0x Guard

Smart contracts security assessment

Final report

Tariff: Standard

BankaiCoin

December 2021

0xguard.com

hello@0xguard.com

Contents

Introduction

The report has been prepared for BankaiCoin team.

The code is published to Github. The audit was done after commit 8f52e68. The recheck was done after the commit ea59cdf

Users should check if they interact with the same contracts as were audited.

Name	BankaiCoin
Audit date	2021-12-09 - 2021-12-11
Language	Solidity
Platform	Ethereum

Contracts checked

Name	Address
BankaiCoin	
PreIcoEscrow	
BankaiCoinCrowdsale	
BankaiCoinAirdrop	

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

Known vulnerabilities checked

Title	Check result
Unencrypted Private Data On-Chain	passed
Code With No Effects	not passed
Message call with hardcoded gas amount	passed
Typographical Error	passed
DoS With Block Gas Limit	passed
Presence of unused variables	not passed
Incorrect Inheritance Order	passed
Requirement Violation	passed
Weak Sources of Randomness from Chain Attributes	passed
Shadowing State Variables	passed
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	not passed
Outdated Compiler Version	not passed
Integer Overflow and Underflow	passed
Function Default Visibility	passed

Classification of issue severity

High severity	High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.
Medium severity	Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.
Low severity	Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

1. Mint is not restricted (BankaiCoin)

If the owner account gets compromised an unlimited number of coins can be minted.

Recommendation: We recommend removing mint() function or renouncing ownership after deployment.

Update: The issue was fixed. The mint() function was removed.

Medium severity issues

1. Bypass checks in buyTokensFromEscrow (PreIcoEscrow)

A user can bypass checks in buyTokensFromEscrow function by direct buyTokens() call.

Update: The issue was fixed. The buyTokensFromEscrow function has been commented out.

2. Bypass checks in buyTokensFromCrowdsale (BankaiCoinCrowdsale)

A user can bypass checks in buyTokensFromCrowdsalefunction by direct buyTokens() call.

Update: The issue was fixed. The buyTokensFromCrowdsale function has been commented out.

3. BKAI value is checked on ETH value (BankaiCoinCrowdsale)

Function buyTokensFromCrowdsale have require there minimumAmountOfBKAI is checked on ETH value of msg.value.

Update: The issue was fixed. The buyTokensFromCrowdsale function has been commented out.

4. Not full realization of ERC20 standard (BankaiCoinAirdrop)

BankaiCoinAirdrop implements functions from ERC20 token standard but lacks some regarding metadata (name(), symbol(), decimals()).

Update: The issue was fixed. Now BankaiCoinAirdrop is not an ERC20 token standard.

Low severity issues

1. Pausable is not used (BankaiCoin)

Contract BankaiCoin is Pausable, but Pausable is not used.

Recommendation: We recommend removing unused libraries and contracts.

Update: The issue was fixed. A whenNotPaused modifier has been added to the setCrowdsaleAddress function.

2. Emit events on importart value changes (BankaiCoin)

We recommend add emit events on importart value changes, such as mint, chainges bankaiCoinCrowdsaleAddress and bankaiCoinEscrowAddress, transfers.

Update: The issue was fixed. Emit events have been added.

3. External and public function types (BankaiCoin)

Some functions can be declared external instead of public. This will save gas on calling them.

- mint

- setCrowdsaleAddress

- startPreIco

- startIco

Recommendation: Make these functions external.

Update: The issue was fixed. The functions have been declared external.

4. External and public function types (PreIcoEscrow)

Some functions can be declared external instead of public. This will save gas on calling them.

- buyTokensFromEscrow

- stopPreIco

Update: The issue was fixed. The functions have been declared external.

5. Miniumum BKAI value is checked on ETH value (PreIcoEscrow)

Function buyTokensFromEscrow() have require there minumumAmountOfBKAI is checked on ETH value of msg.value.

Update: The issue was fixed. The buyTokensFromEscrow function has been commented out.

6. Unused local variable bancaiCoin (PreIcoEscrow)

The bancaiCoin variable has been created, but is not used.

