Guard

Smart contracts security assessment

Final report

Tariff: Standard

Idoru Capital

February 2022

0xguard.com

hello@0xguard.com

Contents

Introduction

ERC20 token contract with the payment of dividends, whitelists, and voting. The code is available in the Github repository. The code was checked in 26a65bc commit. ERC20 interface is realized with the use of OpenZeppelin libraries, which is considered the best practice. The md5 sum of the files:

1) Bank.sol is a1232621f2ab8e6ce56e9b9b5b76365c.

2) Dividends.sol is 0412d60baaff8496e302a34e2eec266f.

3) IdoruMinter.sol is 5814a72f48d8a865b693bb6c27f5e8f0.

4) ERC20CVotes.sol is 11aa054e8e7993c1f3387bbe80ba8db7.

5) Idoru.sol is 6f9c8619aa3d14463119404ade8ff7f4.

6) IdoruStablecoin.sol is 7141a4153aa17b68199809f08188c80b.

7) Verifiable.sol is 6745d8f353011db104b56e66a564c989.

Name	Idoru Capital
Audit date	2022-02-16 - 2022-02-17
Language	Solidity
Platform	Ethereum

Contracts checked

Name	Address
ERC20Verifiable	https://github.com/Idoru-Capital/idoru-platform/blob/26a65bc2e77ed58a384300097788911cd310ed6e/contracts/token/Verifiable.sol
Multiple contract	

IdoruMinter	https://github.com/Idoru-Capital/idoru-platform/blob/26a65bc2e77ed58a384300097788911cd310ed6e/contracts/platform/IdoruMinter.sol
IdoruDividends	https://github.com/Idoru-Capital/idoru-platform/blob/26a65bc2e77ed58a384300097788911cd310ed6e/contracts/platform/Dividends.sol
Idoru	https://github.com/Idoru-Capital/idoru-platform/blob/26a65bc2e77ed58a384300097788911cd310ed6e/contracts/token/Idoru.sol

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	passed
Message call with hardcoded gas amount	passed
Typographical Error	not 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
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	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. Price calculation with getReserves() (IdoruMinter)

In the getIdoruAmountOut() function, the _amountOut variable is calculated based on the reserves in the pool. This cannot be done, as the price can be influenced by flash loans(link). Also, after calling the external getReserves() function, your reserves are not sorted using the sortTokens() function, this can lead to incorrect rate calculation.

Recommendation: It is recommended to use price oracles through the backend or Chainlink. You can also use UniswapV2 oracles(link).

2. Unlimited mint (IdoruMinter)

Using the changerewardPoints() function, you can influence the result of the getIdoruAmountOut() function, this can lead to an unlimited mint of tokens in the swapStablecoinIdoru() function.

Recommendation: It is recommended to either remove the changerewardPoints() function or adjust

the number of issued coins.

3. Variable not initialized in constructor (IdoruDividends)

The constructor() does not initialize the bankAddress variable. This mistake can lead to a loss of funds.

Recommendation: You need to initialize this variable in the constructor().

4. Wrong indexing in arrays (IdoruDividends)

The deleteDividends() function removes an element from the dividendBlocks array, but the dividendsPaid and dividendsAmounts arrays remain unchanged in length. This can break the logic of other functions and lead to critical errors.

Recommendation: You need to think about whether the deleteDividends() function is needed at all. If necessary, then come up with another solution for removing dividends so that the indexing in all three arrays is the same.

Medium severity issues

1. Incorrect rate calculation (IdoruMinter)

In line 206 there is a check that _stablecoin is equal to one of two options. The course may differ for the coins being checked, which may lead to an incorrect calculation of the course.

Recommendation: It is recommended to check _stablecoin against one specific stablecoin address.

2. Invalid expression in loop (IdoruDividends)

In the 109 line of the code, the condition i > 0 is checked, which leads to the fact that the zero element of the dividendBlocks array will never enter the loop.

Recommendation: Change the sign in the condition to >=.

3. Non-adjustable mint (Idoru)

The mint() function can be called by anyone with the MINTER role. It is not known how many people have this role and because of this it is difficult to regulate the mint of this token.

Recommendation: It is recommended in such cases to use AccessControlEnumerable from openzeppelin(link).

Low severity issues

1. Unused imports (ERC20Verifiable)

The imports "@openzeppelin/contracts/utils/math/Math.sol" and "@openzeppelin/contracts/utils/math/SafeCast.sol" are not used in the contract.

