Guard

Smart contracts security assessment

final report
Tariff: Most Popular

. . .

Arable.finance

August 2021

0xguard.com

hello@0xguard.com

Contents

Introduction

The report has been prepared for Arable.finance

Name	arable.finance
Audit date	2021-08-19 - 2021-08-22
Language	Solidity
Platform	Polygon

Contracts checked

Name	Address
MasterChef	0x084BE286Dc7621225706d3803b27CBa70e3f02BD
ARABELLAToken	0x93810fe228Fa8C69B08C2D8df3Ec05357C00C625

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) all the issues found by the tools

Manual audit

- Manually analyse smart contracts for security vulnerabilities
- Smart contracts' logic check

Privileged roles

ARABELLAToken

- mint tokens. The ownership of the token was transferred to the MasterChef contract. Only the MasterChef can mint tokens.

MasterChef

- Update emission rate of the token
- Update start block
- Add pool
- Set pool parameters

Known vulnerabilities checked

Title	Check result
Unencrypted Private Data On-Chain	✓ passed
Code With No Effects	✓ passed
Message call with hardcoded gas amount	✓ passed
Typographical Error	✓ passed
DoS With Block Gas Limit	✓ passed
Presence of unused variables	✓ passed
Incorrect Inheritance Order	✓ passed
Requirement Violation	✓ passed
Weak Sources of Randomness from Chain Attributes	✓ passed
Shadowing State Variables	✓ passed

Title	Check result
Incorrect Constructor Name	✓ passed
Block values as a proxy for time	✓ passed
Authorization through tx.origin	✓ passed
DoS with Failed Call	✓ passed
Delegatecall to Untrusted Callee	✓ passed
Use of Deprecated Solidity Functions	✓ passed
Assert Violation	✓ passed
State Variable Default Visibility	✓ passed
Reentrancy	✓ passed
Unprotected SELFDESTRUCT Instruction	✓ passed
Unprotected Ether Withdrawal	✓ passed
Unchecked Call Return Value	✓ passed
Floating Pragma	✓ passed
Outdated Compiler Version	✓ passed
Integer Overflow and Underflow	⊘ not passed
Function Default Visibility	✓ passed

Classification of issues

High severity	Bugs leading to assets theft, locking or any other loss of assets or leading to contract malfunctioning.
Medium severity	Bugs that can trigger a contract failure of malfunctioning.
Low severity	Bugs that do now affect contract functionality. For example, unoptimised gas usage, outdated or unused code, code style violations, etc.

Issues

High severity issues

1. Broken governance mechanism (ARABELLAToken)

The votes in the governance mechanism of the ARABELLAToken can be double spended (see sushi votes attack).

Medium severity issues

1. Possible integer overflow (MasterChef)

Safemath is not used in L1455. If token mints on transfer, the _amount variable may underverflow. Owner of the contract should carefully check the tokens added as pools.

2. MassupdatePools() and updateStartBlock() may run out of block gas limit

Functions massUpdatePools() and updateStartBlock() may run out of block gas limit if a big number of pools is added as they iterate over an unlimited number of pools.

```
function massUpdatePools() public {
    uint256 length = poolInfo.length;
    for (uint256 pid = 0; pid < length; ++pid) {
        updatePool(pid);
    }
}
```

Recommendation: owner of the MasterChef should

3. Emission rate not limited

The owner of the MasterChef can set an arbitrary big value for emission rate. The emission rate must be capped.

```
function updateEmissionRate(uint256 _ARABELLAPerBlock) external onlyOwner {
    massUpdatePools();
    ARABELLAPerBlock = _ARABELLAPerBlock;
    emit UpdateEmissionRate(msg.sender, _ARABELLAPerBlock);
}
```

Recommendation: cap the emission rate.

4. Hidden pool owner's attack

The owner of MasterChef can create a pool with zero allocation and then by setting allocation to a big value may mint significant number of tokens (see Dracula protocol article).

Recommendation: force update pools in the set() function.

