komet.finance

Security Code Review

https://twitter.com/VidarTheAuditor - 30 November 2020

Overview

Project Summary

Project Name	komet.finance
Description	New Decentralized Hedge Fund
Platform	Ethereum, Solidity
Codebase	https://github.com/KometFinance/komet-finance (commit 2defae47f8debb5b4992d096285a5d965d3cdc20)
Deployed contracts	Komet - 0x6CfB6dF56BbdB00226AeFfCdb2CD1FE8Da1ABdA7

Executive Summary

The codebase was found well defined, has proper access restrictions where needed, includes very good comments throughout a code.

We have run extensive static analysis of the codebase as well as standard security assessment utilising industry approved tools. We have found no major issues during our review that are described later in the report, however overall impression of the code is very good.

We have manually reviewed the code and confirmed the functionality wit provided tests.

Project is using industry standard libraries (OpenZeppelin) which is a recommended and good practise.

Project has very comprehensive test library that was used to conduct tests with successful results.

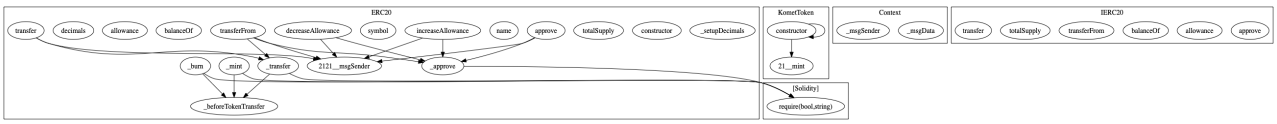
One recommendation was provided to include proper governance on top of MasterUniverse contract.

Disclaimer: The analysis did not include any tokenomics analysis (e.g. APY rates etc).

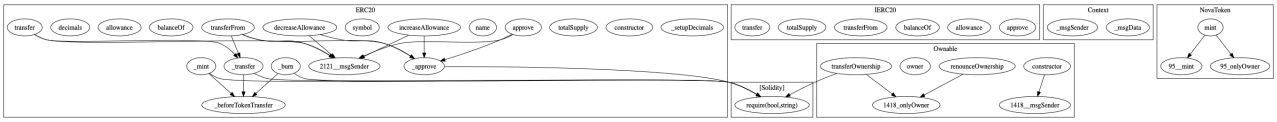
Architecture & Standards

Please find below the calling architecture of the reviewed contracts.

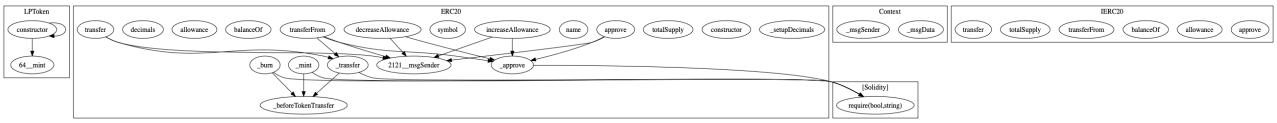
KometToken:



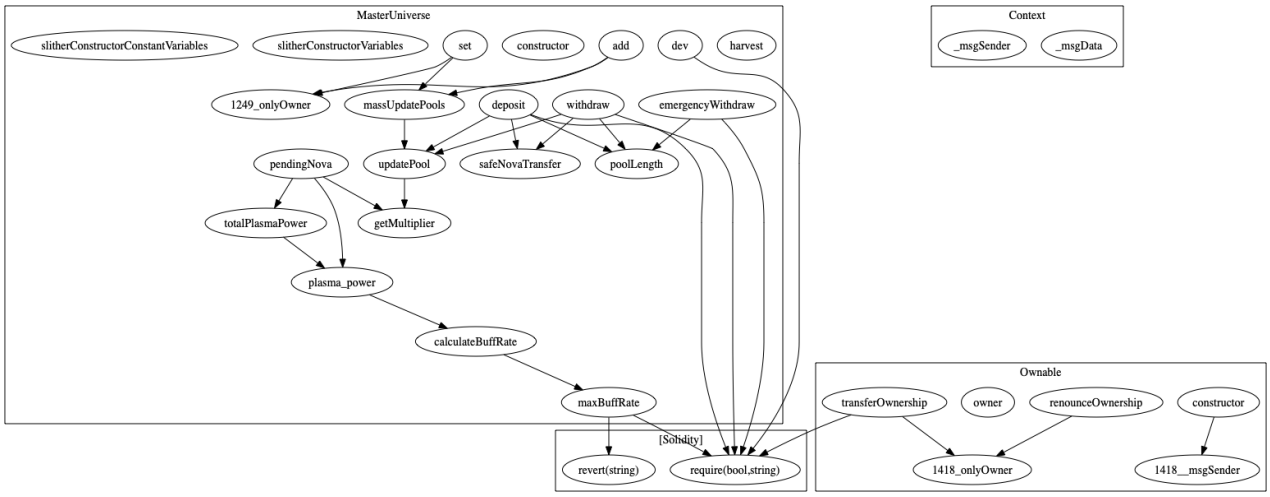
NovaToken:

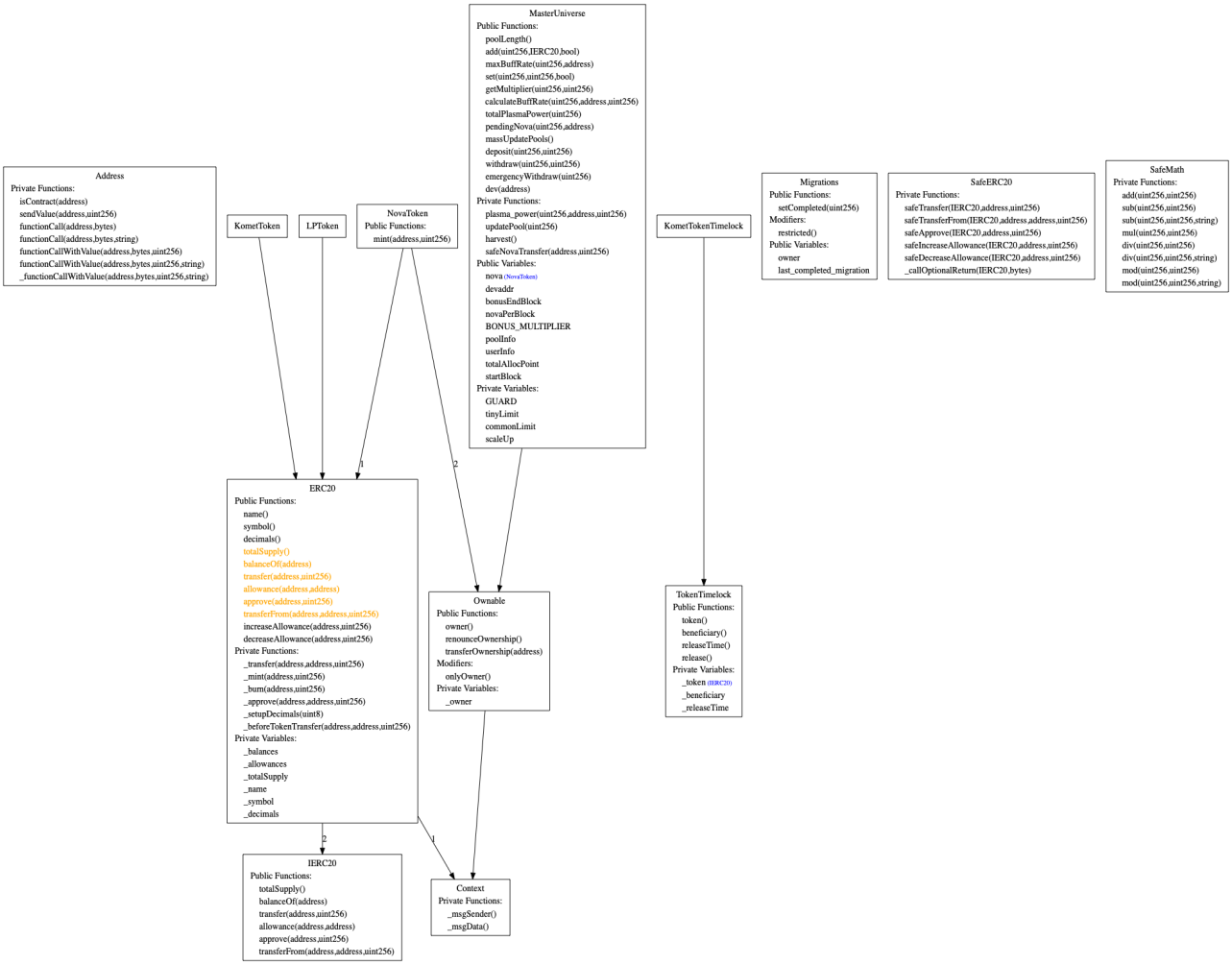


LPToken:



MasterUniverse:




Address
Private Functions:
isContract(address)
sendValue(address,uint256)
functionCall(address,bytes)
functionCall(address,bytes,string)
functionCallWithValue(address,bytes,uint256)
functionCallWithValue(address,bytes,uint256,string)
_functionCallWithValue(address,bytes,uint256,string)
KometToken
LPToken
NovaToken
Public Functions:
mint(address,uint256)
MasterUniverse
Public Functions:
poolLength()
add(uint256,IERC20,bool)
maxBuffRate(uint256,address)
set(uint256,uint256,bool)
getMultiplier(uint256,uint256)
calculateBuffRate(uint256,address,uint256)
totalPlasmaPower(uint256)
pendingNova(uint256,address)
massUpdatePools()
deposit(uint256,uint256)
withdraw(uint256,uint256)
emergencyWithdraw(uint256)
dev(address)
Private Functions:
plasma_power(uint256,address,uint256)
updatePool(uint256)
harvest()
safeNovaTransfer(address,uint256)
Public Variables:
nova (NovaToken)
devaddr
bonusEndBlock
novaPerBlock
BONUS_MULTIPLIER
poolInfo
userInfo
totalAllocPoint
startBlock
Private Variables:
GUARD
tinyLimit
commonLimit
scaleUp
KometTokenTimelock
Migrations
Public Functions:
setCompleted(uint256)
Modifiers:
restricted()
Public Variables:
owner
last_completed_migration
SafeERC20
Private Functions:
safeTransfer(IERC20,address,uint256)
safeTransferFrom(IERC20,address,address,uint256)
safeApprove(IERC20,address,uint256)
safeIncreaseAllowance(IERC20,address,uint256)
safeDecreaseAllowance(IERC20,address,uint256)
_callOptionalReturn(IERC20,bytes)
SafeMath
Private Functions:
add(uint256,uint256)
sub(uint256,uint256)
sub(uint256,uint256,string)
mul(uint256,uint256)
div(uint256,uint256)
div(uint256,uint256,string)
mod(uint256,uint256)
mod(uint256,uint256,string)
ERC20
Public Functions:
name()
symbol()
decimals()
totalSupply()
balanceOf(address)
transfer(address,uint256)
allowance(address,address)
approve(address,uint256)
transferFrom(address,address,uint256)
increaseAllowance(address,uint256)
decreaseAllowance(address,uint256)
Private Functions:
_transfer(address,address,uint256)
_mint(address,uint256)
_burn(address,uint256)
_approve(address,address,uint256)
_setupDecimals(uint8)
_beforeTokenTransfer(address,address,uint256)
Private Variables:
_balances
_allowances
_totalSupply
_name
_symbol
_decimals
Ownable
Public Functions:
owner()
renounceOwnership()
transferOwnership(address)
Modifiers:
onlyOwner()
Private Variables:
_owner
TokenTimelock
Public Functions:
token()
beneficiary()
releaseTime()
release()
Private Variables:
_token (IERC20)
_beneficiary
_releaseTime
IERC20
Public Functions:
totalSupply()
balanceOf(address)
transfer(address,uint256)
allowance(address,address)
approve(address,uint256)
transferFrom(address,address,uint256)
Context
Private Functions:
_msgSender()
_msgData()
3
2
2
1