Recommendation: It is recommended to remove unused imports as they make the code less readable.

2. Floating Pragma (Multiple contract)

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Recommendation: Lock the pragma version and also consider known bugs (link) for the compiler version that is chosen.

3. Few events or they are not used (Multiple contract)

Many functions from the contracts lack events:

1) Bank.sol:

- addBankUser()

- removeBankUser()

- withdrawTokens()

- withdrawEther()

2) IdoruMinter.sol:

- setPresaleTokensToMint()

- setAvailableTokensToMint()

- changerewardPoints()

- changePricePresale()

- changeIdoruAddress()

- changeBankAddress()

- mintIdoru().

3) In Dividends.sol created events are not used:

- event PayeeAdded()

- event ERC20PaymentReleased()

- event PaymentReceived()

Recommendation: Create events for these functions and use the ones already created.

4. Typos (Multiple contract)

Typos reduce the code's readability.

1) Dividends.sol:77 line typo in word "becasue"

2) IdoruMinter.sol: 52 line, 94 line, 100 line missing sign "_" in 1000_000

Recommendation: Fix the typos.

5. Variables must have a public modifier (Multiple contract)

The following files contain variables that should be public but are private:

1) IdoruMinter.sol 23-34 lines

2) Dividends.sol 31-33 lines

Recommendation: For transparency of users' work with the platform, it is recommended to make these variables public.

6. Multiplication after division (IdoruMinter)

In 149 and 169 lines there is multiplication after division, which increases the possible error in integer math.

Recommendation: It is recommended to change the order of calculation of the expression in these lines.

7. Lacks a zero-check on constructor (IdoruMinter)

Recommendation: Add input validations with require in constructor().

8. Invalid error names (IdoruMinter)

In the lines 89, 97, 127-128, 145-148, 165-168 invalid error names are used in require statements.

For example, in the require(_fixedPricePresale > 0, "Negative presale price"), if _fixedPricePresale is zero, then it is not a negative number.

Recommendation: It is recommended to change the names of the errors to more appropriate ones according to the meaning of the code.

9. No tests (IdoruDividends)

Tests are needed to quickly find errors in the code and check business logic.

Recommendation: It is recommended to write covered tests for the IdoruDividends contract.

10. No getters for state variables (IdoruDividends)

Recommendation: It is recommended to create get functions for these variables.

11. Redundant code (IdoruDividends)

The claimDividends() function uses a call to withdrawAllDividendsView() which can be replaced by a call to withdrawAllDividendsChangeState(). Line 137 is redundant.

Recommendation: It is recommended to delete line 137 and change line 138 to `uint256 amount = withdrawAllDividendsChangeState();`.

12. High gas consumption in cycles (IdoruDividends)

Loops over arrays are used in many functions of this contract(). This greatly increases the cost of gas during the execution of the contract.

Recommendation: An enumerableMap from openzeppelin(link) or binary search can help you. It is also possible that writing a data structure for quick summation on an array segment will help you(dynamic programming and RSQ task).

13. Variables must be immutable (IdoruDividends)

State variables in the 31-33 lines must be declared immutable.

Recommendation: Make these state variables immutable and remove set functions for them.

14. Redundant access to the state variable (IdoruDividends)

The following lines contain redundant access to variables:

1) 110-113 lines used dividendBlocks[i]

2) 94-98 lines used dividendsAmounts[blockWithdraw]

3) 125-130 lines used dividendBlocks[i]

Recommendation: Make a copy of the variable in memory and refer to it. This will save gas consumption.

15. Functions not used (IdoruDividends)

The withdrawAllDividendsChangeState() and claimDividends() functions are not called anywhere and have a private modifier.

Recommendation: Complete these functions or remove them from the codebase. Unused code worsens code readability and increases deployment gas consumption.

Conclusion

Idoru Capital ERC20Verifiable, Multiple contract, IdoruMinter, IdoruDividends, Idoru contracts were audited. 4 high, 3 medium, 15 low 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.