Low severity issues

1. Outdated compiler version

Outdated compiler version is used (v0.6.12+commit.27d51765). We recommend using the latest stable version of the Solidity compiler.

Conclusion

Arable.finance MasterChef and ARABELLAToken contracts were audited. 1 high and 4 medium severity issues were found.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability)set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without 0xGuard prior written consent.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts 0xGuard to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. 0xGuard’s position is that each company and individual are responsible for their own due diligence and continuous security. 0xGuard’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Static code analysis result

Number of lines: 1553 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 10 (+ 0 in dependencies, + 0 tests)
Number of optimization issues: 13
Number of informational issues: 61
Number of low issues: 14
Number of medium issues: 13
Number of high issues: 0

ERCs: ERC20

Name	#functions	ERCS	ERC20 info	Complex code	Features
SafeMath	13			⊘	
Address	11			⊘	Send ETH
					Delegatecall
					Assembly
ARABELLAToken	47	ERC20	∞ Minting	✓	Ecrecover
			Approve Race Cond.		Assembly
SafeBEP20	6			⊘	Send ETH
					Tokens interaction
MasterChef	25			⊘	Tokens interaction

INFO:Detectors:
MasterChef.pendingARABELLA(uint256,address) (contracts/Greeter.sol#1403-1413) performs a multiplication on the result of a division:
-ARABELLAReward = multiplier.mul(ARABELLAPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (contracts/Greeter.sol#1409)
-accARABELLAPerShare = accARABELLAPerShare.add(ARABELLAReward.mul(1e18).div(pool.lpSupply)) (contracts/Greeter.sol#1410)
MasterChef.updatePool(uint256) (contracts/Greeter.sol#1424-1439) performs a multiplication on the result of a division:
-ARABELLAReward = multiplier.mul(ARABELLAPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (contracts/Greeter.sol#1434)
-pool.accARABELLAPerShare = pool.accARABELLAPerShare.add(ARABELLAReward.mul(1e18).div(pool.lpSupply)) (contracts/Greeter.sol#1437)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
ARABELLAToken._writeCheckpoint(address,uint32,uint256,uint256) (contracts/Greeter.sol#1140-1158) uses a dangerous strict equality:
- nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (contracts/Greeter.sol#1150)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in MasterChef.add(uint256,IBEP20,uint16,bool) (contracts/Greeter.sol#1361-1382):
External calls:
- massUpdatePools() (contracts/Greeter.sol#1367)
- ARABELLA.mint(address(this),ARABELLAReward) (contracts/Greeter.sol#1436)
State variables written after the call(s):
- poolExistence[_lpToken] = true (contracts/Greeter.sol#1371)
- poolInfo.push(PoolInfo(_lpToken,_allocPoint,lastRewardBlock,0,_depositFeeBP,0)) (contracts/Greeter.sol#1372-1379)
- totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/Greeter.sol#1370)
Reentrancy in MasterChef.deposit(uint256,uint256) (contracts/Greeter.sol#1442-1468):
External calls:
- updatePool(_pid) (contracts/Greeter.sol#1445)
- ARABELLA.mint(address(this),ARABELLAReward) (contracts/Greeter.sol#1436)
- safeARABELLATransfer(msg.sender,pending) (contracts/Greeter.sol#1449)
- transferSuccess = ARABELLA.transfer(_to,ARABELLABal) (contracts/Greeter.sol#1512)

- transferSuccess = ARABELLA.transfer(_to,_amount) (contracts/Greeter.sol#1514)
- pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (contracts/Greeter.sol#1454)
- pool.lpToken.safeTransfer(feeAddress,depositFee) (contracts/Greeter.sol#1458)

State variables written after the call(s):
- pool.lpSupply = pool.lpSupply.add(_amount).sub(depositFee) (contracts/Greeter.sol#1460)
- user.amount = user.amount.add(_amount).sub(depositFee) (contracts/Greeter.sol#1459)

Reentrancy in MasterChef.deposit(uint256,uint256) (contracts/Greeter.sol#1442-1468):

