Tower Finance

Smart Contract Audit (Final Report)

February 11th 2022

For:
Prepared By: Entersoft Pte Ltd of 1B Trengganu Street, Singapore 058455

Contents

Revision History and Version Control

Version	Date	Author(s)	Description
1.0	11th February 2022	Abhishek Sharma	Initial Draft of Final Report
1.0	11th February 2022	Jake Lemke	Reviewed
1.0	11th February 2022	Jake Lemke	Released Final Report

Entersoft was commissioned to perform a source code review on their solidity smart contract.

The review was conducted between January 27th to February 7th, 2021. The report is organized into the following sections.

- Executive Summary: A high-level overview of the security audit findings.
- Technical analysis: Our detailed analysis of the Smart Contract code

The information in this report should be used to understand overall code quality, security, correctness, and meaning that code will work as described in the smart contract.

1.0 Disclaimer

This is a limited audit report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to: (i) smart contract best coding practices and issues in the framework and algorithms based on white paper, code, the details of which are set out in this report, (Smart Contract audit). To get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us based on what it says or does not say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full. DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Entersoft Australia and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (Entersoft) owe no duty of care towards you or any other person, nor does Entersoft make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Entersoft hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Entersoft hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Entersoft, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the Smart contract is purely based on the smart contract code shared with us alone.

2.0 Overview

2.1 Project Overview

During the period of **January 27th, 2021, to February 11th, 2021**– Entersoft performed smart contract security audits for **Tower Finance.**

2.2 Scope

The scope of this audit was to analyse and document the smart contract codebase for quality, security, and correctness.

OUT-OF-SCOPE: External contracts, External Oracles, other smart contracts in the repository or imported smart contracts.

2.3 Project Summary

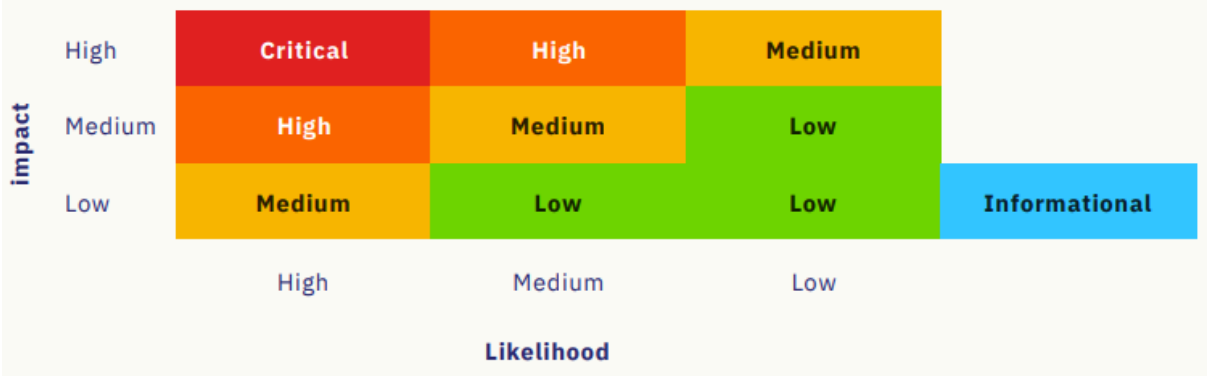
Project Name	Tower Finance
Codebase	https://github.com/towerfinance/tower-v2-contracts-audit
Verified	Yes
Audited	Yes
Vulnerabilities / Issues	As per report. Section 2.6

2.4 Audit Summary

Delivery Date	11th of February 2021
Method of Audit	Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis, Theo.
Consultants Engaged	2

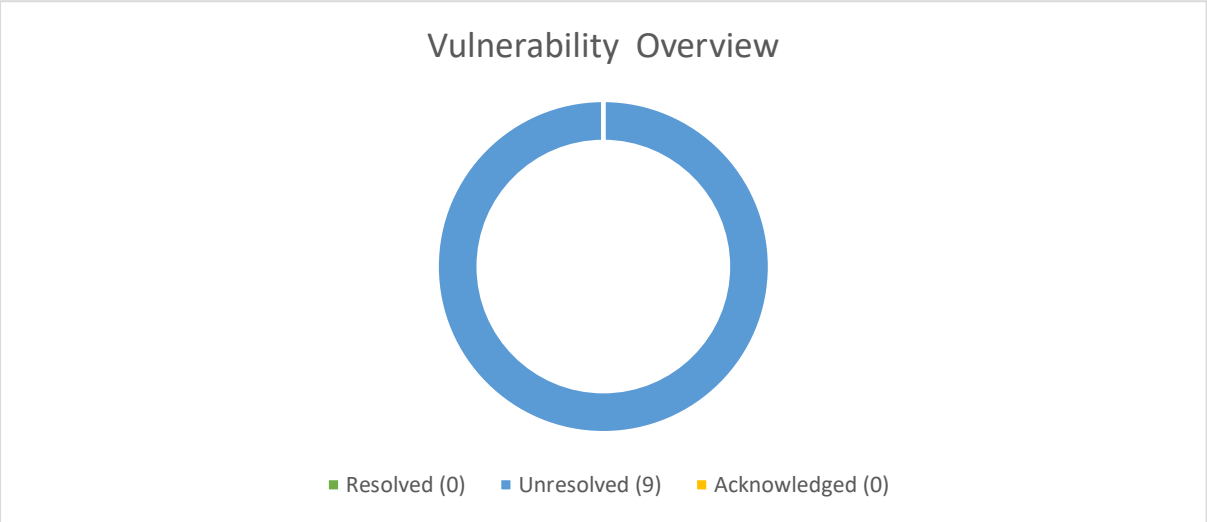
2.5 Security Level references

Every issue in this report was assigned a severity level from the following classification table:



2.6 Vulnerability Summary

Total Critical	0
Total High	0
Total Medium	6
Total Low	0
Total Informational	3



2.7 Audit Results Overview

Audit Item	Audit Subclass	Audit Result
Overflow	-	Passed
Race Conditions	-	Passed
Permissions	Permission Vulnerability Audit Excessive Auditing Authority	Passed
Safety Design	Zeppelin Safe Math	Passed
DDOS Attack	Call Function Security	Passed
Gas Optimization	-	Passed
Design Logic	-	Passed
Know Attacks	-	Passed
Overall Audit Result	-	Passed

3.0 Executive Summary

3.1 Findings

TWR-001	State variable shadow	Informational	**RESOLVED**
TWR-002	Dead Code	Informational	**RESOLVED**
TWR-003	Function does not implement	Informational	**RESOLVED**
TRW-004	Unchecked transfer	Medium	**RESOLVED**
TWR-005	Missing Zero Address Validation	Medium	**RESOLVED**
TWR-006	Event should be emitted before external call/ Re-entrancy Attack	Medium	**RESOLVED**
TWR-007	Re-entrancy Possibility	Medium	**RESOLVED**
TWR-008	Divide before multiply	Medium	**RESOLVED**
TWR-009	Unused return	Medium	**RESOLVED**

3.2 Comments

Overall, the smart contracts are very well written and they adhere to best security practices and industry guidelines.

No instances of Integer Overflow and Underflow vulnerabilities or Back-Door Entry were identified in the contract during the audit. However, relying on other contracts might cause the Re-entrancy Vulnerability.

Entersoft identified the smart contract has multiple owners. Having more than one owner for a smart contract is fine and very common practice. In general, smart contracts will have many functions, out of which some are business-critical functions that need to be restricted from public users, and some functions that need to be publicly accessible.

To control access to the business-critical functionalities, modifiers are used in the smart contract code.

Rather than delegating all the business-critical functions to a single address/owner, they can be split and assigned to two different roles (owner and operator) which in turn reduces the risk.

At the same time, the private keys should always be isolated by storing them on a hardware wallet like Ledger, and should only be used when signing the transactions.

Additionally, a number of medium and informational severity issues were identified during the audit. Entersoft recommended that these are fixed.

4.0 Vulnerabilities

4.1 Tower Smart Contract

4.1.1 State variable shadow

Severity	Confidence	Status
Informational	High	Resolved

Description:

Detection of state variables shadowed.

TowerERC20.constructor(string,string,address)._name (contracts/ERC20/TowerERC20.sol#32) shadows:
- ERC20._name (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#42) (state variable)
TowerERC20.constructor(string,string,address)._symbol (contracts/ERC20/TowerERC20.sol#33) shadows:
- ERC20._symbol (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#43) (state variable)

Remediation:
Remove the state variable shadowing.

4.1.2 Dead-code

Severity	Confidence	Status
Informational	High	Resolved

Description:

Functions that are not sued.

Context.msgData Context.sol (contracts/libraries/Context.sol#18-20) is never used and should be removed

Remediation:
Remove unused functions.

4.1.3 Function does not implement

Severity	Confidence	Status
Informational	High	Resolved

Description:

Functions that are not implemented on most-derived contracts.

```
function mintByFarm(address _to, uint256 _amt) public virtual override;
```

Remediation:
Implement all unimplemented functions in any contract you intend to use directly (not simply inherit from).

4.2 CubeStake

4.2.1 Unchecked transfer

Severity	Confidence	Status
Medium	High	Resolved

Description:

The return value of an external transfer/transferFrom call is not checked

```
    cube.transferFrom(msg.sender, address(this), _amount);
```

Remediation:
Use SafeERC20, or ensure that the transfer/transferFrom return value is checked.

4.2.2 Missing Zero Address validation

Severity	Confidence	Status
Medium	High	Resolved

Description:

missing zero address validation.

```
function setOperator(address _operator) public onlyOwner {
    operator = _operator;
    emit OperatorUpdated(operator);
}
```

Location: Common/towerProtocol.sol

Remediation:
Check that the address is not zero.