Slither output

```
ERC20Bank.withdrawTokens(address,uint256) (contracts/platform/Bank.sol#43-51) ignores
return value by token.transfer(msg.sender,_amount) (contracts/platform/Bank.sol#50)
IdoruDividends.payDividends(uint256,uint256) (contracts/platform/Dividends.sol#53-59)
ignores return value by stableERC20.transferFrom(bankAddress,address(this),amount)
(contracts/platform/Dividends.sol#58)
IdoruDividends.deleteDividends(uint256) (contracts/platform/Dividends.sol#61-84)
ignores return value by stableERC20.transferFrom(address(this),msg.sender,remaining)
(contracts/platform/Dividends.sol#82)
IdoruDividends.claimDividends() (contracts/platform/Dividends.sol#135-140) ignores
return value by stableERC20.transferFrom(address(this),msg.sender,amount) (contracts/
platform/Dividends.sol#139)
IdoruMinter.swapStablecoinIdoru(uint256,address) (contracts/platform/
IdoruMinter.sol#201-225) ignores return value by
stableERC20.transferFrom(msg.sender,bankAddress,_stablecoinAmount) (contracts/platform/
IdoruMinter.sol#211)
IdoruMinter.mintIdoruPresale(uint256,address) (contracts/platform/
IdoruMinter.sol#233-257) ignores return value by
stableERC20.transferFrom(msg.sender,bankAddress,_stablecoinAmount) (contracts/platform/
IdoruMinter.sol#243)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-
transfer

IdoruDividends.bankAddress (contracts/platform/Dividends.sol#33) is never initialized.
It is used in:
        - IdoruDividends.payDividends(uint256,uint256) (contracts/platform/
Dividends.sol#53-59)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-
state-variables

IdoruMinter.getIdoruAmountOut(uint256) (contracts/platform/IdoruMinter.sol#135-150)
performs a multiplication on the result of a division:
        -_amountOut = (((_amountIn * res1) / res0) * rewardPoints) / 10_000 (contracts/
platform/IdoruMinter.sol#149)
IdoruMinter.getIdoruAmountIn(uint256) (contracts/platform/IdoruMinter.sol#155-170)
performs a multiplication on the result of a division:
        -_amountIn = (((_amountOut * res0) / res1) * 10_000) / rewardPoints (contracts/
platform/IdoruMinter.sol#169)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply

UniswapV2Library.getAmountsOut(address,uint256,address[]).i (contracts/uniswap/
UniswapV2Library.sol#119) is a local variable never initialized
IdoruDividends.deleteDividends(uint256).indexOfBlock (contracts/platform/
Dividends.sol#68) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-
local-variables

IdoruMinter.changeIdoruAddress(address) (contracts/platform/IdoruMinter.sol#106-108)
should emit an event for:
        - idoruAddress = _addr (contracts/platform/IdoruMinter.sol#107)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
access-control

IdoruMinter.changerewardPoints(uint256) (contracts/platform/IdoruMinter.sol#88-91)
should emit an event for:
        - rewardPoints = _rewardPoints (contracts/platform/IdoruMinter.sol#90)
IdoruMinter.changePricePresale(uint256) (contracts/platform/IdoruMinter.sol#96-101)
should emit an event for:
        - fixedPricePresale = (_fixedPricePresale *
uint256(IIdoru(idoruAddress).decimals())) / 1000_000 (contracts/platform/
IdoruMinter.sol#98-100)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
arithmetic

IdoruDividends.constructor(address,address).stablecoinAddress_ (contracts/platform/
Dividends.sol#43) lacks a zero-check on :
                - _stablecoinToken = stablecoinAddress_ (contracts/platform/
Dividends.sol#44)
IdoruDividends.constructor(address,address).idoruTokenAddress_ (contracts/platform/
Dividends.sol#43) lacks a zero-check on :
                - _idoruToken = idoruTokenAddress_ (contracts/platform/
Dividends.sol#45)
IdoruMinter.constructor(address,address,address,address,address)._uniswapFactory
(contracts/platform/IdoruMinter.sol#37) lacks a zero-check on :
                - uniswapFactoryAddress = _uniswapFactory (contracts/platform/
IdoruMinter.sol#43)
IdoruMinter.constructor(address,address,address,address,address)._idoru (contracts/