Komet, Nova and LPToken are fully ERC-20 compliant.

```
# Check KometToken

## Check functions
[✓] totalSupply() is present
        [✓] totalSupply() -> () (correct return value)
        [✓] totalSupply() is view
[✓] balanceOf(address) is present
        [✓] balanceOf(address) -> () (correct return value)
        [✓] balanceOf(address) is view
[✓] transfer(address,uint256) is present
        [✓] transfer(address,uint256) -> () (correct return value)
        [✓] Transfer(address,address,uint256) is emitted
[✓] transferFrom(address,address,uint256) is present
        [✓] transferFrom(address,address,uint256) -> () (correct return value)
        [✓] Transfer(address,address,uint256) is emitted
[✓] approve(address,uint256) is present
        [✓] approve(address,uint256) -> () (correct return value)
        [✓] Approval(address,address,uint256) is emitted
[✓] allowance(address,address) is present
        [✓] allowance(address,address) -> () (correct return value)
        [✓] allowance(address,address) is view
[✓] name() is present
        [✓] name() -> () (correct return value)
        [✓] name() is view
[✓] symbol() is present
        [✓] symbol() -> () (correct return value)
        [✓] symbol() is view
[✓] decimals() is present
        [✓] decimals() -> () (correct return value)
        [✓] decimals() is view

## Check events
[✓] Transfer(address,address,uint256) is present
        [✓] parameter 0 is indexed
        [✓] parameter 1 is indexed
[✓] Approval(address,address,uint256) is present
        [✓] parameter 0 is indexed
        [✓] parameter 1 is indexed


        [✓] KometToken has increaseAllowance(address,uint256)
```

```
# Check NovaToken

## Check functions
[✓] totalSupply() is present
        [✓] totalSupply() -> () (correct return value)
        [✓] totalSupply() is view
[✓] balanceOf(address) is present
        [✓] balanceOf(address) -> () (correct return value)
        [✓] balanceOf(address) is view
[✓] transfer(address,uint256) is present
        [✓] transfer(address,uint256) -> () (correct return value)
        [✓] Transfer(address,address,uint256) is emitted
[✓] transferFrom(address,address,uint256) is present
        [✓] transferFrom(address,address,uint256) -> () (correct return value)
        [✓] Transfer(address,address,uint256) is emitted
[✓] approve(address,uint256) is present
        [✓] approve(address,uint256) -> () (correct return value)
        [✓] Approval(address,address,uint256) is emitted
[✓] allowance(address,address) is present
        [✓] allowance(address,address) -> () (correct return value)
        [✓] allowance(address,address) is view
[✓] name() is present
        [✓] name() -> () (correct return value)
        [✓] name() is view
[✓] symbol() is present
        [✓] symbol() -> () (correct return value)
        [✓] symbol() is view
[✓] decimals() is present
        [✓] decimals() -> () (correct return value)
        [✓] decimals() is view

## Check events
[✓] Transfer(address,address,uint256) is present
        [✓] parameter 0 is indexed
        [✓] parameter 1 is indexed
[✓] Approval(address,address,uint256) is present
        [✓] parameter 0 is indexed
        [✓] parameter 1 is indexed


        [✓] NovaToken has increaseAllowance(address,uint256)
```

```
# Check LPToken

## Check functions
[✓] totalSupply() is present
        [✓] totalSupply() -> () (correct return value)
        [✓] totalSupply() is view
[✓] balanceOf(address) is present
        [✓] balanceOf(address) -> () (correct return value)
        [✓] balanceOf(address) is view
[✓] transfer(address,uint256) is present
        [✓] transfer(address,uint256) -> () (correct return value)
        [✓] Transfer(address,address,uint256) is emitted
[✓] transferFrom(address,address,uint256) is present
        [✓] transferFrom(address,address,uint256) -> () (correct return value)
        [✓] Transfer(address,address,uint256) is emitted
[✓] approve(address,uint256) is present
        [✓] approve(address,uint256) -> () (correct return value)
        [✓] Approval(address,address,uint256) is emitted
[✓] allowance(address,address) is present
        [✓] allowance(address,address) -> () (correct return value)
        [✓] allowance(address,address) is view
[✓] name() is present
        [✓] name() -> () (correct return value)
        [✓] name() is view
[✓] symbol() is present
        [✓] symbol() -> () (correct return value)
        [✓] symbol() is view
[✓] decimals() is present
        [✓] decimals() -> () (correct return value)
        [✓] decimals() is view

## Check events
[✓] Transfer(address,address,uint256) is present
        [✓] parameter 0 is indexed
        [✓] parameter 1 is indexed
[✓] Approval(address,address,uint256) is present
        [✓] parameter 0 is indexed
        [✓] parameter 1 is indexed


        [✓] LPToken has increaseAllowance(address,uint256)
```

