AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Agronium

DATED : 15 Jan, 2024

MANUAL TESTING

Centralization – Enabling Trades
Severity: High
function: Enabling Trades
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, "Trading is already enabled");
  tradingEnabled = true;
  startTradingBlock = 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 give investors more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad-faith actions by the original owner.

AUDIT SUMMARY

Project name – Agronium

Date: 15 Jan, 2024

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	1	0	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/0xc3c01ad7f17ed2c3bc70b9b92abe7d2113a933d9#code

Token Information

Token Name : Agronium

Token Symbol: AGC

Decimals: 18

Token Supply: 1,000,000,000

Network: BscScan

Token Type: BEP-20

Token Address:
0xd79f6488650Fa87d4A69935910Af4eC12e1748D3

Checksum:
Ab7acbefe2a12642d388659dffd20712

Owner:
0xCad58De6002dC91CF157ac4Ffb56B627E07EdF6A
(at time of writing the audit)

Deployer:
0xCad58De6002dC91CF157ac4Ffb56B627E07EdF6A

TOKEN OVERVIEW

Fees:

Buy Fee: 0-5%

Sell Fee: 0-5%

Transfer Fee: 0-5%

Fees Privilege: Owner

Ownership: Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges:

- **Whitelist to transfer without enabling trades**
- **Enabling trades**

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**	1
◆ **Medium-Risk**	1
◆ **Low-Risk**	0
◆ **Gas Optimization / Suggestions**	2