platform/IdoruMinter.sol#38) lacks a zero-check on :
```

```
                - idoruAddress = _idoru (contracts/platform/IdoruMinter.sol#44)
IdoruMinter.constructor(address,address,address,address,address)._stablecoin (contracts/
platform/IdoruMinter.sol#39) lacks a zero-check on :
                - usdStableCoin = _stablecoin (contracts/platform/IdoruMinter.sol#45)
IdoruMinter.constructor(address,address,address,address,address)._idoruStablecoin
(contracts/platform/IdoruMinter.sol#40) lacks a zero-check on :
                - usdIdoruCoin = _idoruStablecoin (contracts/platform/
IdoruMinter.sol#46)
IdoruMinter.constructor(address,address,address,address,address)._bank (contracts/
platform/IdoruMinter.sol#41) lacks a zero-check on :
                - bankAddress = _bank (contracts/platform/IdoruMinter.sol#47)
IdoruMinter.changeIdoruAddress(address)._addr (contracts/platform/IdoruMinter.sol#106)
lacks a zero-check on :
                - idoruAddress = _addr (contracts/platform/IdoruMinter.sol#107)
IdoruMinter.changeBankAddress(address)._addr (contracts/platform/IdoruMinter.sol#113)
lacks a zero-check on :
                - bankAddress = _addr (contracts/platform/IdoruMinter.sol#114)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-
address-validation

IdoruDividends.withdrawDividendsBlock(uint256) (contracts/platform/
Dividends.sol#86-100) has external calls inside a loop: pastTotalSupply =
IIdoru(_idoruToken).getPastTotalSupply(blockWithdraw) (contracts/platform/
Dividends.sol#91-93)
IdoruDividends.withdrawDividendsBlock(uint256) (contracts/platform/
Dividends.sol#86-100) has external calls inside a loop: toWithdraw =
((dividendsAmounts[blockWithdraw] + dividendsPaid[blockWithdraw]) *
IIdoru(_idoruToken).minHoldingValue(msg.sender,blockWithdraw)) / pastTotalSupply
(contracts/platform/Dividends.sol#94-97)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-
a-loop

Reentrancy in IdoruMinter.mintIdoruPresale(uint256,address) (contracts/platform/
IdoruMinter.sol#233-257):
        External calls:
        - stableERC20.transferFrom(msg.sender,bankAddress,_stablecoinAmount) (contracts/
platform/IdoruMinter.sol#243)
        State variables written after the call(s):
        - presaleTokensToMint -= idoruAmount (contracts/platform/IdoruMinter.sol#247)
Reentrancy in IdoruMinter.swapStablecoinIdoru(uint256,address) (contracts/platform/
IdoruMinter.sol#201-225):
```

```
        External calls:
        - stableERC20.transferFrom(msg.sender,bankAddress,_stablecoinAmount) (contracts/
platform/IdoruMinter.sol#211)
        State variables written after the call(s):
        - availableTokensToMint -= idoruAmount (contracts/platform/IdoruMinter.sol#215)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-2

Reentrancy in IdoruMinter.mintIdoruPresale(uint256,address) (contracts/platform/
IdoruMinter.sol#233-257):
        External calls:
        - stableERC20.transferFrom(msg.sender,bankAddress,_stablecoinAmount) (contracts/
platform/IdoruMinter.sol#243)
        - mintIdoru(msg.sender,idoruAmount) (contracts/platform/IdoruMinter.sol#249)
                - IIdoru(idoruAddress).mint(to,amount) (contracts/platform/
IdoruMinter.sol#180)
        Event emitted after the call(s):
        - SwapForIdoru(_stablecoin,idoruAddress,_stablecoinAmount,idoruAmount)
(contracts/platform/IdoruMinter.sol#251-256)
Reentrancy in IdoruMinter.swapStablecoinIdoru(uint256,address) (contracts/platform/
IdoruMinter.sol#201-225):
        External calls:
        - stableERC20.transferFrom(msg.sender,bankAddress,_stablecoinAmount) (contracts/
platform/IdoruMinter.sol#211)
        - mintIdoru(msg.sender,idoruAmount) (contracts/platform/IdoruMinter.sol#217)
                - IIdoru(idoruAddress).mint(to,amount) (contracts/platform/
IdoruMinter.sol#180)
        Event emitted after the call(s):
        - SwapForIdoru(_stablecoin,idoruAddress,_stablecoinAmount,idoruAmount)
(contracts/platform/IdoruMinter.sol#219-224)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3

