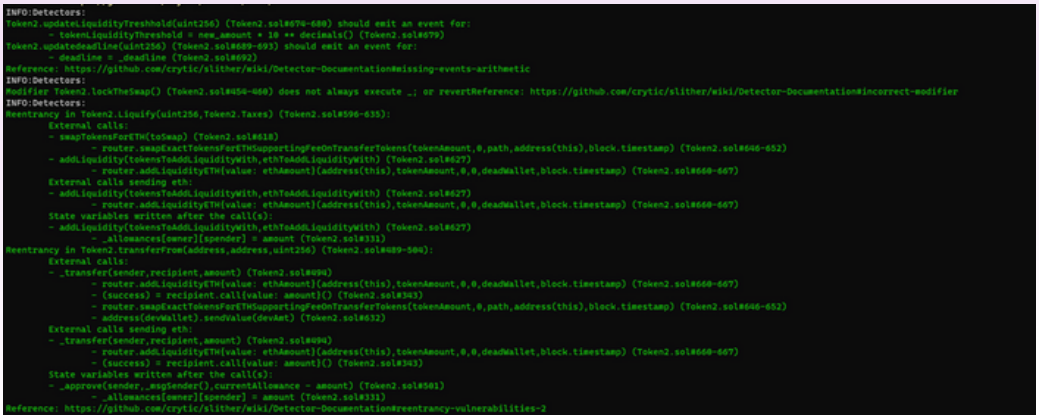
Severity	Found
Critical	0
High-Risk	1
Medium-Risk	0
Low-Risk	2
Gas Optimization / Suggestions	1

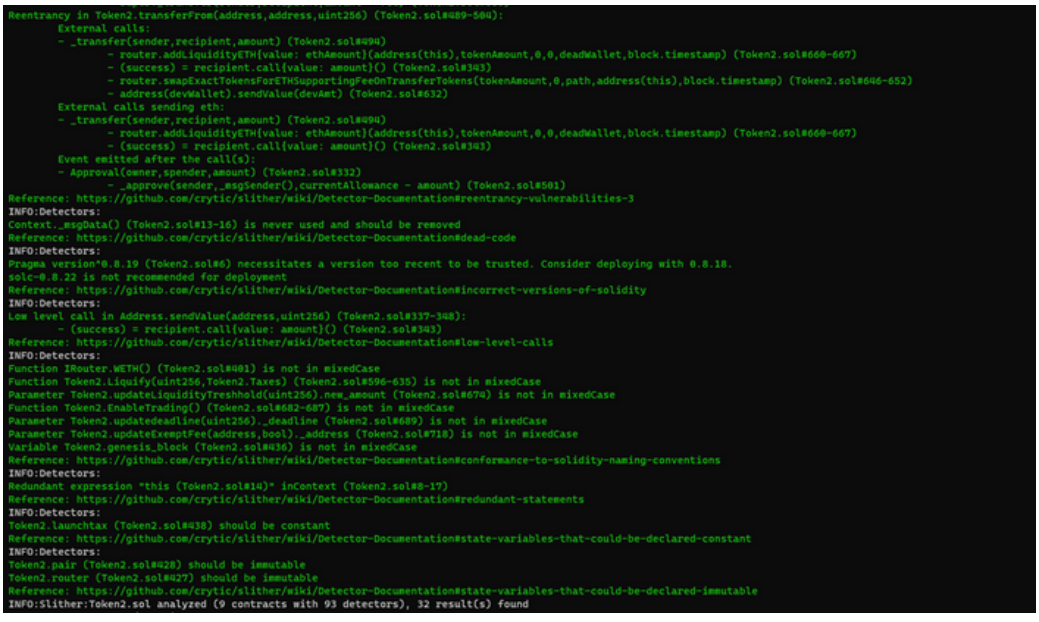
POINTS TO NOTE

- Owner can enable/disable swapping
- Owner can change the swap threshold of not more than 1% of total supply
- Owner can enable trading only once
- Owner can update the deadline not more than 5 blocks
- Owner can enable/disable wallet limit
- Owner can update tax buy not more than 5 and sell not more than 15
- Owner can exclude wallets from maximum transaction limit.
- Owner can exclude multiple address from fees
- Owner can claim ETH from the contract
- Owner can claim stuck tokens