4.2.3 Event should be emitted before external call/ Re-entrancy Attack

Severity	Confidence	Status
Medium	High	Resolved

Description:

Event emitted after external call

```
        cube.transferFrom(profitController, address(this), _amount);
    emit ProfitDistributed(_amount);
```

Remediation:
Emit event before last external call

4.3 Farm

4.3.1 Re-entrancy Possibility

Severity	Confidence	Status
Medium	High	Resolved

Description:

Farm._calcAccCubeToAdd(uint256) (contracts/farm/Farm.sol#211-223) has external calls inside a loop: _lpSupply = lpToken[_pid].balanceOf(address(this)) (contracts/farm/Farm.sol#213)

Farm._calcAccCubeToAdd(uint256) (contracts/farm/Farm.sol#211-223) has external calls inside a loop: _lpDecimals = ERC20(address(lpToken[_pid])).decimals() (contracts/farm/Farm.sol#214)

```
    for (uint256 i = _user.vestings.length; i < VESTING_COUNT; i++) {
        _user.vestings.push(VestingInfo({startTime: 0, amount: 0}));
    }
```

Remediation:
Check safe call inside a loop

4.3.2 Divide before multiply

Severity	Confidence	Status
Medium	High	Resolved

Description:

Solidity integer division might truncate. As a result, performing multiplication before division can sometimes avoid loss of precision.

```
    uint256 _oneThird = RATIO_PRECISION / 3;

    uint256 _treasuryAmt = (_amount * 2 * _oneThird) / RATIO_PRECISION;
```

Remediation:
Consider ordering multiplication before division

4.4 Bank

4.4.1 Unused return

Severity	Confidence	Status
Medium	High	Resolved

Description:

The return value of an external call is not stored in a local or state variable.

```
        cube.safeTransferFrom(msg.sender, address(this), _requiredCubeAmt);
        cube.approve(address(swapController), 0);
        cube.approve(address(swapController), _requiredCubeAmt);
```

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

5.0 Tower Finance Functional Tests

The following is the list of functions tested and checked for vulnerabilities during audit:

Function Name()	Technical Result	Logical Result	Overall Result
safeWithdrawBnb	Pass	Pass	Pass
approve	Pass	Pass	Pass
allowance	Pass	Pass	Pass
transfer	Pass	Pass	Pass
transferFrom	Pass	Pass	Pass

6.0 Automated Testing

Automated testing is carried out with the following tools:

- Slither
- Mythril
- Echidna
- Manticore

6.1 Slither

Slither is an open-source Solidity static analysis framework. This tool provides rich information about Ethereum smart contracts while ensuring all critical properties. It runs a suite of vulnerability detectors, prints visual information about contract details, and provides an API to easily write custom analyses.

No major issues were found. Some false positive errors were reported by the tool. All the other issues have been categorized above according to their level of severity.

ERC20

```
TowerERC20.constructor(string,string,address)._name (ERC20/TowerERC20.sol#70) shadows:
        - ERC20._name (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#42) (state variable)
TowerERC20.constructor(string,string,address)._symbol (ERC20/TowerERC20.sol#71) shadows:
        - ERC20._symbol (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#43) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Different versions of Solidity is used:
        - Version used: ['0.8.4', '^0.8.0']
        - 0.8.4 (ERC20/TowerERC20.sol#2)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter OnlyBank.setBank(address)._bank (ERC20/TowerERC20.sol#41) is not in mixedCase
Parameter TowerERC20.burn(address,uint256)._sender (ERC20/TowerERC20.sol#77) is not in mixedCase
Parameter TowerERC20.burn(address,uint256)._amt (ERC20/TowerERC20.sol#77) is not in mixedCase
Parameter TowerERC20.mintByBank(address,uint256)._to (ERC20/TowerERC20.sol#86) is not in mixedCase
Parameter TowerERC20.mintByBank(address,uint256)._amt (ERC20/TowerERC20.sol#86) is not in mixedCase
Parameter TowerERC20.addBurnAddress(address)._burnAddress (ERC20/TowerERC20.sol#93) is not in mixedCase
Parameter TowerERC20.removeBurnAddress(address)._burnAddress (ERC20/TowerERC20.sol#98) is not in mixedCase
Parameter TowerERC20.addFarm(address)._farm (ERC20/TowerERC20.sol#103) is not in mixedCase
Parameter TowerERC20.removeFarm(address)._farm (ERC20/TowerERC20.sol#108) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

TowerERC20 (ERC20/TowerERC20.sol#48-112) does not implement functions:
```

```
TowerERC20 (ERC20/TowerERC20.sol#48-112) does not implement functions:
        - TowerERC20.mintByFarm(address,uint256) (ERC20/TowerERC20.sol#91)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

mintByBank(address,uint256) should be declared external:
        - TowerERC20.mintByBank(address,uint256) (ERC20/TowerERC20.sol#86-89)
mintByFarm(address,uint256) should be declared external:
        - TowerERC20.mintByFarm(address,uint256) (ERC20/TowerERC20.sol#91)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#62-65)
name() should be declared external:
        - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#62-64)
symbol() should be declared external:
        - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#70-72)
decimals() should be declared external:
        - ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#87-89)
totalSupply() should be declared external:
        - ERC20.totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#94-96)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#101-103)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#113-116)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#121-123)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#132-135)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#150-162)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#176-179)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#195-203)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
ERC20/TowerERC20.sol analyzed (8 contracts with 77 detectors), 34 result(s) found
```

Cube Stake

```
CubeStake.stake(uint256) (stake/CubeStake.sol#96-120) ignores return value by cube.transferFrom(msg.sender,address(this),_amount) (stak
CubeStake.unstake(uint256) (stake/CubeStake.sol#122-141) ignores return value by cube.transfer(msg.sender,cubeAmount) (stake/CubeStake
CubeStake.distribute(uint256) (stake/CubeStake.sol#173-190) ignores return value by cube.transferFrom(profitController,address(this),_a
ake.sol#188)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

CubeStake.cubePerShare() (stake/CubeStake.sol#143-157) uses a dangerous strict equality:
        - totalShares == 0 || totalCube == 0 (stake/CubeStake.sol#147)
CubeStake.stake(uint256) (stake/CubeStake.sol#96-120) uses a dangerous strict equality:
        - _totalShares == 0 || _totalCube == 0 (stake/CubeStake.sol#105)
CubeStake.userLockInfo(address) (stake/CubeStake.sol#159-171) uses a dangerous strict equality:
        - unlockTime == 0 (stake/CubeStake.sol#166)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

TowerProtocol.setOperator(address)._operator (common/TowerProtocol.sol#43) lacks a zero-check on :
                - operator = _operator (common/TowerProtocol.sol#44)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in CubeStake.distribute(uint256) (stake/CubeStake.sol#173-190):
        External calls:
        - cube.transferFrom(profitController,address(this),_amount) (stake/CubeStake.sol#188)
        Event emitted after the call(s):
        - ProfitDistributed(_amount) (stake/CubeStake.sol#189)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

CubeStake.calcAccAmt() (stake/CubeStake.sol#65-73) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp < epoch.lastRewardTime (stake/CubeStake.sol#66)
CubeStake.updateStakeLock() (stake/CubeStake.sol#75-83) uses timestamp for comparisons
        Dangerous comparisons:
        - epoch.lockAmount > accAmt (stake/CubeStake.sol#77)
CubeStake.unstake(uint256) (stake/CubeStake.sol#122-141) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(userInfo.unlockTime <= block.timestamp,Stake: Not expired) (stake/CubeStake.sol#127)
CubeStake.userLockInfo(address) (stake/CubeStake.sol#159-171) uses timestamp for comparisons
        Dangerous comparisons:
```

```
CubeStake.userLockInfo(address) (stake/CubeStake.sol#159-171) uses timestamp for comparisons
        Dangerous comparisons:
        - unlockTime == 0 (stake/CubeStake.sol#166)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#27-37) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#33-35)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#196-216) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#208-211)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.8.4', '^0.8.0']
        - 0.8.4 (common/TowerProtocol.sol#2)
        - 0.8.4 (interfaces/ICubeStake.sol#2)
        - 0.8.4 (interfaces/IProfitController.sol#2)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/proxy/utils/Initializable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#4)
        - 0.8.4 (stake/CubeStake.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#80-82) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#109-115) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#169-171) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#179-188) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#142-144) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#152-161) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#55-60) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
```

```
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#55-60) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
SafeCast.toInt128(int256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#152-155) is never used and should be removed
SafeCast.toInt16(int256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#206-209) is never used and should be removed
SafeCast.toInt256(uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#236-240) is never used and should be removed
SafeCast.toInt32(int256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#188-191) is never used and should be removed
SafeCast.toInt64(int256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#170-173) is never used and should be removed
SafeCast.toInt8(int256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#224-227) is never used and should be removed
SafeCast.toUint128(uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#47-50) is never used and should be removed
SafeCast.toUint16(uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#107-110) is never used and should be removed
SafeCast.toUint224(uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#32-35) is never used and should be removed
SafeCast.toUint256(int256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#134-137) is never used and should be removed
SafeCast.toUint32(uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#92-95) is never used and should be removed
SafeCast.toUint64(uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#77-80) is never used and should be removed
SafeCast.toUint8(uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#122-125) is never used and should be removed
SafeCast.toUint96(uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#62-65) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#45-58) is never used and should
ved
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#69-80) is never used a
ld be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#60-67) is never used a
ld be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#29-36) is never use
hould be removed
SafeMath.div(uint256,uint256,string) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#191-200) is never used and should be removed
SafeMath.mod(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#151-153) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#217-226) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#168-177) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#22-28) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#64-69) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#76-81) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#47-57) is never used and should be removed
SafeMath.trySub(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#35-40) is never used and should be removed
TowerProtocol._currentBlockTs() (common/TowerProtocol.sol#48-50) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/proxy/utils/Initializable.sol#4) allows old versions
```