IdoruDividends.claimDividends() (contracts/platform/Dividends.sol#135-140) is never
used and should be removed
IdoruDividends.withdrawAllDividendsChangeState() (contracts/platform/
Dividends.sol#118-133) is never used and should be removed
SafeMath.add(uint256,uint256) (contracts/uniswap/SafeMath.sol#8-10) is never used and
should be removed
SafeMath.mul(uint256,uint256) (contracts/uniswap/SafeMath.sol#16-18) is never used and
should be removed
```

```
SafeMath.sub(uint256,uint256) (contracts/uniswap/SafeMath.sol#12-14) is never used and
should be removed
UniswapV2Library.getAmountIn(uint256,uint256,uint256) (contracts/uniswap/
UniswapV2Library.sol#95-108) is never used and should be removed
UniswapV2Library.getAmountOut(uint256,uint256,uint256) (contracts/uniswap/
UniswapV2Library.sol#78-92) is never used and should be removed
UniswapV2Library.getAmountsIn(address,uint256,address[]) (contracts/uniswap/
UniswapV2Library.sol#130-146) is never used and should be removed
UniswapV2Library.getAmountsOut(address,uint256,address[]) (contracts/uniswap/
UniswapV2Library.sol#111-127) is never used and should be removed
UniswapV2Library.quote(uint256,uint256,uint256) (contracts/uniswap/
UniswapV2Library.sol#64-75) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.2 (contracts/interfaces/Idoru.interface.sol#2) allows old versions
Pragma version^0.8.2 (contracts/platform/Bank.sol#5) allows old versions
Pragma version^0.8.2 (contracts/platform/Dividends.sol#4) allows old versions
Pragma version^0.8.2 (contracts/platform/IdoruMinter.sol#2) allows old versions
Pragma version^0.8.2 (contracts/token/Constants.sol#3) allows old versions
Pragma version^0.8.2 (contracts/token/ERC20CVotes.sol#3) allows old versions
Pragma version^0.8.2 (contracts/token/Idoru.sol#3) allows old versions
Pragma version^0.8.2 (contracts/token/IdoruStablecoin.sol#2) allows old versions
Pragma version^0.8.2 (contracts/token/Verifiable.sol#3) allows old versions
Pragma version^0.8.2 (contracts/uniswap/SafeMath.sol#3) allows old versions
Pragma version^0.8.2 (contracts/uniswap/UniswapV2Library.sol#3) allows old versions
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity

Idoru (contracts/token/Idoru.sol#19-78) should inherit from IIdoru (contracts/
interfaces/Idoru.interface.sol#9-27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-
inheritance

Parameter ERC20Bank.isBankUser(address)._user (contracts/platform/Bank.sol#17) is not
in mixedCase
Parameter ERC20Bank.addBankUser(address)._user (contracts/platform/Bank.sol#29) is not
in mixedCase
Parameter ERC20Bank.removeBankUser(address)._user (contracts/platform/Bank.sol#36) is
not in mixedCase
Parameter ERC20Bank.withdrawTokens(address,uint256)._token (contracts/platform/
```

```
Bank.sol#43) is not in mixedCase
Parameter ERC20Bank.withdrawTokens(address,uint256)._amount (contracts/platform/
Bank.sol#43) is not in mixedCase
Parameter ERC20Bank.withdrawEther(uint256)._amount (contracts/platform/Bank.sol#56) is
not in mixedCase
Parameter IdoruMinter.setPresaleTokensToMint(uint256)._tokens (contracts/platform/
IdoruMinter.sol#71) is not in mixedCase
Parameter IdoruMinter.setAvailableTokensToMint(uint256)._tokens (contracts/platform/
IdoruMinter.sol#79) is not in mixedCase
Parameter IdoruMinter.changerewardPoints(uint256)._rewardPoints (contracts/platform/
IdoruMinter.sol#88) is not in mixedCase
Parameter IdoruMinter.changePricePresale(uint256)._fixedPricePresale (contracts/platform/
IdoruMinter.sol#96) is not in mixedCase
Parameter IdoruMinter.changeIdoruAddress(address)._addr (contracts/platform/
IdoruMinter.sol#106) is not in mixedCase
Parameter IdoruMinter.changeBankAddress(address)._addr (contracts/platform/
IdoruMinter.sol#113) is not in mixedCase
Parameter IdoruMinter.getIdoruPresaleAmountOut(uint256)._amountIn (contracts/platform/
IdoruMinter.sol#122) is not in mixedCase
Parameter IdoruMinter.getIdoruAmountOut(uint256)._amountIn (contracts/platform/
IdoruMinter.sol#135) is not in mixedCase
Parameter IdoruMinter.getIdoruAmountIn(uint256)._amountOut (contracts/platform/