External calls:
- updatePool(_pid) (contracts/Greeter.sol#1445)
  - ARABELLA.mint(address(this),ARABELLAReward) (contracts/Greeter.sol#1436)
- safeARABELLATransfer(msg.sender,pending) (contracts/Greeter.sol#1449)
  - transferSuccess = ARABELLA.transfer(_to,ARABELLABal) (contracts/Greeter.sol#1512)
  - transferSuccess = ARABELLA.transfer(_to,_amount) (contracts/Greeter.sol#1514)
- pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (contracts/Greeter.sol#1454)

State variables written after the call(s):
- pool.lpSupply = pool.lpSupply.add(_amount) (contracts/Greeter.sol#1463)
- user.amount = user.amount.add(_amount) (contracts/Greeter.sol#1462)

Reentrancy in MasterChef.emergencyWithdraw(uint256) (contracts/Greeter.sol#1490-1505):

External calls:
- pool.lpToken.safeTransfer(address(msg.sender),amount) (contracts/Greeter.sol#1496)

State variables written after the call(s):
- pool.lpSupply = pool.lpSupply.sub(amount) (contracts/Greeter.sol#1499)
- pool.lpSupply = 0 (contracts/Greeter.sol#1501)

Reentrancy in MasterChef.set(uint256,uint256,uint16,bool) (contracts/Greeter.sol#1385-1395):

External calls:
- massUpdatePools() (contracts/Greeter.sol#1388)
  - ARABELLA.mint(address(this),ARABELLAReward) (contracts/Greeter.sol#1436)

State variables written after the call(s):
- poolInfo[_pid].allocPoint = _allocPoint (contracts/Greeter.sol#1391)
- poolInfo[_pid].depositFeeBP = _depositFeeBP (contracts/Greeter.sol#1392)
- totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint) (contracts/Greeter.sol#1390)

Reentrancy in MasterChef.updateEmissionRate(uint256) (contracts/Greeter.sol#1535-1539):

External calls:
- massUpdatePools() (contracts/Greeter.sol#1536)
  - ARABELLA.mint(address(this),ARABELLAReward) (contracts/Greeter.sol#1436)

State variables written after the call(s):
- ARABELLAPerBlock = _ARABELLAPerBlock (contracts/Greeter.sol#1537)

Reentrancy in MasterChef.updatePool(uint256) (contracts/Greeter.sol#1424-1439):
External calls:
- ARABELLA.mint(address(this),ARABELLAReward) (contracts/Greeter.sol#1436)
State variables written after the call(s):
- pool.accARABELLAPerShare = pool.accARABELLAPerShare.add(ARABELLAReward.mul(1e18).div(pool.lpSupply)) (contracts/Greeter.sol#1437)
- pool.lastRewardBlock = block.number (contracts/Greeter.sol#1438)
Reentrancy in MasterChef.withdraw(uint256,uint256) (contracts/Greeter.sol#1471-1487):
External calls:
- updatePool(_pid) (contracts/Greeter.sol#1475)
- ARABELLA.mint(address(this),ARABELLAReward) (contracts/Greeter.sol#1436)
- safeARABELLATransfer(msg.sender,pending) (contracts/Greeter.sol#1478)
- transferSuccess = ARABELLA.transfer(_to,ARABELLABal) (contracts/Greeter.sol#1512)
- transferSuccess = ARABELLA.transfer(_to,_amount) (contracts/Greeter.sol#1514)
State variables written after the call(s):
- user.amount = user.amount.sub(_amount) (contracts/Greeter.sol#1481)
Reentrancy in MasterChef.withdraw(uint256,uint256) (contracts/Greeter.sol#1471-1487):
External calls:
- updatePool(_pid) (contracts/Greeter.sol#1475)
- ARABELLA.mint(address(this),ARABELLAReward) (contracts/Greeter.sol#1436)
- safeARABELLATransfer(msg.sender,pending) (contracts/Greeter.sol#1478)
- transferSuccess = ARABELLA.transfer(_to,ARABELLABal) (contracts/Greeter.sol#1512)
- transferSuccess = ARABELLA.transfer(_to,_amount) (contracts/Greeter.sol#1514)
- pool.lpToken.safeTransfer(address(msg.sender),_amount) (contracts/Greeter.sol#1482)
State variables written after the call(s):
- pool.lpSupply = pool.lpSupply.sub(_amount) (contracts/Greeter.sol#1483)
- user.rewardDebt = user.amount.mul(pool.accARABELLAPerShare).div(1e18) (contracts/Greeter.sol#1485)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
MasterChef.add(uint256,IBEP20,uint16,bool) (contracts/Greeter.sol#1361-1382) ignores return value by _lpToken.balanceOf(address(this)) (contracts/Greeter.sol#1363)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
BEP20.constructor(string,string).name (contracts/Greeter.sol#670) shadows:
- BEP20.name() (contracts/Greeter.sol#686-688) (function)
- IBEP20.name() (contracts/Greeter.sol#570) (function)