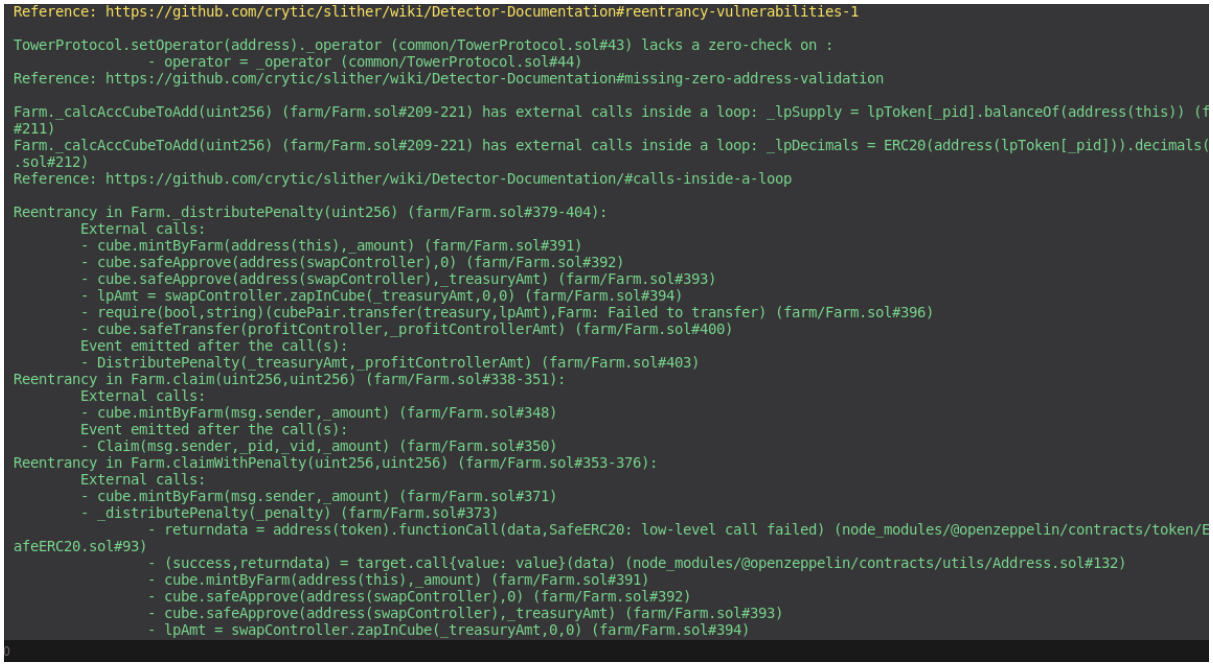
```
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#4) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#55-60):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#58)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#1
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#132)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#152-161):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#159)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#179-188):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#186)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter TowerProtocol.setOperator(address)._operator (common/TowerProtocol.sol#43) is not in mixedCase
Parameter CubeStake.init(address)._profitController (stake/CubeStake.sol#61) is not in mixedCase
Parameter CubeStake.pendingxCube(address)._user (stake/CubeStake.sol#85) is not in mixedCase
Parameter CubeStake.stake(uint256)._amount (stake/CubeStake.sol#96) is not in mixedCase
Parameter CubeStake.unstake(uint256)._share (stake/CubeStake.sol#122) is not in mixedCase
Parameter CubeStake.userLockInfo(address)._user (stake/CubeStake.sol#159) is not in mixedCase
Parameter CubeStake.distribute(uint256)._amount (stake/CubeStake.sol#173) is not in mixedCase
Parameter CubeStake.transferTo(address,address,uint256)._receiver (stake/CubeStake.sol#198) is not in mixedCase
Parameter CubeStake.transferTo(address,address,uint256)._token (stake/CubeStake.sol#199) is not in mixedCase
Parameter CubeStake.transferTo(address,address,uint256)._amount (stake/CubeStake.sol#200) is not in mixedCase
Parameter CubeStake.setProfitController(address)._profitController (stake/CubeStake.sol#211) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

TowerProtocol.RATIO_PRECISION (common/TowerProtocol.sol#9) is never used in CubeStake (stake/CubeStake.sol#13-216)
TowerProtocol.PRICE_PRECISION (common/TowerProtocol.sol#10) is never used in CubeStake (stake/CubeStake.sol#13-216)
```

```
TowerProtocol.RATIO_PRECISION (common/TowerProtocol.sol#9) is never used in CubeStake (stake/CubeStake.sol#13-216)
TowerProtocol.PRICE_PRECISION (common/TowerProtocol.sol#10) is never used in CubeStake (stake/CubeStake.sol#13-216)
TowerProtocol.USDC_PRECISION (common/TowerProtocol.sol#11) is never used in CubeStake (stake/CubeStake.sol#13-216)
TowerProtocol.MISSING_PRECISION (common/TowerProtocol.sol#12) is never used in CubeStake (stake/CubeStake.sol#13-216)
TowerProtocol.TOWER_PRECISION (common/TowerProtocol.sol#13) is never used in CubeStake (stake/CubeStake.sol#13-216)
TowerProtocol.CUBE_PRECISION (common/TowerProtocol.sol#14) is never used in CubeStake (stake/CubeStake.sol#13-216)
TowerProtocol.SWAP_FEE_PRECISION (common/TowerProtocol.sol#15) is never used in CubeStake (stake/CubeStake.sol#13-216)
TowerProtocol.ADDRESS_USDC (common/TowerProtocol.sol#17-18) is never used in CubeStake (stake/CubeStake.sol#13-216)
TowerProtocol.ADDRESS_WMATIC (common/TowerProtocol.sol#19-20) is never used in CubeStake (stake/CubeStake.sol#13-216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#62-65)
name() should be declared external:
        - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#62-64)
symbol() should be declared external:
        - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#70-72)
decimals() should be declared external:
        - ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#87-89)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#113-116)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#121-123)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#132-135)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#150-162)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#176-179)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#195-203)
pendingxCube(address) should be declared external:
        - CubeStake.pendingxCube(address) (stake/CubeStake.sol#85-94)
stake(uint256) should be declared external:
        - CubeStake.stake(uint256) (stake/CubeStake.sol#96-120)
```

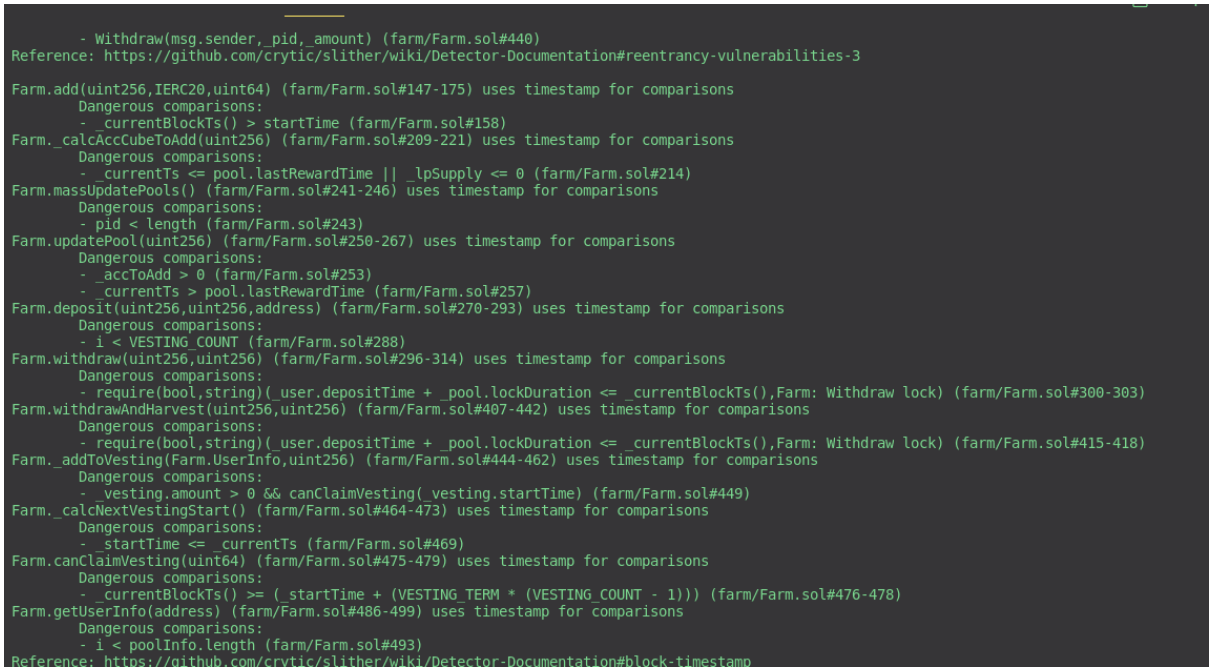
```
TowerProtocol.ADDRESS_WMATIC (common/TowerProtocol.sol#19-20) is never used in CubeStake (stake/CubeStake.sol#13-216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#62-65)
name() should be declared external:
        - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#62-64)
symbol() should be declared external:
        - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#70-72)
decimals() should be declared external:
        - ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#87-89)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#113-116)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#121-123)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#132-135)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#150-162)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#176-179)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#195-203)
pendingxCube(address) should be declared external:
        - CubeStake.pendingxCube(address) (stake/CubeStake.sol#85-94)
stake(uint256) should be declared external:
        - CubeStake.stake(uint256) (stake/CubeStake.sol#96-120)
unstake(uint256) should be declared external:
        - CubeStake.unstake(uint256) (stake/CubeStake.sol#122-141)
setStakePaused() should be declared external:
        - CubeStake.setStakePaused() (stake/CubeStake.sol#192-195)
transferTo(address,address,uint256) should be declared external:
        - CubeStake.transferTo(address,address,uint256) (stake/CubeStake.sol#197-209)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
stake/CubeStake.sol analyzed (15 contracts with 77 detectors), 102 result(s) found
```