IdoruMinter.sol#155) is not in mixedCase
Parameter IdoruMinter.swapStablecoinIdoru(uint256,address)._stablecoinAmount (contracts/
platform/IdoruMinter.sol#201) is not in mixedCase
Parameter IdoruMinter.swapStablecoinIdoru(uint256,address)._stablecoin (contracts/
platform/IdoruMinter.sol#201) is not in mixedCase
Parameter IdoruMinter.mintIdoruPresale(uint256,address)._stablecoinAmount (contracts/
platform/IdoruMinter.sol#233) is not in mixedCase
Parameter IdoruMinter.mintIdoruPresale(uint256,address)._stablecoin (contracts/platform/
IdoruMinter.sol#233) is not in mixedCase
Parameter ERC20CVotes.changeMinHoldingBlocks(uint256)._minHoldingBlocks (contracts/
token/ERC20CVotes.sol#27) is not in mixedCase
Parameter ERC20CVotes.minHoldingValue(address,uint256)._addr (contracts/token/
ERC20CVotes.sol#41) is not in mixedCase
Parameter ERC20Verifiable.verifyAddress(address)._addr (contracts/token/
Verifiable.sol#23) is not in mixedCase
Parameter ERC20Verifiable.unVerifyAddress(address)._addr (contracts/token/
Verifiable.sol#30) is not in mixedCase
Parameter ERC20Verifiable.isVerified(address)._addr (contracts/token/Verifiable.sol#35) is
not in mixedCase
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

IdoruDividends.bankAddress (contracts/platform/Dividends.sol#33) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

removeBankUser(address) should be declared external:
        - ERC20Bank.removeBankUser(address) (contracts/platform/Bank.sol#36-38)
withdrawTokens(address,uint256) should be declared external:
        - ERC20Bank.withdrawTokens(address,uint256) (contracts/platform/Bank.sol#43-51)
withdrawEther(uint256) should be declared external:
        - ERC20Bank.withdrawEther(uint256) (contracts/platform/Bank.sol#56-60)
payDividends(uint256,uint256) should be declared external:
        - IdoruDividends.payDividends(uint256,uint256) (contracts/platform/
Dividends.sol#53-59)
deleteDividends(uint256) should be declared external:
        - IdoruDividends.deleteDividends(uint256) (contracts/platform/
Dividends.sol#61-84)
setPresaleTokensToMint(uint256) should be declared external:
        - IdoruMinter.setPresaleTokensToMint(uint256) (contracts/platform/
IdoruMinter.sol#71-74)
setAvailableTokensToMint(uint256) should be declared external:
        - IdoruMinter.setAvailableTokensToMint(uint256) (contracts/platform/
IdoruMinter.sol#79-82)
changerewardPoints(uint256) should be declared external:
        - IdoruMinter.changerewardPoints(uint256) (contracts/platform/
IdoruMinter.sol#88-91)
changePricePresale(uint256) should be declared external:
        - IdoruMinter.changePricePresale(uint256) (contracts/platform/
IdoruMinter.sol#96-101)
changeIdoruAddress(address) should be declared external:
        - IdoruMinter.changeIdoruAddress(address) (contracts/platform/
IdoruMinter.sol#106-108)
changeBankAddress(address) should be declared external:
        - IdoruMinter.changeBankAddress(address) (contracts/platform/
IdoruMinter.sol#113-115)
getIdoruAmountIn(uint256) should be declared external:
        - IdoruMinter.getIdoruAmountIn(uint256) (contracts/platform/
IdoruMinter.sol#155-170)
swapStablecoinIdoru(uint256,address) should be declared external:
```

```
        - IdoruMinter.swapStablecoinIdoru(uint256,address) (contracts/platform/
IdoruMinter.sol#201-225)
mintIdoruPresale(uint256,address) should be declared external:
        - IdoruMinter.mintIdoruPresale(uint256,address) (contracts/platform/
IdoruMinter.sol#233-257)
pause() should be declared external:
        - IdoruStableCoin.pause() (contracts/token/IdoruStablecoin.sol#12-14)
unpause() should be declared external:
        - IdoruStableCoin.unpause() (contracts/token/IdoruStablecoin.sol#16-18)
mint(address,uint256) should be declared external:
        - IdoruStableCoin.mint(address,uint256) (contracts/token/
IdoruStablecoin.sol#20-22)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external
. analyzed (35 contracts with 77 detectors), 94 result(s) found
```

0x Guard