BEP20.constructor(string,string).symbol (contracts/Greeter.sol#670) shadows:
- BEP20.symbol() (contracts/Greeter.sol#700-702) (function)
- IBEP20.symbol() (contracts/Greeter.sol#565) (function)
BEP20.allowance(address,address).owner (contracts/Greeter.sol#734) shadows:
- Ownable.owner() (contracts/Greeter.sol#329-331) (function)
BEP20._approve(address,address,uint256).owner (contracts/Greeter.sol#906) shadows:
- Ownable.owner() (contracts/Greeter.sol#329-331) (function)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
MasterChef.constructor(ARABELLAToken,address,address,uint256,uint256)._devaddr (contracts/Greeter.sol#1338) lacks a zero-check on :
- devaddr = _devaddr (contracts/Greeter.sol#1344)
MasterChef.constructor(ARABELLAToken,address,address,uint256,uint256)._feeAddress (contracts/Greeter.sol#1339) lacks a zero-check on :
- feeAddress = _feeAddress (contracts/Greeter.sol#1345)
MasterChef.setDevAddress(address)._devaddr (contracts/Greeter.sol#1520) lacks a zero-check on :
- devaddr = _devaddr (contracts/Greeter.sol#1522)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in MasterChef.add(uint256,IBEP20,uint16,bool) (contracts/Greeter.sol#1361-1382):
External calls:
- massUpdatePools() (contracts/Greeter.sol#1367)
- ARABELLA.mint(address(this),ARABELLAReward) (contracts/Greeter.sol#1436)
Event emitted after the call(s):
- addPool(poolInfo.length - 1,address(_lpToken),_allocPoint,_depositFeeBP) (contracts/Greeter.sol#1381)
Reentrancy in MasterChef.deposit(uint256,uint256) (contracts/Greeter.sol#1442-1468):
External calls:
- updatePool(_pid) (contracts/Greeter.sol#1445)
- ARABELLA.mint(address(this),ARABELLAReward) (contracts/Greeter.sol#1436)
- safeARABELLATransfer(msg.sender,pending) (contracts/Greeter.sol#1449)
- transferSuccess = ARABELLA.transfer(_to,ARABELLABal) (contracts/Greeter.sol#1512)
- transferSuccess = ARABELLA.transfer(_to,_amount) (contracts/Greeter.sol#1514)
- pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (contracts/Greeter.sol#1454)
- pool.lpToken.safeTransfer(feeAddress,depositFee) (contracts/Greeter.sol#1458)
Event emitted after the call(s):
- Deposit(msg.sender,_pid,_amount) (contracts/Greeter.sol#1467)