Cube Farm

```
dev-varun@power-station:~/Documents/tower-v2-contracts-audit-main$ slither farm/Farm.sol
Compilation warnings/errors on farm/Farm.sol:
Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SP
se>" to each source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.
--> node_modules/@uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol

Warning: Contract code size exceeds 24576 bytes (a limit introduced in Spurious Dragon). This contract may not be deployable on mainnet. Consider e
the optimizer (with a low "runs" value!), turning off revert strings, or using libraries.
  --> farm/Farm.sol:15:1:
   |
15 | contract Farm is IFarm, TowerProtocol, EnableSwap {
   | ^ (Relevant source part starts here and spans across multiple lines).



Farm._calcNextVestingStart() (farm/Farm.sol#464-473) uses a weak PRNG: "_startTime = (_currentTs) - (_currentTs % VESTING_TERM) + GENESIS_TIME (far
ol#466-468)"
Farm.vestingSlot() (farm/Farm.sol#481-484) uses a weak PRNG: "uint8((_vestringStart / VESTING_TERM) % VESTING_COUNT) (farm/Farm.sol#483)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Farm._calcAccCubeToAdd(uint256) (farm/Farm.sol#209-221) performs a multiplication on the result of a division:
        -_cubeReward = (_time * cubePerSecond * pool.allocPoint) / totalAllocPoint (farm/Farm.sol#218-219)
        -(_cubeReward * (10 ** _lpDecimals)) / _lpSupply (farm/Farm.sol#220)
Farm._distributePenalty(uint256) (farm/Farm.sol#379-404) performs a multiplication on the result of a division:
        -_oneThird = RATIO_PRECISION / 3 (farm/Farm.sol#387)
        -_treasuryAmt = (_amount * 2 * _oneThird) / RATIO_PRECISION (farm/Farm.sol#389)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Reentrancy in Farm.deposit(uint256,uint256,address) (farm/Farm.sol#270-293):
        External calls:
        - lpToken[_pid].safeTransferFrom(msg.sender,address(this),_amount) (farm/Farm.sol#278)
        State variables written after the call(s):
        - _user.pid = _pid (farm/Farm.sol#281)
        - _user.amount += _amount (farm/Farm.sol#282)
        - _user.rewardDebt += int256((_amount * _pool.accCubePerShare) / CUBE_PRECISION) (farm/Farm.sol#283-285)
        - _user.depositTime = _currentBlockTs() (farm/Farm.sol#286)
        - _user.vestings.push(VestingInfo(0,0)) (farm/Farm.sol#289)
```



```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

TowerProtocol.setOperator(address)._operator (common/TowerProtocol.sol#43) lacks a zero-check on :
                - operator = _operator (common/TowerProtocol.sol#44)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Farm._calcAccCubeToAdd(uint256) (farm/Farm.sol#209-221) has external calls inside a loop: _lpSupply = lpToken[_pid].balanceOf(address(this)) (f
#211)
Farm._calcAccCubeToAdd(uint256) (farm/Farm.sol#209-221) has external calls inside a loop: _lpDecimals = ERC20(address(lpToken[_pid])).decimals(
.sol#212)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in Farm._distributePenalty(uint256) (farm/Farm.sol#379-404):
        External calls:
        - cube.mintByFarm(address(this),_amount) (farm/Farm.sol#391)
        - cube.safeApprove(address(swapController),0) (farm/Farm.sol#392)
        - cube.safeApprove(address(swapController),_treasuryAmt) (farm/Farm.sol#393)
        - lpAmt = swapController.zapInCube(_treasuryAmt,0,0) (farm/Farm.sol#394)
        - require(bool,string)(cubePair.transfer(treasury,lpAmt),Farm: Failed to transfer) (farm/Farm.sol#396)
        - cube.safeTransfer(profitController,_profitControllerAmt) (farm/Farm.sol#400)
        Event emitted after the call(s):
        - DistributePenalty(_treasuryAmt,_profitControllerAmt) (farm/Farm.sol#403)
Reentrancy in Farm.claim(uint256,uint256) (farm/Farm.sol#338-351):
        External calls:
        - cube.mintByFarm(msg.sender,_amount) (farm/Farm.sol#348)
        Event emitted after the call(s):
        - Claim(msg.sender,_pid,_vid,_amount) (farm/Farm.sol#350)
Reentrancy in Farm.claimWithPenalty(uint256,uint256) (farm/Farm.sol#353-376):
        External calls:
        - cube.mintByFarm(msg.sender,_amount) (farm/Farm.sol#371)
        - _distributePenalty(_penalty) (farm/Farm.sol#373)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/E
afeERC20.sol#93)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#132)
                - cube.mintByFarm(address(this),_amount) (farm/Farm.sol#391)
                - cube.safeApprove(address(swapController),0) (farm/Farm.sol#392)
                - cube.safeApprove(address(swapController),_treasuryAmt) (farm/Farm.sol#393)
                - lpAmt = swapController.zapInCube(_treasuryAmt,0,0) (farm/Farm.sol#394)
)
```



```
        - Withdraw(msg.sender,_pid,_amount) (farm/Farm.sol#440)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Farm.add(uint256,IERC20,uint64) (farm/Farm.sol#147-175) uses timestamp for comparisons
        Dangerous comparisons:
        - _currentBlockTs() > startTime (farm/Farm.sol#158)
Farm._calcAccCubeToAdd(uint256) (farm/Farm.sol#209-221) uses timestamp for comparisons
        Dangerous comparisons:
        - _currentTs <= pool.lastRewardTime || _lpSupply <= 0 (farm/Farm.sol#214)
Farm.massUpdatePools() (farm/Farm.sol#241-246) uses timestamp for comparisons
        Dangerous comparisons:
        - pid < length (farm/Farm.sol#243)
Farm.updatePool(uint256) (farm/Farm.sol#250-267) uses timestamp for comparisons
        Dangerous comparisons:
        - _accToAdd > 0 (farm/Farm.sol#253)
        - _currentTs > pool.lastRewardTime (farm/Farm.sol#257)
Farm.deposit(uint256,uint256,address) (farm/Farm.sol#270-293) uses timestamp for comparisons
        Dangerous comparisons:
        - i < VESTING_COUNT (farm/Farm.sol#288)
Farm.withdraw(uint256,uint256) (farm/Farm.sol#296-314) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_user.depositTime + _pool.lockDuration <= _currentBlockTs(),Farm: Withdraw lock) (farm/Farm.sol#300-303)
Farm.withdrawAndHarvest(uint256,uint256) (farm/Farm.sol#407-442) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_user.depositTime + _pool.lockDuration <= _currentBlockTs(),Farm: Withdraw lock) (farm/Farm.sol#415-418)
Farm._addToVesting(Farm.UserInfo,uint256) (farm/Farm.sol#444-462) uses timestamp for comparisons
        Dangerous comparisons:
        - _vesting.amount > 0 && canClaimVesting(_vesting.startTime) (farm/Farm.sol#449)
Farm._calcNextVestingStart() (farm/Farm.sol#464-473) uses timestamp for comparisons
        Dangerous comparisons:
        - _startTime <= _currentTs (farm/Farm.sol#469)
Farm.canClaimVesting(uint64) (farm/Farm.sol#475-479) uses timestamp for comparisons
        Dangerous comparisons:
        - _currentBlockTs() >= (_startTime + (VESTING_TERM * (VESTING_COUNT - 1))) (farm/Farm.sol#476-478)
Farm.getUserInfo(address) (farm/Farm.sol#486-499) uses timestamp for comparisons
        Dangerous comparisons:
        - i < poolInfo.length (farm/Farm.sol#493)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#196-216) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#208-211)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Farm.add(uint256,IERC20,uint64) (farm/Farm.sol#147-175) compares to a boolean constant:
        -require(bool,string)(addedTokens[address(_lpToken)] == false,Token already added) (farm/Farm.sol#152)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity is used:
        - Version used: ['0.8.4', '>=0.5.0', '^0.8.0']
        - 0.8.4 (common/EnableSwap.sol#2)
        - 0.8.4 (common/TowerProtocol.sol#2)
        - 0.8.4 (farm/Farm.sol#2)
        - 0.8.4 (interfaces/IFarm.sol#2)
        - 0.8.4 (interfaces/ISwapController.sol#2)
        - 0.8.4 (interfaces/ITowerERC20.sol#2)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/proxy/utils/Initializable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#4)
        - >=0.5.0 (node_modules/@uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#80-82) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#109-115) is never used and should be r
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#169-171) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#179-188) is never used and should be rem
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#142-144) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#152-161) is never used and should be remov
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#55-60) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
ERC20._burn(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#273-288) is never used and should be removed
```

```
ERC20._mint(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#250-260) is never used and should be removed
Initializable._isConstructor() (node_modules/@openzeppelin/contracts/proxy/utils/Initializable.sol#77-79) is never used and should be removed
SafeCast.toInt128(int256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#152-155) is never used and should be removed
SafeCast.toInt16(int256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#206-209) is never used and should be removed
SafeCast.toInt256(uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#236-240) is never used and should be removed
SafeCast.toInt32(int256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#188-191) is never used and should be removed
SafeCast.toInt64(int256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#170-173) is never used and should be removed
SafeCast.toInt8(int256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#224-227) is never used and should be removed
SafeCast.toUint128(uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#47-50) is never used and should be removed
SafeCast.toUint16(uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#107-110) is never used and should be removed
SafeCast.toUint224(uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#32-35) is never used and should be removed
SafeCast.toUint32(uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#92-95) is never used and should be removed
SafeCast.toUint8(uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#122-125) is never used and should be removed
SafeCast.toUint96(uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#62-65) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#69-80) is never used and s
ld be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#60-67) is never used and s
ld be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#4) allows old versions
Pragma version>=0.5.0 (node_modules/@uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#55-60):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#58)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#123-134):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#132)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#152-161):
```

```
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#186)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter EnableSwap.setSwapController(address)._swapController (common/EnableSwap.sol#13) is not in mixedCase
Parameter TowerProtocol.setOperator(address)._operator (common/TowerProtocol.sol#43) is not in mixedCase
Parameter Farm.add(uint256,IERC20,uint64)._allocPoint (farm/Farm.sol#148) is not in mixedCase
Parameter Farm.add(uint256,IERC20,uint64)._lpToken (farm/Farm.sol#149) is not in mixedCase
Parameter Farm.add(uint256,IERC20,uint64)._lockDuration (farm/Farm.sol#150) is not in mixedCase
Parameter Farm.set(uint256,uint256,uint64)._pid (farm/Farm.sol#179) is not in mixedCase
Parameter Farm.set(uint256,uint256,uint64)._allocPoint (farm/Farm.sol#180) is not in mixedCase
Parameter Farm.set(uint256,uint256,uint64)._lockDuration (farm/Farm.sol#181) is not in mixedCase
Parameter Farm.setCubePerSecond(uint256)._cubePerSecond (farm/Farm.sol#196) is not in mixedCase
Parameter Farm.setVestingPenalty(uint256)._penalty (farm/Farm.sol#203) is not in mixedCase
Parameter Farm.pendingCube(uint256,address)._pid (farm/Farm.sol#224) is not in mixedCase
Parameter Farm.pendingCube(uint256,address)._user (farm/Farm.sol#224) is not in mixedCase
Parameter Farm.updatePool(uint256)._pid (farm/Farm.sol#250) is not in mixedCase
Parameter Farm.deposit(uint256,uint256,address)._pid (farm/Farm.sol#271) is not in mixedCase
Parameter Farm.deposit(uint256,uint256,address)._amount (farm/Farm.sol#272) is not in mixedCase
Parameter Farm.deposit(uint256,uint256,address)._to (farm/Farm.sol#273) is not in mixedCase
Parameter Farm.withdraw(uint256,uint256)._pid (farm/Farm.sol#296) is not in mixedCase
Parameter Farm.withdraw(uint256,uint256)._amount (farm/Farm.sol#296) is not in mixedCase
Parameter Farm.harvest(uint256)._pid (farm/Farm.sol#317) is not in mixedCase
Parameter Farm.claim(uint256,uint256)._pid (farm/Farm.sol#338) is not in mixedCase
Parameter Farm.claim(uint256,uint256)._vid (farm/Farm.sol#338) is not in mixedCase
Parameter Farm.claimWithPenalty(uint256,uint256)._pid (farm/Farm.sol#353) is not in mixedCase
Parameter Farm.claimWithPenalty(uint256,uint256)._vid (farm/Farm.sol#353) is not in mixedCase
Parameter Farm.withdrawAndHarvest(uint256,uint256)._pid (farm/Farm.sol#407) is not in mixedCase
Parameter Farm.withdrawAndHarvest(uint256,uint256)._amount (farm/Farm.sol#407) is not in mixedCase
Parameter Farm.canClaimVesting(uint64)._startTime (farm/Farm.sol#475) is not in mixedCase
Parameter Farm.getUserInfo(address)._user (farm/Farm.sol#486) is not in mixedCase
Parameter Farm.getLpToken(uint256)._pid (farm/Farm.sol#501) is not in mixedCase
Parameter Farm.setProfitController(address)._profitController (farm/Farm.sol#505) is not in mixedCase
Parameter Farm.setCube(address,address)._cube (farm/Farm.sol#511) is not in mixedCase
Parameter Farm.setCube(address,address)._cubePair (farm/Farm.sol#511) is not in mixedCase
Parameter Farm.setBankSafe(address)._bankSafe (farm/Farm.sol#519) is not in mixedCase
Parameter Farm.setTreasury(address)._treasury (farm/Farm.sol#525) is not in mixedCase
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (node_modules/@uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol#20) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (node_modules/@uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol#21) is not in mixedCase
0
```

```
TowerProtocol.USDC_PRECISION (common/TowerProtocol.sol#11) is never used in Farm (farm/Farm.sol#15-530)
TowerProtocol.MISSING_PRECISION (common/TowerProtocol.sol#12) is never used in Farm (farm/Farm.sol#15-530)
TowerProtocol.TOWER_PRECISION (common/TowerProtocol.sol#13) is never used in Farm (farm/Farm.sol#15-530)
TowerProtocol.SWAP_FEE_PRECISION (common/TowerProtocol.sol#15) is never used in Farm (farm/Farm.sol#15-530)
TowerProtocol.ADDRESS_USDC (common/TowerProtocol.sol#17-18) is never used in Farm (farm/Farm.sol#15-530)
TowerProtocol.ADDRESS_WMATIC (common/TowerProtocol.sol#19-20) is never used in Farm (farm/Farm.sol#15-530)
Farm.ONE_DAY (farm/Farm.sol#39) is never used in Farm (farm/Farm.sol#15-530)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

poolLength() should be declared external:
        - Farm.poolLength() (farm/Farm.sol#141-143)
add(uint256,IERC20,uint64) should be declared external:
        - Farm.add(uint256,IERC20,uint64) (farm/Farm.sol#147-175)
set(uint256,uint256,uint64) should be declared external:
        - Farm.set(uint256,uint256,uint64) (farm/Farm.sol#178-193)
setCubePerSecond(uint256) should be declared external:
        - Farm.setCubePerSecond(uint256) (farm/Farm.sol#196-201)
deposit(uint256,uint256,address) should be declared external:
        - Farm.deposit(uint256,uint256,address) (farm/Farm.sol#270-293)
withdraw(uint256,uint256) should be declared external:
        - Farm.withdraw(uint256,uint256) (farm/Farm.sol#296-314)
harvest(uint256) should be declared external:
        - Farm.harvest(uint256) (farm/Farm.sol#317-336)
claim(uint256,uint256) should be declared external:
        - Farm.claim(uint256,uint256) (farm/Farm.sol#338-351)
claimWithPenalty(uint256,uint256) should be declared external:
        - Farm.claimWithPenalty(uint256,uint256) (farm/Farm.sol#353-376)
withdrawAndHarvest(uint256,uint256) should be declared external:
        - Farm.withdrawAndHarvest(uint256,uint256) (farm/Farm.sol#407-442)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#62-65)
name() should be declared external:
        - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#62-64)
symbol() should be declared external:
        - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#70-72)
decimals() should be declared external:
```

```
        - Farm.withdraw(uint256,uint256) (farm/Farm.sol#296-314)
harvest(uint256) should be declared external:
        - Farm.harvest(uint256) (farm/Farm.sol#317-336)
claim(uint256,uint256) should be declared external:
        - Farm.claim(uint256,uint256) (farm/Farm.sol#338-351)
claimWithPenalty(uint256,uint256) should be declared external:
        - Farm.claimWithPenalty(uint256,uint256) (farm/Farm.sol#353-376)
withdrawAndHarvest(uint256,uint256) should be declared external:
        - Farm.withdrawAndHarvest(uint256,uint256) (farm/Farm.sol#407-442)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#62-65)
name() should be declared external:
        - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#62-64)
symbol() should be declared external:
        - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#70-72)
decimals() should be declared external:
        - ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#87-89)
totalSupply() should be declared external:
        - ERC20.totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#94-96)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#101-103)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#113-116)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#121-123)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#132-135)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#150-162)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#176-179)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#195-203)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Bank