INHERITANCE TREE
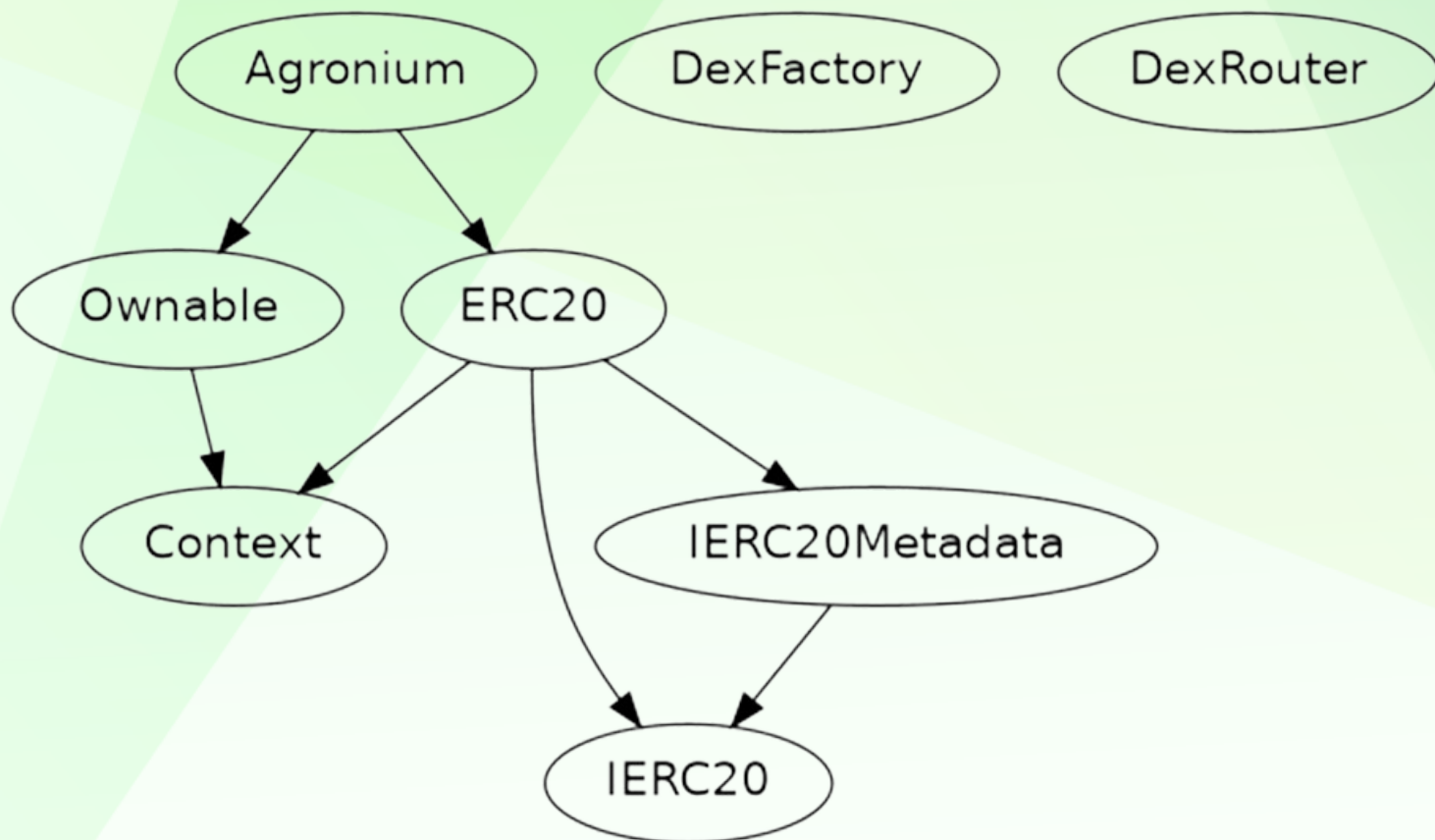
POINTS TO NOTE

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can Enable trading.
- The owner can set buy, sell, and transfer fees of not more than 5%.
- The owner can set the marketing wallet address.
- The owner can clear ERC20 tokens.

STATIC ANALYSIS

```
INFO:Detectors:
Reentrancy in Agronium.internalSwap() (Agronium.sol#476-486):
        External calls:
        - swapToETH(balanceOf(address(this))) (Agronium.sol#482)
                - uniswapRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(_amount,0,path,address(this),block.timestamp) (Agronium.sol#494-500)
        - (success) = marketingWallet.call{value: address(this).balance}() (Agronium.sol#483)
        External calls sending eth:
        - (success) = marketingWallet.call{value: address(this).balance}() (Agronium.sol#483)
        State variables written after the call(s):
        - isSwapping = false (Agronium.sol#485)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in Agronium._transfer(address,address,uint256) (Agronium.sol#447-474):
        External calls:
        - internalSwap() (Agronium.sol#470)
                - uniswapRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(_amount,0,path,address(this),block.timestamp) (Agronium.sol#494-500)
                - (success) = marketingWallet.call{value: address(this).balance}() (Agronium.sol#483)
        External calls sending eth:
        - internalSwap() (Agronium.sol#470)
                - (success) = marketingWallet.call{value: address(this).balance}() (Agronium.sol#483)
        Event emitted after the call(s):
        - Transfer(from,to,amount) (Agronium.sol#197)
                - super._transfer(_from,_to,toTransfer) (Agronium.sol#473)
Reentrancy in Agronium.clearERC20(address,address,uint256) (Agronium.sol#412-416):
        External calls:
        - IERC20(_tokenAddr).transfer(_to,_amount) (Agronium.sol#414)
        Event emitted after the call(s):
        - ClearedERC20(_tokenAddr,_to,_amount) (Agronium.sol#415)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Context._msgData() (Agronium.sol#34-36) is never used and should be removed
ERC20._burn(address,uint256) (Agronium.sol#217-233) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