Reentrancy in MasterChef.emergencyWithdraw(uint256) (contracts/Greeter.sol#1490-1505):
External calls:
- pool.lpToken.safeTransfer(address(msg.sender),amount) (contracts/Greeter.sol#1496)
Event emitted after the call(s):
- EmergencyWithdraw(msg.sender,_pid,amount) (contracts/Greeter.sol#1504)
Reentrancy in MasterChef.set(uint256,uint256,uint16,bool) (contracts/Greeter.sol#1385-1395):
External calls:
- massUpdatePools() (contracts/Greeter.sol#1388)
- ARABELLA.mint(address(this),ARABELLAReward) (contracts/Greeter.sol#1436)
Event emitted after the call(s):
- setPool(_pid,address(poolInfo[_pid].lpToken),_allocPoint,_depositFeeBP) (contracts/Greeter.sol#1394)
Reentrancy in MasterChef.updateEmissionRate(uint256) (contracts/Greeter.sol#1535-1539):
External calls:
- massUpdatePools() (contracts/Greeter.sol#1536)
- ARABELLA.mint(address(this),ARABELLAReward) (contracts/Greeter.sol#1436)
Event emitted after the call(s):
- UpdateEmissionRate(msg.sender,_ARABELLAPerBlock) (contracts/Greeter.sol#1538)
Reentrancy in MasterChef.withdraw(uint256,uint256) (contracts/Greeter.sol#1471-1487):
External calls:
- updatePool(_pid) (contracts/Greeter.sol#1475)
- ARABELLA.mint(address(this),ARABELLAReward) (contracts/Greeter.sol#1436)
- safeARABELLATransfer(msg.sender,pending) (contracts/Greeter.sol#1478)
- transferSuccess = ARABELLA.transfer(_to,ARABELLABal) (contracts/Greeter.sol#1512)
- transferSuccess = ARABELLA.transfer(_to,_amount) (contracts/Greeter.sol#1514)
- pool.lpToken.safeTransfer(address(msg.sender),_amount) (contracts/Greeter.sol#1482)
Event emitted after the call(s):
- Withdraw(msg.sender,_pid,_amount) (contracts/Greeter.sol#1486)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
ARABELLAToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (contracts/Greeter.sol#1006-1047) uses timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(now <= expiry,ARABELLA::delegateBySig: signature expired) (contracts/Greeter.sol#1045)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (contracts/Greeter.sol#385-394) uses assembly
- INLINE ASM (contracts/Greeter.sol#392)
Address._verifyCallResult(bool,bytes,string) (contracts/Greeter.sol#530-547) uses assembly
- INLINE ASM (contracts/Greeter.sol#539-542)

ARABELLAToken.getChainId() (contracts/Greeter.sol#1165-1169) uses assembly
- INLINE ASM (contracts/Greeter.sol#1167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
MasterChef.nonDuplicated(IBEP20) (contracts/Greeter.sol#1355-1358) compares to a boolean constant:
-require(bool,string)(poolExistence[_lpToken] == false,nonDuplicated: duplicated) (contracts/Greeter.sol#1356)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Address.functionCall(address,bytes) (contracts/Greeter.sol#438-440) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (contracts/Greeter.sol#463-465) is never used and should be removed
Address.functionDelegateCall(address,bytes) (contracts/Greeter.sol#512-514) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (contracts/Greeter.sol#522-528) is never used and should be removed
Address.functionStaticCall(address,bytes) (contracts/Greeter.sol#488-490) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (contracts/Greeter.sol#498-504) is never used and should be removed
Address.sendValue(address,uint256) (contracts/Greeter.sol#412-418) is never used and should be removed
BEP20._burn(address,uint256) (contracts/Greeter.sol#884-890) is never used and should be removed
BEP20._burnFrom(address,uint256) (contracts/Greeter.sol#923-930) is never used and should be removed
Context._msgData() (contracts/Greeter.sol#83-86) is never used and should be removed
SafeBEP20.safeApprove(IBEP20,address,uint256) (contracts/Greeter.sol#1210-1224) is never used and should be removed
SafeBEP20.safeDecreaseAllowance(IBEP20,address,uint256) (contracts/Greeter.sol#1235-1245) is never used and should be removed
SafeBEP20.safeIncreaseAllowance(IBEP20,address,uint256) (contracts/Greeter.sol#1226-1233) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/Greeter.sol#274-277) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/Greeter.sol#236-239) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/Greeter.sol#294-297) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (contracts/Greeter.sol#108-112) is never used and should be removed