```
Reentrancy in BankSafe.enterInvest() (bank/BankSafe.sol#114-132):
        External calls:
        - _depositInvest(_investmentAmount) (bank/BankSafe.sol#129)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/u
afeERC20.sol#93)
                - collat.safeApprove(address(aaveLendingPool),0) (bank/BankSafe.sol#173)
                - collat.safeApprove(address(aaveLendingPool),investingAmt) (bank/BankSafe.sol#174)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#132)
                - aaveLendingPool.deposit(address(collat),investingAmt,address(this),0) (bank/BankSafe.sol#175-180)
        External calls sending eth:
        - _depositInvest(_investmentAmount) (bank/BankSafe.sol#129)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#132)
        State variables written after the call(s):
        - isInvestEntered = true (bank/BankSafe.sol#130)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

Bank.calcEcr() (bank/Bank.sol#128-142) performs a multiplication on the result of a division:
        -_totalCollatValueE18 = (totalCollatAmt() * MISSING_PRECISION * oracle.collatPrice()) / PRICE_PRECISION (bank/Bank.sol#132-134)
        -_ecr = (_totalCollatValueE18 * RATIO_PRECISION) / tower.totalSupply() (bank/Bank.sol#136-137)
Bank.mint(uint256,uint256,uint256) (bank/Bank.sol#170-231) performs a multiplication on the result of a division:
        -_collatValueE18 = (_collatIn * MISSING_PRECISION * _collatPrice) / PRICE_PRECISION (bank/Bank.sol#185-187)
        -_towerOut = (_collatValueE18 * RATIO_PRECISION) / tcr (bank/Bank.sol#188)
Bank.mint(uint256,uint256,uint256) (bank/Bank.sol#170-231) performs a multiplication on the result of a division:
```

```
Bank.mint(uint256,uint256,uint256) (bank/Bank.sol#170-231) performs a multiplication on the result of a division:
        -_towerOut = (_collatValueE18 * RATIO_PRECISION) / tcr (bank/Bank.sol#188)
        -_towerFee = (_towerOut * mintFee) / RATIO_PRECISION (bank/Bank.sol#198)
Bank.mint(uint256,uint256,uint256) (bank/Bank.sol#170-231) performs a multiplication on the result of a division:
        -_requiredCubeAmt = ((_towerOut -  _collatValueE18) * PRICE_PRECISION) / _cubePrice (bank/Bank.sol#193-195)
        -_minCollatAmt = (_requiredCubeAmt * _cubePrice * PRICE_PRECISION * (RATIO_PRECISION - zapSlippage)) / RATIO_PRECISION / PRICE_P
rice / 2 / MISSING_PRECISION (bank/Bank.sol#206-214)
Bank.zapMint(uint256,uint256) (bank/Bank.sol#233-297) performs a multiplication on the result of a division:
        -_collatFee = ((_collatIn * mintFee) / RATIO_PRECISION) (bank/Bank.sol#249)
        -_towerFee = (_collatFee * MISSING_PRECISION * _collatPrice) / PRICE_PRECISION (bank/Bank.sol#250-251)
Bank.redeem(uint256,uint256,uint256) (bank/Bank.sol#299-349) performs a multiplication on the result of a division:
        -_collatOut = (_towerToRedeem * PRICE_PRECISION) / oracle.collatPrice() / MISSING_PRECISION (bank/Bank.sol#310-312)
        -_collatOut = (_collatOut * ecr) / RATIO_PRECISION (bank/Bank.sol#318)
Bank.arbRedeem(uint256) (bank/Bank.sol#427-462) performs a multiplication on the result of a division:
        -_collatOut = (_towerIn * PRICE_PRECISION) / oracle.collatPrice() / MISSING_PRECISION (bank/Bank.sol#437-439)
        -_collatOut = (_collatOut * ecr) / RATIO_PRECISION (bank/Bank.sol#445)
Bank.mintTowerByProfit(uint256) (bank/Bank.sol#615-629) performs a multiplication on the result of a division:
        -_available = ((tower.totalSupply() * (ecr - tcr))) / tcr (bank/Bank.sol#619)
        -_available = _available - ((_available * safe.excessCollateralSafetyMargin()) / RATIO_PRECISION) (bank/Bank.sol#621-624)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

BankSafe.calcCollateralReserveRatio() (bank/BankSafe.sol#265-278) uses a dangerous strict equality:
        - _globalCollateralBalance == 0 (bank/BankSafe.sol#272)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in Bank.collect() (bank/Bank.sol#351-371):
        External calls:
        - safe.transferCollatTo(msg.sender,_collatOut) (bank/Bank.sol#363)
        State variables written after the call(s):
        - redeemCubeBal[msg.sender] = 0 (bank/Bank.sol#367)
Reentrancy in Bank.recollateralize(uint256,uint256) (bank/Bank.sol#468-511):
        External calls:
        - collat.safeTransferFrom(msg.sender,address(safe),_collatIn) (bank/Bank.sol#502)
        - cube.mintByBank(msg.sender,_cubeOut) (bank/Bank.sol#503)
        State variables written after the call(s):
        - rctHourlyCum[_curEpochHr()] += _cubeOut (bank/Bank.sol#507)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Bank.mint(uint256,uint256,uint256) (bank/Bank.sol#170-231) ignores return value by cube.approve(address(swapController),0) (bank/Bank.sol#217)
Bank.mint(uint256,uint256,uint256) (bank/Bank.sol#170-231) ignores return value by cube.approve(address(swapController),_requiredCubeAmt) (bank/Bank.sol#
8)
Bank.mint(uint256,uint256,uint256) (bank/Bank.sol#170-231) ignores return value by swapController.zapInCube(_requiredCubeAmt,_minCollatAmt,0) (bank/Bank.
l#219)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Bank.init(address,address,address,address,address,address,address,address,address,address,uint256) (bank/Bank.sol#68-105) should emit an event for:
        - tcr = _tcr (bank/Bank.sol#98)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