Findings

Number of contracts: 6 (including inherited ones)

Use: Openzeppelin-ERC20, Openzeppelin-Ownable, Openzeppelin-SafeMath

Name	# functions	ERCS	ERC20 info	Complex code	Features
KometToken	27	ERC20	No Minting Approve Race Cond.	No	
KometTokenTimelock	6			No	Tokens interaction
LPToken	27	ERC20	No Minting Approve Race Cond.	No	
NovaToken	31	ERC20	∞ Minting Approve Race Cond.	No	
MasterUniverse	26			No	Tokens interaction
Migrations	2			No	

NovaToken has minting functionality, however it is not an issue if the ownership structure will be established on the mainet using the provided deployment scripts.

See Deployment section for more details.

Static Analysis Findings

High issues: None

Medium issues:

Divide before multiply:

```
MasterUniverse.pendingNova(uint256,address) (MasterUniverse.sol#248-276) performs a multiplication on the result of a division:
        -novaReward = multiplier.mul(novaPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (MasterUniverse.sol#265-268)
        -accNovaPerShare = accNovaPerShare.add(novaReward.mul(1e12).div(totalPlasma)) (MasterUniverse.sol#270-272)
MasterUniverse.updatePool(uint256) (MasterUniverse.sol#287-309) performs a multiplication on the result of a division:
        -novaReward = multiplier.mul(novaPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (MasterUniverse.sol#298-301)
        -pool.accNovaPerShare = pool.accNovaPerShare.add(novaReward.mul(1e12).div(lpSupply)) (MasterUniverse.sol#305-307)
```

Solidity integer division might truncate. It may happen that performing multiplication before divison might reduce precision.

Unused Return:

```
MasterUniverse.safeNovaTransfer(address,uint256) (MasterUniverse.sol#395-398) ignores return value by nova.transfer(_to,novaBal) (MasterUniverse.sol#397)
MasterUniverse.safeNovaTransfer(address,uint256) (MasterUniverse.sol#395-398) ignores return value by nova.transfer(_to,_amount) (MasterUniverse.sol#397)
```

Ensure that all the return values of the function calls are used.

[Manual Check] **Not an issue** as transfer function from ERC20.sol return true but it reverts from within _transfer is an error.

Low/Informational issues:

Block timestamps:

```
MasterUniverse.calculateBuffRate(uint256,address,uint256) (MasterUniverse.sol#192-214) uses timestamp for comparisons
        Dangerous comparisons:
        - d + 86400 <= _now_ (MasterUniverse.sol#203)
TokenTimelock.constructor(IERC20,address,uint256) (@openzeppelin/contracts/token/ERC20/TokenTimelock.sol#26-32) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(releaseTime_ > block.timestamp,TokenTimelock: release time is before current time) (@openzeppelin/contracts/token/ERC20/TokenTimelock.sol#28)
TokenTimelock.release() (@openzeppelin/contracts/token/ERC20/TokenTimelock.sol#58-66) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= _releaseTime,TokenTimelock: current time is before release time) (@openzeppelin/contracts/token/ERC20/TokenTimelock.sol#60)
```

It is possible that miners could manipulate block.timestamp. block.timestamp.

[Manual Check] Timestamps are part of the staking functionality, the risk is not significant.

Dynamic Tests

We have run fuzzing / property-based testing of Ethereum smarts contracts. It was using sophisticated grammar-based fuzzing campaigns based on a contract ABI to falsify user-defined predicates or Solidity assertions.

There were also dynamic tests run on EVM byte code to detect common vulnerabilities including integer underflows, owner-overwrite-to-Ether-withdrawal, and others.

```
The analysis was completed successfully. No issues were detected.
```

The tests were successfully passed.

Manual Checks

The codebase has good deployment scripts done according to industry standard using Truffle solution.

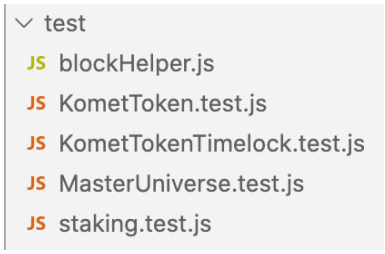
We have successfully deployed the solution into local private chain according to documentation provided.

```
Summary
=======
> Total deployments:   6
> Final cost:          0.17344544 ETH
```

There were no major issues found during manual review.