7. safeTransfer instead transfer (PreIcoEscrow)

The result of the token transfer() function is not checked.

```
    function stopPreIco(IERC20 token, address to) public {
        ...
        token.transfer(to, erc20balance);
        ...
    }
```

Recommendation: We recommend transferring tokens with safeTransfer() function from SafeERC20 library instead of transfer().

Update: The issue was fixed. The transfer() function has been replaced by safe Transfer().

8. safeTransfer instead transfer (BankaiCoinCrowdsale)

We recommend token safeTransfer() should be used instead of transfer() because transfer() result not checked.

Update: The issue was fixed. The transfer() function has been replaced by safe Transfer().

9. External and public function types (BankaiCoinCrowdsale)

Some functions can be declared external instead of public. This will save gas on calling them.

- buyTokensFromCrowdsale

- stopIco

Update: The buyTokensFromCrowdsale function has been commented out.

10. Unused local variable bancaiCoin (BankaiCoinCrowdsale)

The bancaiCoin variable has been created, but is not used.

11. External and public function types (BankaiCoinAirdrop)

Some functions can be declared external instead of public. This will save gas on calling them.

- newAirdropTransfer()

Update: The issue was fixed. The newAirdropTransfer() function has been removed.

Conclusion

BankaiCoin BankaiCoin, PreIcoEscrow, BankaiCoinCrowdsale, BankaiCoinAirdrop contracts were audited. 1 high, 4 medium, 11 low severity issues were found.Users should check if they interact with the same contracts as were audited.

Update. All high and medium severity issues were fixed. 13 of 16 issues were fixed total.