OnlyArbitrager.setArbitrager(address)._arbitrager (common/OnlyArbitrager.sol#16) lacks a zero-check on :
                - arbitrager = _arbitrager (common/OnlyArbitrager.sol#17)
TowerProtocol.setOperator(address)._operator (common/TowerProtocol.sol#43) lacks a zero-check on :
                - operator = _operator (common/TowerProtocol.sol#44)
Bank.init(address,address,address,address,address,address,address,address,address,address,uint256)._dustbin (bank/Bank.sol#75) lacks a zero-check on :
                - dustbin = _dustbin (bank/Bank.sol#96)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in BankSafe._withdrawInvest() (bank/BankSafe.sol#184-197):
        External calls:
        - newBalance = aaveLendingPool.withdraw(address(collat),balanceOfAToken(),address(this)) (bank/BankSafe.sol#185-189)
        State variables written after the call(s):
        - investingAmt = 0 (bank/BankSafe.sol#194)
Reentrancy in Bank.collect() (bank/Bank.sol#351-371):
        External calls:
        - safe.transferCollatTo(msg.sender,_collatOut) (bank/Bank.sol#363)
        State variables written after the call(s):
        - unclaimedCube -= _cubeOut (bank/Bank.sol#368)
Reentrancy in BankSafe.exitInvest() (bank/BankSafe.sol#134-143):
        External calls:
        - profit = _withdrawInvest() (bank/BankSafe.sol#141)
                - newBalance = aaveLendingPool.withdraw(address(collat),balanceOfAToken(),address(this)) (bank/BankSafe.sol#185-189)
        State variables written after the call(s):
        - isInvestEntered = false (bank/BankSafe.sol#142)
```

```
    - 0.8.4 (common/OnlyBank.sol#2)
    - 0.8.4 (common/TowerProtocol.sol#2)
    - 0.8.4 (interfaces/IBank.sol#2)
    - 0.8.4 (interfaces/IBankSafe.sol#2)
    - 0.8.4 (interfaces/IPriceOracle.sol#2)
    - 0.8.4 (interfaces/IProfitController.sol#2)
    - 0.8.4 (interfaces/ISwapController.sol#2)
    - 0.8.4 (interfaces/ITowerERC20.sol#2)
    - 0.8.4 (libraries/Babylonian.sol#2)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/proxy/utils/Initializable.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/Math.sol#4)
    - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#4)
    - >=0.5.0 (node_modules/@uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol#3)
erence: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

ress.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#80-82) is never used and should be removed
ress.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#109-115) is never used and should
ress.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#169-171) is never used and should be remove
ress.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#179-188) is never used and should be
ress.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#142-144) is never used and should be removed
ress.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#152-161) is never used and should be
ress.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#55-60) is never used and should be removed
text._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
h.average(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/Math.sol#28-31) is never used and should be removed
h.ceilDiv(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/Math.sol#39-42) is never used and should be removed
eCast.toInt128(int256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#152-155) is never used and should be removed
eCast.toInt16(int256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#206-209) is never used and should be removed
eCast.toInt256(uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#236-240) is never used and should be removed
eCast.toInt32(int256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#188-191) is never used and should be removed
eCast.toInt64(int256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#170-173) is never used and should be removed
eCast.toInt8(int256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#224-227) is never used and should be removed
eCast.toUint128(uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#47-50) is never used and should be removed
```

```
  Function IAaveIncentivesController.REWARD_TOKEN() (aave/IAaveIncentivesController.sol#34) is not in mixedCase
  Parameter Bank.init(address,address,address,address,address,address,address,address,address,address,uint256)._collat (bank/Bank.sol#69) is not in
  Parameter Bank.init(address,address,address,address,address,address,address,address,address,address,uint256)._tower (bank/Bank.sol#70) is not in
  Parameter Bank.init(address,address,address,address,address,address,address,address,address,address,uint256)._cube (bank/Bank.sol#71) is not in m
  Parameter Bank.init(address,address,address,address,address,address,address,address,address,address,uint256)._cubePair (bank/Bank.sol#72) is not
  se
  Parameter Bank.init(address,address,address,address,address,address,address,address,address,address,uint256)._oracle (bank/Bank.sol#73) is not in
  Parameter Bank.init(address,address,address,address,address,address,address,address,address,address,uint256)._safe (bank/Bank.sol#74) is not in m
  Parameter Bank.init(address,address,address,address,address,address,address,address,address,address,uint256)._dustbin (bank/Bank.sol#75) is not i
  e
  Parameter Bank.init(address,address,address,address,address,address,address,address,address,address,uint256)._arbitrager (bank/Bank.sol#76) is no
  Case
  Parameter Bank.init(address,address,address,address,address,address,address,address,address,address,uint256)._profitController (bank/Bank.sol#77)
   mixedCase
  Parameter Bank.init(address,address,address,address,address,address,address,address,address,address,uint256)._swapController (bank/Bank.sol#78) i
  ixedCase
  Parameter Bank.init(address,address,address,address,address,address,address,address,address,address,uint256)._tcr (bank/Bank.sol#79) is not in mi
  Parameter Bank.setContracts(address,address,address,address)._safe (bank/Bank.sol#108) is not in mixedCase
  Parameter Bank.setContracts(address,address,address,address)._dustbin (bank/Bank.sol#109) is not in mixedCase
  Parameter Bank.setContracts(address,address,address,address)._profitController (bank/Bank.sol#110) is not in mixedCase
  Parameter Bank.setContracts(address,address,address,address)._oracle (bank/Bank.sol#111) is not in mixedCase
  Parameter Bank.mint(uint256,uint256,uint256)._collatIn (bank/Bank.sol#171) is not in mixedCase
  Parameter Bank.mint(uint256,uint256,uint256)._cubeIn (bank/Bank.sol#172) is not in mixedCase
  Parameter Bank.mint(uint256,uint256,uint256)._towerOutMin (bank/Bank.sol#173) is not in mixedCase
  Parameter Bank.zapMint(uint256,uint256)._collatIn (bank/Bank.sol#233) is not in mixedCase
  Parameter Bank.zapMint(uint256,uint256)._towerOutMin (bank/Bank.sol#233) is not in mixedCase
  Parameter Bank.redeem(uint256,uint256,uint256)._towerIn (bank/Bank.sol#300) is not in mixedCase
  Parameter Bank.redeem(uint256,uint256,uint256)._cubeOutMin (bank/Bank.sol#301) is not in mixedCase
  Parameter Bank.redeem(uint256,uint256,uint256)._collatOutMin (bank/Bank.sol#302) is not in mixedCase
  Parameter Bank.arbMint(uint256)._collatIn (bank/Bank.sol#373) is not in mixedCase
  Parameter Bank.arbRedeem(uint256)._towerIn (bank/Bank.sol#427) is not in mixedCase
  Parameter Bank.recollateralize(uint256,uint256)._collatIn (bank/Bank.sol#468) is not in mixedCase
  Parameter Bank.recollateralize(uint256,uint256)._cubeOutMin (bank/Bank.sol#468) is not in mixedCase
  Parameter Bank.mintTowerByProfit(uint256)._amount (bank/Bank.sol#615) is not in mixedCase
  Parameter BankRecollatStates.setPoolCeiling(uint256)._poolCeiling (bank/BankRecollatStates.sol#24) is not in mixedCase
  Parameter BankRecollatStates.setRctMaxPerHour(uint256)._rctMaxPerHour (bank/BankRecollatStates.sol#29) is not in mixedCase
  Parameter BankSafe.transferCollatTo(address,uint256)._to (bank/BankSafe.sol#75) is not in mixedCase
⚠ 0
```

```
Parameter BankRecollatStates.setRctMaxPerHour(uint256)._rctMaxPerHour (bank/BankRecollatStates.sol#29) is not in mixedCase
Parameter BankSafe.transferCollatTo(address,uint256)._to (bank/BankSafe.sol#75) is not in mixedCase
Parameter BankSafe.transferCollatTo(address,uint256)._amt (bank/BankSafe.sol#75) is not in mixedCase
Parameter BankSafe.transferCubeTo(address,uint256)._to (bank/BankSafe.sol#96) is not in mixedCase
Parameter BankSafe.transferCubeTo(address,uint256)._amt (bank/BankSafe.sol#96) is not in mixedCase
Parameter BankSafe.extractProfit(uint256)._amount (bank/BankSafe.sol#199) is not in mixedCase
Parameter BankSafe.setProfitController(address)._profitController (bank/BankSafe.sol#236) is not in mixedCase
Parameter BankSafe.transferTo(address,address,uint256)._receiver (bank/BankSafe.sol#287) is not in mixedCase
Parameter BankSafe.transferTo(address,address,uint256)._token (bank/BankSafe.sol#288) is not in mixedCase
Parameter BankSafe.transferTo(address,address,uint256)._amount (bank/BankSafe.sol#289) is not in mixedCase
Parameter BankSafe.setIdleCollateralUtilizationRatio(uint256)._idleCollateralUtilizationRatio (bank/BankSafe.sol#301) is not in mixedCase
Parameter BankSafe.setReservedCollateralThreshold(uint256)._reservedCollateralThreshold (bank/BankSafe.sol#315) is not in mixedCase
Parameter BankSafe.setExcessCollateralSafetyMargin(uint256)._excessCollateralSafetyMargin (bank/BankSafe.sol#326) is not in mixedCase
Parameter BankSafe.setAaveLendingPool(address)._aaveLendingPool (bank/BankSafe.sol#337) is not in mixedCase
Parameter BankSafe.setAaveIncentiveController(address)._aaveIncentivesController (bank/BankSafe.sol#343) is not in mixedCase
Parameter BankStates.setPriceBand(uint256)._priceBand (bank/BankStates.sol#67) is not in mixedCase
Parameter BankStates.setTcrMovement(uint256)._tcrMovement (bank/BankStates.sol#72) is not in mixedCase
Parameter BankStates.setUpdatePeriod(uint32)._updatePeriod (bank/BankStates.sol#77) is not in mixedCase
Parameter BankStates.setTcrMin(uint256)._tcrMin (bank/BankStates.sol#85) is not in mixedCase
Parameter BankStates.setEcrMin(uint256)._ecrMin (bank/BankStates.sol#90) is not in mixedCase
Parameter BankStates.setMintFee(uint256)._mintFee (bank/BankStates.sol#110) is not in mixedCase
Parameter BankStates.setRedeemFee(uint256)._redeemFee (bank/BankStates.sol#115) is not in mixedCase
Parameter BankStates.setZapSlippage(uint256)._zapSlippage (bank/BankStates.sol#120) is not in mixedCase
Parameter BankStates.setSwapFee(uint256)._swapFee (bank/BankStates.sol#125) is not in mixedCase
Parameter EnableSwap.setSwapController(address)._swapController (common/EnableSwap.sol#13) is not in mixedCase
Parameter OnlyArbitrager.setArbitrager(address)._arbitrager (common/OnlyArbitrager.sol#16) is not in mixedCase
Parameter OnlyBank.setBank(address)._bank (common/OnlyBank.sol#28) is not in mixedCase
Parameter TowerProtocol.setOperator(address)._operator (common/TowerProtocol.sol#43) is not in mixedCase
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (node_modules/@uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol#20) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (node_modules/@uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol#21) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (node_modules/@uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol#38) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Bank.slitherConstructorVariables() (bank/Bank.sol#22-643) uses literals with too many digits:
        - poolCeiling = 10000000e6 (bank/BankRecollatStates.sol#17)
BankSafe.slitherConstructorConstantVariables() (bank/BankSafe.sol#16-354) uses literals with too many digits:
        - EXCESS_COLLATERAL_SAFETY_MARGIN_MIN = 100000 (bank/BankSafe.sol#33)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
```