```
INFO:Detectors:
Pragma version^0.8.15 (Agronium.sol#25) 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 Agronium.internalSwap() (Agronium.sol#476-486):
        - (success) = marketingWallet.call{value: address(this).balance}() (Agronium.sol#483)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function DexRouter.WETH() (Agronium.sol#280) is not in mixedCase
Event Agronium.marketingWalletChanged(address) (Agronium.sol#327) is not in CapWords
Parameter Agronium.setBuyTaxes(uint256)._marketingTax (Agronium.sol#361) is not in mixedCase
Parameter Agronium.setSellTaxes(uint256)._marketingTax (Agronium.sol#367) is not in mixedCase
Parameter Agronium.setTransferFees(uint256)._transferTaxes (Agronium.sol#373) is not in mixedCase
Parameter Agronium.setSwapTokensAtAmount(uint256)._newAmount (Agronium.sol#379) is not in mixedCase
Parameter Agronium.setWhitelistStatus(address,bool)._wallet (Agronium.sol#392) is not in mixedCase
Parameter Agronium.setWhitelistStatus(address,bool)._status (Agronium.sol#392) is not in mixedCase
Parameter Agronium.setmarketingWallet(address)._newmarketing (Agronium.sol#397) is not in mixedCase
Parameter Agronium.clearERC20(address,address,uint256)._tokenAddr (Agronium.sol#412) is not in mixedCase
Parameter Agronium.clearERC20(address,address,uint256)._to (Agronium.sol#412) is not in mixedCase
Parameter Agronium.clearERC20(address,address,uint256)._amount (Agronium.sol#412) is not in mixedCase
Parameter Agronium.checkWhitelist(address)._wallet (Agronium.sol#418) is not in mixedCase
Parameter Agronium.swapToETH(uint256)._amount (Agronium.sol#489) is not in mixedCase
Constant Agronium._totalSupply (Agronium.sol#305) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Reentrancy in Agronium.clearETH(uint256) (Agronium.sol#406-410):
        External calls:
        - address(msg.sender).transfer(weiAmount) (Agronium.sol#408)
        Event emitted after the call(s):
        - ClearedETH(weiAmount) (Agronium.sol#409)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
Agronium.slitherConstructorVariables() (Agronium.sol#302-506) uses literals with too many digits:
        - swapTokensAtAmount = _totalSupply / 100000 (Agronium.sol#314)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Slither:Agronium.sol analyzed (8 contracts with 93 detectors), 29 result(s) found
```

FUNCTIONAL TESTING

1- Approve **(passed):**

https://testnet.bscscan.com/tx/0x0cc549a1523acdc42ec2b4d7b133715eff587cc427678a76a23e669b1c0c2a3e

2- Increase Allowance **(passed):**

https://testnet.bscscan.com/tx/0xbacdc245c5f784ce06b0f724966ab34dfd8b1c4462ae3065b058946c130b55b6

3- Decrease Allowance **(passed):**

https://testnet.bscscan.com/tx/0xf00262455b29b18344575583b90c9a85f331827e797b54bb9314f88932e0c77d

4- Enable Trading **(passed):**

https://testnet.bscscan.com/tx/0x2782ec729a737992b4a673177d778aaee287043dbb75ab45f00f5a3b4841a651

5- Set Sell Taxes **(passed):**

https://testnet.bscscan.com/tx/0xe3b386ccc1d853227892143c1c74c87aaf35610ac6460c189d83deed525cbbc4

6- Set Buy Taxes **(passed):**

https://testnet.bscscan.com/tx/0x205da0f9641863ed6b98cadca73a6cc3865a108a3966005d6d63ce3128499d85

7- Set Transfer Taxes **(passed):**

https://testnet.bscscan.com/tx/0xe60849482e34d6f01440a5e2e6340e27aa0dfc8ec08f473850728bf85bbcc1dd

MANUAL TESTING

Centralization – Enabling Trades
Severity: High
function: Enabling Trades
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, "Trading is already enabled");
  tradingEnabled = true;
  startTradingBlock = 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 give investors more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad-faith actions by the original owner.

MANUAL TESTING

Centralization – Missing Require Check
Severity: Medium
Function: setmarketingWallet
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 setmarketingWallet(address _newmarketing) external
onlyOwner {
require(
    _newmarketing != address(0),
"can not set marketing to dead wallet"
   );
   marketingWallet = _newmarketing;
emit marketingWalletChanged(_newmarketing);
  }
```

Suggestion:

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

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 ^0.8.15;

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.

event InternalSwapStatusUpdated(bool indexed _status);

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