STATIC ANALYSIS







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- Enable Tarding (passed):
https://testnet.bscscan.com/tx/0x3aa7d9e55ae8f5d2e99649b769bd5cd3d4e185352de1703eafbd8b334f95be0b

2- Transfer Ownership (passed):
https://testnet.bscscan.com/tx/0x13bb8af7ecda105b8a61cb769bbdcde27793646e791be68f1ce1cc4cca4e1108

3- Bulk Exempt Fee (passed):
https://testnet.bscscan.com/tx/0x581e858130a5e966b6fd62f2fbf9b0deb34dc6bf70028632273cb412180e2d59

4- Transfer (passed):
https://testnet.bscscan.com/tx/0xbe07eeab3ed6d2f8845663fb23da7bbfabc93583fdb1e5ecc240894a2bca1478

5- Approve (passed):
https://testnet.bscscan.com/tx/0xa60bd25d8aaf8c9493212fa285bd0d413bac155a30f00969a96fe8ca59ab52ab

7- Renounce Ownership (passed):
https://testnet.bscscan.com/tx/0xc148653af553bb82245c2c767f1234a349cb50e218eeb595162f434cfa4163c8

MANUAL TESTING

Centralization – Enabling Trades
Severity: High
function: EnableTrading
Status: Open

Overview:
The EnableTrading function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

```
function EnableTrading() external onlyOwner {
 require(!tradingEnabled, "Cannot re-enable trading");
 tradingEnabled = true;
 providingLiquidity = true;
 genesis_block = block.number;
}
```

Suggestion
To reduce centralization and potential manipulation, consider one of the following approaches:
1.Automatically enable trading after a specified condition, such as the completion of a presale, is met.
2.If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can provide investors with more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad faith actions by the original owner

MANUAL TESTING

Severity: Low
subject: floating Pragma Solidity version
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 ^0.8.19;

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

Severity: Low
subject: 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 updateLiquidityTreshhold(uint256 new_amount) external onlyOwner {
   require(
    new_amount <= 1e7,
    "Swap threshold amount should be lower or equal to 1% of tokens"
   );
   tokenLiquidityThreshold = new_amount * 10 ** decimals();
 }

 function EnableTrading() external onlyOwner {
   require(!tradingEnabled, "Cannot re-enable trading");
   tradingEnabled = true;
   providingLiquidity = true;
   genesis_block = block.number;
 }

 function updatedeadline(uint256 _deadline) external onlyOwner {
   require(!tradingEnabled, "Can't change when trading has started");
   require(_deadline < 5, "Deadline should be less than 5 Blocks");
   deadline = _deadline;
 }

 function updateDevWallet(address newWallet) external onlyOwner
   require(newWallet != address(0), "Fee Address cannot be zero address");
   devWallet = newWallet;
 }
```

MANUAL TESTING

```
function updateTax(
uint256 buyDevTax,
uint256 buyLiquidityTax,
uint256 sellDevTax,
uint256 sellLiquidityTax
) external onlyOwner {
require(
(buyDevTax + buyLiquidityTax) <= 5,
"Can't set tax greater than 5%"
);
require(
(sellDevTax + sellLiquidityTax) <= 15,
"Can't set tax greater than 15%"
);
taxes = Taxes(buyDevTax, buyLiquidityTax);
sellTaxes = Taxes(sellDevTax, sellLiquidityTax);
}

function updateExemptFee(address _address, bool state) external onlyOwner {.
exemptFee[_address] = state;
}
```

MANUAL TESTING

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 best practice though to avoid them

```
function _msgData() internal view virtual returns (bytes calldata) {
    this; // silence state mutability warning without generating bytecode - see https://github.com/ethereum/solidity/issues/2691
    return msg.data;
  }
}
```

Suggestion
To reduce high gas fees. It is suggested to remove. unused code from the contract.

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