Automatic Tests

The project has very comprehensive test scripts library using Truffle functionality. They are well written and test all the described functionality.



We have run the test suite during tests and **confirmed all assertions** described in the tests.

We have briefly analysed the tests scripts however the comprehensive analysis was not performed.

```
  Contract: MasterUniverse
    ✓ can be create (141ms)
    ✓ can transfer the ownership to MU (73ms)
    ✓ stakers should have komet  (88ms)
    ✓ stakers should have lp tokens (221ms)
    ✓ can create a new pool (103ms)
    ✓ can receive token from staker (311ms)
    ✓ can get the pending Nova and initial buff rate (125ms)
    ✓ can get the staker buff rate, max buff rate and amount (176ms)
    ✓ the pending Nova evolve after one block mine (136ms)
    ✓ the multiplier will stop after end block (615ms)
    ✓ after 1 day the buffrate should be 50% more (312ms)
    ✓ after 2 days the buffrate should be 50% more again (280ms)
    ✓ after 3 days the buffrate should be 50% more again but capped with the max buffrate (295ms)
    ✓ can have a second staker with and udpated max buff rate (365ms)
{
  novaForStaker1: '0.141630901287553648',
  novaForStaker2: '0.858369098712446351',
  staker1BuffRate: '90',
  staker2BuffRate: '30'
}
    ✓ calculate the pendingNova depending on plasmaPower (577ms)


  15 passing (4s)
```

Deployment & Contract Ownership

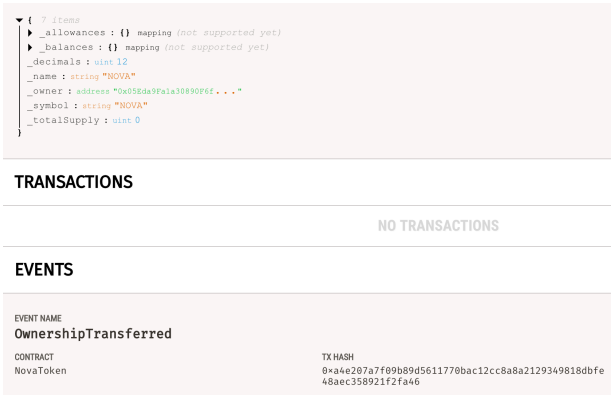
As of Ethereum block #**11360821** only one KometToken is deployed on mainnet at 0x6CfB6dF56BbdB00226AeFfCdb2CD1FE8Da1ABdA7.

The KometToken deployment does not possesses significant risks.

We have use the deployment scripts to deploy on private chain other provided contract to check the ownership structure.

NAME	ADDRESS	TX COUNT	
KometToken	0×D93bE2F84659c7C29B1bCc3A48d59932D876CB85	0	DEPLOYED
KometTokenTimelock	0×a2e7b95E4EEBC9A92fd616A7aF80D6dF2C56C5F6	0	DEPLOYED
LPToken	0×EB975109610aC03F3d40Cd51941B2489f5331745	0	DEPLOYED
MasterUniverse	0×3692A87F047c4cCB9CB32D4793C461b5Fa82A9Ef	0	DEPLOYED
Migrations	0×c03755c955E90a35380918e9a52CBE081eD79467	3	DEPLOYED
NovaToken	0×Dcf3FE8Ef72821877434aE3CDB974EEE9Af16D63	0	DEPLOYED

Ownership of Nova token is set



correctly:

There are no significant risks in the ownership context as far as the mainet deployment will be done via scripts provided.

MasterUniverse is owned by ordinary address. That may posses some risks as it allows to add new pools and set parameters in them.

Recommendations:

1. Deploy a **governance** on top of the ownership of MasterUniverse. It could be a time lock or full Compound like governance system - https://medium.com/compound-finance/compound-governance-5531f524cf68

Disclaimer

The information appearing in this report is for general purposes only and is not intended to provide any legal security guarantees to any individual or entity. As one review is not enough to provide 100% security against any attacks or bugs, it is advisable to conduct more reviews.

The report does not provide personalised investment advice or recommendations, especially does not provide advice to conclude any transactions and it does not provide investment, financial, legal or tax advice.

We are not responsible or liable for any loss which results from the report.

The report should not be considered as an investment advice.