Disclaimer

Slither check output

```
INFO:Detectors:
PreIcoEscrow.stopPreIco(IERC20,address) (PreIcoEscrow.sol#30-36) ignores return value
by token.transfer(to,erc20balance) (PreIcoEscrow.sol#34)
BankaiCoinCrowdsale.stopIco(IERC20,address) (BankaiCoinCrowdsale.sol#34-40) ignores
return value by token.transfer(to,erc20balance) (BankaiCoinCrowdsale.sol#38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-
transfer
INFO:Detectors:
BankaiCoin.bankaiCoinCrowdsale (BankaiCoin.sol#16) is never initialized. It is used in:
        - BankaiCoin.slitherConstructorVariables() (BankaiCoin.sol#14-61)
BankaiCoin.preIcoEscrow (BankaiCoin.sol#19) is never initialized. It is used in:
        - BankaiCoin.slitherConstructorVariables() (BankaiCoin.sol#14-61)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-
state-variables
INFO:Detectors:
BankaiCoin.constructor(string,string,uint256).name (BankaiCoin.sol#30) shadows:
        - ERC20Detailed.name() (@openzeppelin/contracts/token/ERC20/
ERC20Detailed.sol#27-29) (function)
BankaiCoin.constructor(string,string,uint256).symbol (BankaiCoin.sol#31) shadows:
        - ERC20Detailed.symbol() (@openzeppelin/contracts/token/ERC20/
ERC20Detailed.sol#35-37) (function)
Crowdsale.constructor(uint256,address,IERC20).rate (@openzeppelin/contracts/crowdsale/
Crowdsale.sol#57) shadows:
        - Crowdsale.rate() (@openzeppelin/contracts/crowdsale/Crowdsale.sol#94-96)
(function)
Crowdsale.constructor(uint256,address,IERC20).wallet (@openzeppelin/contracts/crowdsale/
Crowdsale.sol#57) shadows:
        - Crowdsale.wallet() (@openzeppelin/contracts/crowdsale/Crowdsale.sol#87-89)
(function)
Crowdsale.constructor(uint256,address,IERC20).token (@openzeppelin/contracts/crowdsale/
Crowdsale.sol#57) shadows:
        - Crowdsale.token() (@openzeppelin/contracts/crowdsale/Crowdsale.sol#80-82)
(function)
PreIcoEscrow.constructor(uint256,address,IERC20).rate (PreIcoEscrow.sol#13) shadows:
        - Crowdsale.rate() (@openzeppelin/contracts/crowdsale/Crowdsale.sol#94-96)
(function)
PreIcoEscrow.constructor(uint256,address,IERC20).wallet (PreIcoEscrow.sol#14) shadows:
        - Crowdsale.wallet() (@openzeppelin/contracts/crowdsale/Crowdsale.sol#87-89)
(function)
PreIcoEscrow.constructor(uint256,address,IERC20).token (PreIcoEscrow.sol#15) shadows:
        - Crowdsale.token() (@openzeppelin/contracts/crowdsale/Crowdsale.sol#80-82)
(function)
PreIcoEscrow.stopPreIco(IERC20,address).token (PreIcoEscrow.sol#30) shadows:
        - Crowdsale.token() (@openzeppelin/contracts/crowdsale/Crowdsale.sol#80-82)
(function)
BankaiCoinCrowdsale.constructor(uint256,address,IERC20).rate
(BankaiCoinCrowdsale.sol#15) shadows:
        - Crowdsale.rate() (@openzeppelin/contracts/crowdsale/Crowdsale.sol#94-96)
(function)
BankaiCoinCrowdsale.constructor(uint256,address,IERC20).wallet
(BankaiCoinCrowdsale.sol#16) shadows:
        - Crowdsale.wallet() (@openzeppelin/contracts/crowdsale/Crowdsale.sol#87-89)
(function)
BankaiCoinCrowdsale.constructor(uint256,address,IERC20).token
(BankaiCoinCrowdsale.sol#17) shadows:
        - Crowdsale.token() (@openzeppelin/contracts/crowdsale/Crowdsale.sol#80-82)
(function)
BankaiCoinCrowdsale.stopIco(IERC20,address).token (BankaiCoinCrowdsale.sol#34) shadows:
        - Crowdsale.token() (@openzeppelin/contracts/crowdsale/Crowdsale.sol#80-82)
(function)
ERC20Detailed.constructor(string,string,uint8).name (@openzeppelin/contracts/token/
ERC20/ERC20Detailed.sol#18) shadows:
        - ERC20Detailed.name() (@openzeppelin/contracts/token/ERC20/
ERC20Detailed.sol#27-29) (function)
ERC20Detailed.constructor(string,string,uint8).symbol (@openzeppelin/contracts/token/
ERC20/ERC20Detailed.sol#18) shadows:
        - ERC20Detailed.symbol() (@openzeppelin/contracts/token/ERC20/
ERC20Detailed.sol#35-37) (function)
ERC20Detailed.constructor(string,string,uint8).decimals (@openzeppelin/contracts/token/
ERC20/ERC20Detailed.sol#18) shadows:
        - ERC20Detailed.decimals() (@openzeppelin/contracts/token/ERC20/
ERC20Detailed.sol#51-53) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-
shadowing
INFO:Detectors:
BankaiCoin.setCrowdsaleAddress(address,address)._crowdsaleAddress (BankaiCoin.sol#43)
lacks a zero-check on :
                - bankaiCoinCrowdsaleAddress = _crowdsaleAddress (BankaiCoin.sol#44)
```

```
BankaiCoin.setCrowdsaleAddress(address,address)._escrowAddress (BankaiCoin.sol#43)
lacks a zero-check on :
                - bankaiCoinEscrowAddress = _escrowAddress (BankaiCoin.sol#45)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-
address-validation
INFO:Detectors:
Reentrancy in Crowdsale.buyTokens(address) (@openzeppelin/contracts/crowdsale/
Crowdsale.sol#111-128):
        External calls:
        - _processPurchase(beneficiary,tokens) (@openzeppelin/contracts/crowdsale/
Crowdsale.sol#121)
                - _token.safeTransfer(beneficiary,tokenAmount) (@openzeppelin/contracts/
crowdsale/Crowdsale.sol#162)
                - (success,returndata) = address(token).call(data) (@openzeppelin/
contracts/token/ERC20/SafeERC20.sol#67)
        Event emitted after the call(s):
        - TokensPurchased(_msgSender(),beneficiary,weiAmount,tokens) (@openzeppelin/
contracts/crowdsale/Crowdsale.sol#122)
Reentrancy in BankaiCoinCrowdsale.stopIco(IERC20,address)
(BankaiCoinCrowdsale.sol#34-40):
        External calls:
        - token.transfer(to,erc20balance) (BankaiCoinCrowdsale.sol#38)
        Event emitted after the call(s):
        - TransferSent(msg.sender,to,erc20balance) (BankaiCoinCrowdsale.sol#39)
Reentrancy in PreIcoEscrow.stopPreIco(IERC20,address) (PreIcoEscrow.sol#30-36):
        External calls:
        - token.transfer(to,erc20balance) (PreIcoEscrow.sol#34)
        Event emitted after the call(s):
        - TransferSent(msg.sender,to,erc20balance) (PreIcoEscrow.sol#35)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3
INFO:Detectors:
Address.isContract(address) (@openzeppelin/contracts/utils/Address.sol#24-33) uses
assembly
        - INLINE ASM (@openzeppelin/contracts/utils/Address.sol#31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['>=0.4.22<0.9.0', '^0.5.0', '^0.5.5']
        - ^0.5.0 (@openzeppelin/contracts/access/Roles.sol#1)
        - ^0.5.0 (@openzeppelin/contracts/token/ERC20/ERC20Burnable.sol#1)
        - ^0.5.0 (@openzeppelin/contracts/math/SafeMath.sol#1)
        - ^0.5.0 (@openzeppelin/contracts/access/roles/PauserRole.sol#1)
        - ^0.5.0 (BankaiCoinAirdrop.sol#2)
        - ^0.5.0 (@openzeppelin/contracts/token/ERC20/SafeERC20.sol#1)
        - ^0.5.5 (@openzeppelin/contracts/utils/Address.sol#1)
        - ^0.5.0 (BankaiCoin.sol#2)
        - ^0.5.0 (@openzeppelin/contracts/utils/ReentrancyGuard.sol#1)
        - ^0.5.0 (@openzeppelin/contracts/GSN/Context.sol#1)
        - ^0.5.0 (@openzeppelin/contracts/crowdsale/Crowdsale.sol#1)
        - ^0.5.0 (@openzeppelin/contracts/lifecycle/Pausable.sol#1)
        - ^0.5.0 (PreIcoEscrow.sol#1)
        - ^0.5.0 (BankaiCoinCrowdsale.sol#2)
        - ^0.5.0 (@openzeppelin/contracts/token/ERC20/ERC20Detailed.sol#1)
        - >=0.4.22<0.9.0 (Migrations.sol#2)
        - ^0.5.0 (@openzeppelin/contracts/ownership/Ownable.sol#1)
        - ^0.5.0 (@openzeppelin/contracts/token/ERC20/ERC20.sol#1)
        - ^0.5.0 (@openzeppelin/contracts/token/ERC20/IERC20.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-
pragma-directives-are-used
INFO:Detectors:
BankaiCoin.bankaiCoinCrowdsaleAddress (BankaiCoin.sol#17) is set pre-construction with
a non-constant function or state variable:
        - address(bankaiCoinCrowdsale)
BankaiCoin.bankaiCoinEscrowAddress (BankaiCoin.sol#20) is set pre-construction with a
non-constant function or state variable:
        - address(preIcoEscrow)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-
initializing-state-variables
INFO:Detectors:
Pragma version^0.5.0 (@openzeppelin/contracts/access/Roles.sol#1) allows old versions
Pragma version^0.5.0 (@openzeppelin/contracts/token/ERC20/ERC20Burnable.sol#1) allows
old versions
Pragma version^0.5.0 (@openzeppelin/contracts/math/SafeMath.sol#1) allows old versions
Pragma version^0.5.0 (@openzeppelin/contracts/access/roles/PauserRole.sol#1) allows old
versions
Pragma version^0.5.0 (BankaiCoinAirdrop.sol#2) allows old versions
Pragma version^0.5.0 (@openzeppelin/contracts/token/ERC20/SafeERC20.sol#1) allows old
versions
Pragma version^0.5.5 (@openzeppelin/contracts/utils/Address.sol#1) is known to contain
severe issues (https://solidity.readthedocs.io/en/latest/bugs.html)
Pragma version^0.5.0 (BankaiCoin.sol#2) allows old versions
Pragma version^0.5.0 (@openzeppelin/contracts/utils/ReentrancyGuard.sol#1) allows old
versions
Pragma version^0.5.0 (@openzeppelin/contracts/GSN/Context.sol#1) allows old versions
Pragma version^0.5.0 (@openzeppelin/contracts/crowdsale/Crowdsale.sol#1) allows old
versions
Pragma version^0.5.0 (@openzeppelin/contracts/lifecycle/Pausable.sol#1) allows old
versions
Pragma version^0.5.0 (PreIcoEscrow.sol#1) allows old versions
Pragma version^0.5.0 (BankaiCoinCrowdsale.sol#2) allows old versions
Pragma version^0.5.0 (@openzeppelin/contracts/token/ERC20/ERC20Detailed.sol#1) allows
old versions
Pragma version>=0.4.22<0.9.0 (Migrations.sol#2) is too complex
Pragma version^0.5.0 (@openzeppelin/contracts/ownership/Ownable.sol#1) allows old
versions
Pragma version^0.5.0 (@openzeppelin/contracts/token/ERC20/ERC20.sol#1) allows old
versions
Pragma version^0.5.0 (@openzeppelin/contracts/token/ERC20/IERC20.sol#1) allows old
versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity
INFO:Detectors:
Low level call in SafeERC20.callOptionalReturn(IERC20,bytes) (@openzeppelin/contracts/
token/ERC20/SafeERC20.sol#55-74):
        - (success,returndata) = address(token).call(data) (@openzeppelin/contracts/
token/ERC20/SafeERC20.sol#67)
Low level call in Address.sendValue(address,uint256) (@openzeppelin/contracts/utils/
Address.sol#63-69):
        - (success) = recipient.call.value(amount)() (@openzeppelin/contracts/utils/
Address.sol#67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls
INFO:Detectors:
Parameter BankaiCoinAirdrop.newAirdropTransfer(address,uint256)._to
(BankaiCoinAirdrop.sol#8) is not in mixedCase
Parameter BankaiCoinAirdrop.newAirdropTransfer(address,uint256)._airdropAmount
(BankaiCoinAirdrop.sol#8) is not in mixedCase
Parameter BankaiCoin.setCrowdsaleAddress(address,address)._crowdsaleAddress
(BankaiCoin.sol#43) is not in mixedCase
Parameter BankaiCoin.setCrowdsaleAddress(address,address)._escrowAddress
(BankaiCoin.sol#43) is not in mixedCase
Parameter PreIcoEscrow.buyTokensFromEscrow(address)._beneficiary (PreIcoEscrow.sol#22)
```

```
is not in mixedCase
Parameter BankaiCoinCrowdsale.buyTokensFromCrowdsale(address)._beneficiary
(BankaiCoinCrowdsale.sol#26) is not in mixedCase
Variable Migrations.last_completed_migration (Migrations.sol#6) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (@openzeppelin/contracts/GSN/Context.sol#24)" inContext
(@openzeppelin/contracts/GSN/Context.sol#13-27)
Redundant expression "this (@openzeppelin/contracts/crowdsale/Crowdsale.sol#142)"
inCrowdsale (@openzeppelin/contracts/crowdsale/Crowdsale.sol#21-200)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-
statements
INFO:Detectors:
Variable BankaiCoin.permission0 (BankaiCoin.sol#27) is too similar to
BankaiCoin.permission1 (BankaiCoin.sol#26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-
are-too-similar
INFO:Detectors:
burn(uint256) should be declared external:
        - ERC20Burnable.burn(uint256) (@openzeppelin/contracts/token/ERC20/
ERC20Burnable.sol#17-19)
burnFrom(address,uint256) should be declared external:
        - ERC20Burnable.burnFrom(address,uint256) (@openzeppelin/contracts/token/ERC20/