SafeMath.tryDiv(uint256,uint256) (contracts/Greeter.sol#144-147) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (contracts/Greeter.sol#154-157) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (contracts/Greeter.sol#129-137) is never used and should be removed
SafeMath.trySub(uint256,uint256) (contracts/Greeter.sol#119-122) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (contracts/Greeter.sol#412-418):
- (success) = recipient.call{value: amount}() (contracts/Greeter.sol#416)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (contracts/Greeter.sol#473-480):
- (success,returndata) = target.call{value: value}(data) (contracts/Greeter.sol#478)
Low level call in Address.functionStaticCall(address,bytes,string) (contracts/Greeter.sol#498-504):
- (success,returndata) = target.staticcall(data) (contracts/Greeter.sol#502)
Low level call in Address.functionDelegateCall(address,bytes,string) (contracts/Greeter.sol#522-528):
- (success,returndata) = target.delegatecall(data) (contracts/Greeter.sol#526)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter ARABELLAToken.mint(address,uint256)._to (contracts/Greeter.sol#936) is not in mixedCase
Parameter ARABELLAToken.mint(address,uint256)._amount (contracts/Greeter.sol#936) is not in mixedCase
Variable ARABELLAToken._delegates (contracts/Greeter.sol#948) is not in mixedCase
Event MasterChefaddPool(uint256,address,uint256,uint256) (contracts/Greeter.sol#1332) is not in CapWords
Event MasterChefsetPool(uint256,address,uint256,uint256) (contracts/Greeter.sol#1333) is not in CapWords
Parameter MasterChef.add(uint256,IBEP20,uint16,bool)._allocPoint (contracts/Greeter.sol#1361) is not in mixedCase
Parameter MasterChef.add(uint256,IBEP20,uint16,bool)._lpToken (contracts/Greeter.sol#1361) is not in mixedCase
Parameter MasterChef.add(uint256,IBEP20,uint16,bool)._depositFeeBP (contracts/Greeter.sol#1361) is not in mixedCase
Parameter MasterChef.add(uint256,IBEP20,uint16,bool)._withUpdate (contracts/Greeter.sol#1361) is not in mixedCase
Parameter MasterChef.set(uint256,uint256,uint16,bool)._pid (contracts/Greeter.sol#1385) is not in mixedCase

```
INFO:Detectors:
Redundant expression "this (contracts/Greeter.sol#84)" inContext (contracts/Greeter.sol#78-87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
	- Ownable.renounceOwnership() (contracts/Greeter.sol#348-351)
transferOwnership(address) should be declared external:
	- Ownable.transferOwnership(address) (contracts/Greeter.sol#357-361)
decimals() should be declared external:
	- BEP20.decimals() (contracts/Greeter.sol#693-695)
symbol() should be declared external:
	- BEP20.symbol() (contracts/Greeter.sol#700-702)
totalSupply() should be declared external:
	- BEP20.totalSupply() (contracts/Greeter.sol#707-709)
transfer(address,uint256) should be declared external:
	- BEP20.transfer(address,uint256) (contracts/Greeter.sol#726-729)
allowance(address,address) should be declared external:
	- BEP20.allowance(address,address) (contracts/Greeter.sol#734-736)
approve(address,uint256) should be declared external:
	- BEP20.approve(address,uint256) (contracts/Greeter.sol#745-748)
transferFrom(address,address,uint256) should be declared external:
	- BEP20.transferFrom(address,address,uint256) (contracts/Greeter.sol#762-774)
increaseAllowance(address,uint256) should be declared external:
	- BEP20.increaseAllowance(address,uint256) (contracts/Greeter.sol#788-791)
decreaseAllowance(address,uint256) should be declared external:
	- BEP20.decreaseAllowance(address,uint256) (contracts/Greeter.sol#807-814)
mint(uint256) should be declared external:
	- BEP20.mint(uint256) (contracts/Greeter.sol#824-827)
mint(address,uint256) should be declared external:
	- ARABELLAToken.mint(address,uint256) (contracts/Greeter.sol#936-939)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:. analyzed (10 contracts with 75 detectors), 101 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Guard