ERC20Burnable.sol#24-26)
addPauser(address) should be declared external:
        - PauserRole.addPauser(address) (@openzeppelin/contracts/access/roles/
PauserRole.sol#27-29)
renouncePauser() should be declared external:
        - PauserRole.renouncePauser() (@openzeppelin/contracts/access/roles/
PauserRole.sol#31-33)
newAirdropTransfer(address,uint256) should be declared external:
        - BankaiCoinAirdrop.newAirdropTransfer(address,uint256)
(BankaiCoinAirdrop.sol#8-10)
mint(address,uint256) should be declared external:
        - BankaiCoin.mint(address,uint256) (BankaiCoin.sol#39-41)
setCrowdsaleAddress(address,address) should be declared external:
        - BankaiCoin.setCrowdsaleAddress(address,address) (BankaiCoin.sol#43-46)
startPreIco() should be declared external:
        - BankaiCoin.startPreIco() (BankaiCoin.sol#48-52)
startIco() should be declared external:
        - BankaiCoin.startIco() (BankaiCoin.sol#54-58)
token() should be declared external:
        - Crowdsale.token() (@openzeppelin/contracts/crowdsale/Crowdsale.sol#80-82)
wallet() should be declared external:
        - Crowdsale.wallet() (@openzeppelin/contracts/crowdsale/Crowdsale.sol#87-89)
rate() should be declared external:
        - Crowdsale.rate() (@openzeppelin/contracts/crowdsale/Crowdsale.sol#94-96)
weiRaised() should be declared external:
        - Crowdsale.weiRaised() (@openzeppelin/contracts/crowdsale/
Crowdsale.sol#101-103)
paused() should be declared external:
        - Pausable.paused() (@openzeppelin/contracts/lifecycle/Pausable.sol#39-41)
pause() should be declared external:
        - Pausable.pause() (@openzeppelin/contracts/lifecycle/Pausable.sol#62-65)
unpause() should be declared external:
        - Pausable.unpause() (@openzeppelin/contracts/lifecycle/Pausable.sol#70-73)
buyTokensFromEscrow(address) should be declared external:
        - PreIcoEscrow.buyTokensFromEscrow(address) (PreIcoEscrow.sol#22-28)
stopPreIco(IERC20,address) should be declared external:
        - PreIcoEscrow.stopPreIco(IERC20,address) (PreIcoEscrow.sol#30-36)
buyTokensFromCrowdsale(address) should be declared external:
        - BankaiCoinCrowdsale.buyTokensFromCrowdsale(address)
(BankaiCoinCrowdsale.sol#26-32)
stopIco(IERC20,address) should be declared external:
        - BankaiCoinCrowdsale.stopIco(IERC20,address) (BankaiCoinCrowdsale.sol#34-40)
name() should be declared external:
        - ERC20Detailed.name() (@openzeppelin/contracts/token/ERC20/
ERC20Detailed.sol#27-29)
symbol() should be declared external:
        - ERC20Detailed.symbol() (@openzeppelin/contracts/token/ERC20/
ERC20Detailed.sol#35-37)
decimals() should be declared external:
        - ERC20Detailed.decimals() (@openzeppelin/contracts/token/ERC20/
ERC20Detailed.sol#51-53)
setCompleted(uint256) should be declared external:
        - Migrations.setCompleted(uint256) (Migrations.sol#16-18)
owner() should be declared external:
        - Ownable.owner() (@openzeppelin/contracts/ownership/Ownable.sol#30-32)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (@openzeppelin/contracts/ownership/
Ownable.sol#56-59)
```

```
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (@openzeppelin/contracts/ownership/
Ownable.sol#65-67)
totalSupply() should be declared external:
        - ERC20.totalSupply() (@openzeppelin/contracts/token/ERC20/ERC20.sol#43-45)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (@openzeppelin/contracts/token/ERC20/
ERC20.sol#70-72)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (@openzeppelin/contracts/token/ERC20/
ERC20.sol#81-84)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (@openzeppelin/contracts/token/
ERC20/ERC20.sol#98-102)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (@openzeppelin/contracts/token/ERC20/
ERC20.sol#116-119)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (@openzeppelin/contracts/token/ERC20/
ERC20.sol#135-138)
```

Guard