```
TowerProtocol.USDC_PRECISION (common/TowerProtocol.sol#11) is never used in Bank (bank/Bank.sol#22-643)
TowerProtocol.TOWER_PRECISION (common/TowerProtocol.sol#13) is never used in Bank (bank/Bank.sol#22-643)
TowerProtocol.ADDRESS_USDC (common/TowerProtocol.sol#17-18) is never used in Bank (bank/Bank.sol#22-643)
TowerProtocol.ADDRESS_WMATIC (common/TowerProtocol.sol#19-20) is never used in Bank (bank/Bank.sol#22-643)
BankStates.LIMIT_SWAP_TIME (bank/BankStates.sol#54) is never used in Bank (bank/Bank.sol#22-643)
TowerProtocol.PRICE_PRECISION (common/TowerProtocol.sol#10) is never used in BankSafe (bank/BankSafe.sol#16-354)
TowerProtocol.USDC_PRECISION (common/TowerProtocol.sol#11) is never used in BankSafe (bank/BankSafe.sol#16-354)
TowerProtocol.MISSING_PRECISION (common/TowerProtocol.sol#12) is never used in BankSafe (bank/BankSafe.sol#16-354)
TowerProtocol.TOWER_PRECISION (common/TowerProtocol.sol#13) is never used in BankSafe (bank/BankSafe.sol#16-354)
TowerProtocol.CUBE_PRECISION (common/TowerProtocol.sol#14) is never used in BankSafe (bank/BankSafe.sol#16-354)
TowerProtocol.SWAP_FEE_PRECISION (common/TowerProtocol.sol#15) is never used in BankSafe (bank/BankSafe.sol#16-354)
TowerProtocol.ADDRESS_USDC (common/TowerProtocol.sol#17-18) is never used in BankSafe (bank/BankSafe.sol#16-354)
TowerProtocol.ADDRESS_WMATIC (common/TowerProtocol.sol#19-20) is never used in BankSafe (bank/BankSafe.sol#16-354)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

BankRecollatStates.bonusRate (bank/BankRecollatStates.sol#11) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

init(address,address,address,address,address,address,address,address,address,address,uint256) should be declared external:
        - Bank.init(address,address,address,address,address,address,address,address,address,address,uint256) (bank/Bank.sol#68-105)
setContracts(address,address,address,address) should be declared external:
        - Bank.setContracts(address,address,address,address) (bank/Bank.sol#107-125)
update() should be declared external:
        - Bank.update() (bank/Bank.sol#151-168)
zapMint(uint256,uint256) should be declared external:
        - Bank.zapMint(uint256,uint256) (bank/Bank.sol#233-297)
claimIncentiveRewards() should be declared external:
        - BankSafe.claimIncentiveRewards() (bank/BankSafe.sol#250-263)
transferTo(address,address,uint256) should be declared external:
        - BankSafe.transferTo(address,address,uint256) (bank/BankSafe.sol#286-298)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#62-65)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
bank/Bank.sol analyzed (29 contracts with 77 detectors), 166 result(s) found
```

7.2 Mythril

TowerERC20

```
dev-varun@power-station:~/Documents/tower-v2-contracts-audit-main$ myth analyze ERC20/TowerERC20.sol -t 1
mythril.interfaces.cli [ERROR]: input files do not contain any valid contracts
dev-varun@power-station:~/Documents/tower-v2-contracts-audit-main$
```

Bank

```
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: setZapSlippage(uint256), txdata: 0xf6fc5516000000000000000000000000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: setZapSlippage(uint256)
PC address: 28618
Estimated Gas Usage: 1460 - 1885
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: common/TowerProtocol.sol:36

require(
            msg.sender == owner() || msg.sender == operator,
            "Tower: sender != operator"
        )

--------------------
Initial State:

Account: [CREATOR], balance: 0x2000000000, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: setZapSlippage(uint256), txdata: 0xf6fc5516000000000000000000000000000000000000000000000000000000000000000000, value: 0x0
```

```
==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: setEcrMin(uint256)
PC address: 25972
Estimated Gas Usage: 1417 - 1842
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: common/TowerProtocol.sol:36

require(
            msg.sender == owner() || msg.sender == operator,
            "Tower: sender != operator"
        )

--------------------
Initial State:

Account: [CREATOR], balance: 0x23, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: setEcrMin(uint256), txdata: 0xd07007ee0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: toggleRedeemPaused()
PC address: 26157
Estimated Gas Usage: 1165 - 1590
```

```
==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: toggleRecollatPaused()
PC address: 17985
Estimated Gas Usage: 1158 - 1583
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: common/TowerProtocol.sol:36

require(
            msg.sender == owner() || msg.sender == operator,
            "Tower: sender != operator"
        )

--------------------
Initial State:

Account: [CREATOR], balance: 0x2, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: toggleRecollatPaused(), txdata: 0xa432a22e, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: setTcrMin(uint256)
PC address: 19613
Estimated Gas Usage: 1381 - 1806
```

```
Account: [ATTACKER], balance: 0x2000000, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [ATTACKER], function: update(), txdata: 0xa2e62045, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: toggleRecollatPaused()
PC address: 17898
Estimated Gas Usage: 1144 - 1569
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: common/TowerProtocol.sol:37

msg.sender == owner() || msg.sender == operator

--------------------
Initial State:

Account: [CREATOR], balance: 0x2, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: toggleRecollatPaused(), txdata: 0xa432a22e, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
```

```
In file: common/OnlyArbitrager.sol:12

require(msg.sender == arbitrager, "OnlyArbitrager: sender != arb")

--------------------
Initial State:

Account: [CREATOR], balance: 0xa400000004000000, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: arbRedeem(uint256), txdata: 0x8d73e154000000000000000000000000000000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on predictable environment variable ====
SWC ID: 116
Severity: Low
Contract: Bank
Function name: update()
PC address: 17360
Estimated Gas Usage: 8940 - 30025
A control flow decision is made based on The block.timestamp environment variable.
The block.timestamp environment variable is used to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block num
ber and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use an
y of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.
--------------------
In file: bank/Bank.sol:155

require(_timeElapsed >= updatePeriod, "Bank: update too soon")

--------------------
Initial State:

Account: [CREATOR], balance: 0x20000a0100009102, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x2000000, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}
```

```
==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: mintTowerByProfit(uint256)
PC address: 13719
Estimated Gas Usage: 1461 - 1886
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: common/TowerProtocol.sol:36

require(
            msg.sender == owner() || msg.sender == operator,
            "Tower: sender != operator"
        )

--------------------
Initial State:

Account: [CREATOR], balance: 0x1, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: mintTowerByProfit(uint256), txdata: 0x86f936590000000000000000000000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: setSwapController(address)
PC address: 14577
Estimated Gas Usage: 1442 - 1867
--------------------
Initial State:

Account: [CREATOR], balance: 0x400000000002, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: update(), txdata: 0xa2e62045, value: 0x0

==== Dependence on predictable environment variable ====
SWC ID: 116
Severity: Low
Contract: Bank
Function name: update()
PC address: 36244
Estimated Gas Usage: 8028 - 28783
A control flow decision is made based on The block.timestamp environment variable.
The block.timestamp environment variable is used to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block num
ber and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use an
y of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.
--------------------
In file: #utility.yul:91
Initial State:

Account: [CREATOR], balance: 0x10000408100004420, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x100000000000000, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [ATTACKER], function: update(), txdata: 0xa2e62045, value: 0x0
Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: setZapSlippage(uint256), txdata: 0xf6fc5516000000000000000000000000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on predictable environment variable ====
SWC ID: 116
Severity: Low
Contract: Bank
Function name: update()
PC address: 31067
Estimated Gas Usage: 7822 - 28577
A control flow decision is made based on The block.timestamp environment variable.
The block.timestamp environment variable is used to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block num
ber and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use an
y of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.
--------------------
In file: node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol:78

require(value <= type(uint64).max, "SafeCast: value doesn't fit in 64 bits")

--------------------
Initial State:

Account: [CREATOR], balance: 0x400000000002, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: update(), txdata: 0xa2e62045, value: 0x0

==== Dependence on predictable environment variable ====
SWC ID: 116
Severity: Low
Contract: Bank
Function name: update()
```

```
------------------
In file: node_modules/@openzeppelin/contracts/access/Ownable.sol:43

require(owner() == _msgSender(), "Ownable: caller is not the owner")

------------------
Initial State:

Account: [CREATOR], balance: 0xc000000000000003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: transferOwnership(address), txdata: 0xf2fde38b0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: setZapSlippage(uint256)
PC address: 28531
Estimated Gas Usage: 1446 - 1871
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
------------------
In file: common/TowerProtocol.sol:37

msg.sender == owner() || msg.sender == operator

------------------
Initial State:

Account: [CREATOR], balance: 0x2104180000000001, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}
```

```
SWC ID: 115
Severity: Low
Contract: Bank
Function name: setMintFee(uint256)
PC address: 28098
Estimated Gas Usage: 1434 - 1859
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
------------------
In file: node_modules/@openzeppelin/contracts/access/Ownable.sol:43

require(owner() == _msgSender(), "Ownable: caller is not the owner")

------------------
Initial State:

Account: [CREATOR], balance: 0x446318c6000000003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: setMintFee(uint256), txdata: 0xeddd0d9c0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: transferOwnership(address)
PC address: 28289
Estimated Gas Usage: 1505 - 1930
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
------------------
In file: node_modules/@openzeppelin/contracts/access/Ownable.sol:43
```

```
Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: setPoolCeiling(uint256), txdata: 0xe86bc6190000000000000000000000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: setPoolCeiling(uint256)
PC address: 27895
Estimated Gas Usage: 1461 - 1886
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
------------------
In file: common/TowerProtocol.sol:36

require(
            msg.sender == owner() || msg.sender == operator,
            "Tower: sender != operator"
        )

------------------
Initial State:

Account: [CREATOR], balance: 0x2000000000, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: setPoolCeiling(uint256), txdata: 0xe86bc6190000000000000000000000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
```

```
==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: setArbitrager(address)
PC address: 27521
Estimated Gas Usage: 1506 - 1931
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: node_modules/@openzeppelin/contracts/access/Ownable.sol:43

require(owner() == _msgSender(), "Ownable: caller is not the owner")

--------------------
Initial State:

Account: [CREATOR], balance: 0x44000000002000003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: setArbitrager(address), txdata: 0xe771ee72000000000000000000000000000000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: setPoolCeiling(uint256)
PC address: 27808
Estimated Gas Usage: 1447 - 1872
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
```

```
Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: collect(), txdata: 0xe5225381, value: 0x0

==== Dependence on predictable environment variable ====
SWC ID: 120
Severity: Low
Contract: Bank
Function name: collect()
PC address: 26756
Estimated Gas Usage: 7216 - 27501
A control flow decision is made based on The block.number environment variable.
The block.number environment variable is used to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block number
 and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any o
f those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.
--------------------
In file: bank/Bank.sol:352

require(
            lastRedeemed[msg.sender] + 1 <= block.number,
            "Bank: collect too soon"
        )

--------------------
Initial State:

Account: [CREATOR], balance: 0x200, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x3, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [ATTACKER], function: collect(), txdata: 0xe5225381, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
```

```
Initial State:

Account: [CREATOR], balance: 0x2, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: toggleRedeemPaused(), txdata: 0xd2a99c84, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: collect()
PC address: 26536
Estimated Gas Usage: 268 - 363
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: common/TowerProtocol.sol:31

require(msg.sender == tx.origin, "Tower: sender != origin")

--------------------
Initial State:

Account: [CREATOR], balance: 0x900000000002800, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:
```

```
Estimated Gas Usage: 1165 - 1590
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: common/TowerProtocol.sol:37

msg.sender == owner() || msg.sender == operator

--------------------
Initial State:

Account: [CREATOR], balance: 0x2, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: toggleRedeemPaused(), txdata: 0xd2a99c84, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: toggleRedeemPaused()
PC address: 26244
Estimated Gas Usage: 1179 - 1604
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: common/TowerProtocol.sol:36

require(
            msg.sender == owner() || msg.sender == operator,
            "Tower: sender != operator"
        )
```

```
Initial State:

Account: [CREATOR], balance: 0x40210000000120281, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: setRctMaxPerHour(uint256), txdata: 0xc40aed170000000000000000000000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: setEcrMin(uint256)
PC address: 25885
Estimated Gas Usage: 1403 - 1828
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: common/TowerProtocol.sol:37

msg.sender == owner() || msg.sender == operator

--------------------
Initial State:

Account: [CREATOR], balance: 0x41000000000043, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: setEcrMin(uint256), txdata: 0xd07007ee0000000000000000000000000000000000000000000000000000000000000000, value: 0x0
```

```
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: common/TowerProtocol.sol:37

msg.sender == owner() || msg.sender == operator

--------------------
Initial State:

Account: [CREATOR], balance: 0x42080000000000003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: setRctMaxPerHour(uint256), txdata: 0xc40aed170000000000000000000000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: setRctMaxPerHour(uint256)
PC address: 25630
Estimated Gas Usage: 1418 - 1843
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: common/TowerProtocol.sol:36

require(
            msg.sender == owner() || msg.sender == operator,
            "Tower: sender != operator"
        )
```

```
SWC ID: 115
Severity: Low
Contract: Bank
Function name: redeem(uint256,uint256,uint256)
PC address: 22970
Estimated Gas Usage: 824 - 919
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: common/TowerProtocol.sol:31

require(msg.sender == tx.origin, "Tower: sender != origin")

--------------------
Initial State:

Account: [CREATOR], balance: 0x3, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: redeem(uint256,uint256,uint256), txdata: 0xb81922050000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: setContracts(address,address,address,address)
PC address: 24876
Estimated Gas Usage: 2123 - 2548
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
```

```
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: zapMint(uint256,uint256), txdata: 0xb15582fc0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: setOperator(address)
PC address: 22691
Estimated Gas Usage: 1528 - 1953
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: node_modules/@openzeppelin/contracts/access/Ownable.sol:43

require(owner() == _msgSender(), "Ownable: caller is not the owner")

--------------------
Initial State:

Account: [CREATOR], balance: 0x4000000000000001, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: setOperator(address), txdata: 0xb3ab15fb0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
```

```
require(
            msg.sender == owner() || msg.sender == operator,
            "Tower: sender != operator"
        )

--------------------
Initial State:

Account: [CREATOR], balance: 0x3, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: setTcrMin(uint256), txdata: 0xafae97a30000000000000000000000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: zapMint(uint256,uint256)
PC address: 19875
Estimated Gas Usage: 699 - 794
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: common/TowerProtocol.sol:31

require(msg.sender == tx.origin, "Tower: sender != origin")

--------------------
Initial State:

Account: [CREATOR], balance: 0x2000000000000003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
```

```
PC address: 19613
Estimated Gas Usage: 1381 - 1806
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: common/TowerProtocol.sol:37

msg.sender == owner() || msg.sender == operator

--------------------
Initial State:

Account: [CREATOR], balance: 0x3f710c0500000003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x1, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [ATTACKER], function: setTcrMin(uint256), txdata: 0xafae97a3000000000000000000000000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: setTcrMin(uint256)
PC address: 19700
Estimated Gas Usage: 1395 - 1820
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: common/TowerProtocol.sol:36

require(
          msg.sender == owner() || msg.sender == operator,
```

```
Contract: Bank
Function name: setSwapController(address)
PC address: 14577
Estimated Gas Usage: 1442 - 1867
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: node_modules/@openzeppelin/contracts/access/Ownable.sol:43

require(owner() == _msgSender(), "Ownable: caller is not the owner")

--------------------
Initial State:

Account: [CREATOR], balance: 0xc6318c7000000001, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: setSwapController(address), txdata: 0x8913c3df0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: arbRedeem(uint256)
PC address: 15046
Estimated Gas Usage: 7185 - 27610
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: common/OnlyArbitrager.sol:12

require(msg.sender == arbitrager, "OnlyArbitrager: sender != arb")
```

```
Account: [CREATOR], balance: 0x2, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: renounceOwnership(), txdata: 0x715018a6, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: mintTowerByProfit(uint256)
PC address: 13632
Estimated Gas Usage: 1447 - 1872
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: common/TowerProtocol.sol:37

msg.sender == owner() || msg.sender == operator

--------------------
Initial State:

Account: [CREATOR], balance: 0x2000000003, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: mintTowerByProfit(uint256), txdata: 0x86f936590000000000000000000000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
```

```
require(
            msg.sender == owner() || msg.sender == operator,
            "Tower: sender != operator"
        )

--------------------
Initial State:

Account: [CREATOR], balance: 0x2a0001, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: setPriceBand(uint256), txdata: 0x6140133b0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: renounceOwnership()
PC address: 12784
Estimated Gas Usage: 1198 - 1623
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: node_modules/@openzeppelin/contracts/access/Ownable.sol:43

require(owner() == _msgSender(), "Ownable: caller is not the owner")

--------------------
Initial State:

Account: [CREATOR], balance: 0x2, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
```

```
Function name: setPriceBand(uint256)
PC address: 12449
Estimated Gas Usage: 1360 - 1785
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: common/TowerProtocol.sol:37

msg.sender == owner() || msg.sender == operator

--------------------
Initial State:

Account: [CREATOR], balance: 0x7fc2, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x1, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [ATTACKER], function: setPriceBand(uint256), txdata: 0x6140133b0000000000000000000000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: setPriceBand(uint256)
PC address: 12536
Estimated Gas Usage: 1374 - 1799
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: common/TowerProtocol.sol:36

require(
            msg.sender == owner() || msg.sender == operator,
```

```
Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: toggleEnableEcr(), txdata: 0x4eb2d562, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: setRedeemFee(uint256)
PC address: 12264
Estimated Gas Usage: 1480 - 1905
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: node_modules/@openzeppelin/contracts/access/Ownable.sol:43

require(owner() == _msgSender(), "Ownable: caller is not the owner")

--------------------
Initial State:

Account: [CREATOR], balance: 0x4000000000000002, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: setRedeemFee(uint256), txdata: 0x5d841af5000000000000000000000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: setPriceBand(uint256)
PC address: 12449
```

```
Contract: Bank
Function name: setUpdatePeriod(uint32)
PC address: 11618
Estimated Gas Usage: 1403 - 1828
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: common/TowerProtocol.sol:36

require(
            msg.sender == owner() || msg.sender == operator,
            "Tower: sender != operator"
        )

--------------------
Initial State:

Account: [CREATOR], balance: 0x50100600000040001, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [CREATOR], function: setUpdatePeriod(uint32), txdata: 0x4ad30b500000000000000000000000000000000000000000000000000000000000000000, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: toggleEnableEcr()
PC address: 11845
Estimated Gas Usage: 1145 - 1570
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
```

```
Initial State:

Account: [CREATOR], balance: 0x100000000000002, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: toggleEnableEcr(), txdata: 0x4eb2d562, value: 0x0

==== Dependence on tx.origin ====
SWC ID: 115
Severity: Low
Contract: Bank
Function name: toggleEnableEcr()
PC address: 11932
Estimated Gas Usage: 1159 - 1584
Use of tx.origin as a part of authorization control.
The tx.origin environment variable has been found to influence a control flow decision. Note that using tx.origin as a security control might cause a situa
tion where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use msg.sender instead.
--------------------
In file: common/TowerProtocol.sol:36

require(
            msg.sender == owner() || msg.sender == operator,
            "Tower: sender != operator"
        )

--------------------
Initial State:

Account: [CREATOR], balance: 0x1, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:
```

Farm

```
dev-varun@power-station:~/Documents/tower-v2-contracts-audit-main$ myth analyze farm/Farm.sol -t 3
Matplotlib is building the font cache; this may take a moment.
The analysis was completed successfully. No issues were detected.

dev-varun@power-station:~/Documents/tower-v2-contracts-audit-main$
```

Stake

```
dev-varun@power-station:~/Documents/tower-v2-contracts-audit-main$ myth analyze stake/CubeStake.sol -t 1
The analysis was completed successfully. No issues were detected.

dev-varun@power-station:~/Documents/tower-v2-contracts-audit-main$
```

7.0 Auditing Approach and Methodologies applied

Throughout the audit of the smart contract; care was taken to ensure:

- Overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per intended behaviour mentioned in the whitepaper.
- Implementation of token standards.
- Efficient use of gas.
- Code is safe from Re-entrancy and other vulnerabilities.

A combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy regarding the scope of the smart contract audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of smart contracts and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

7.1 Structural Analysis

In this step we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure Smart contract is structured in a way that will not result in future problems.

7.2 Static Analysis

Static Analysis of smart contracts was undertaken to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

7.3 Code Review / Manual Analysis

Manual Analysis or review of done to identify new vulnerabilities or to verify the vulnerabilities found during the Static Analysis. The contracts were completely manually analysed, and their logic was checked and compared with the one described in the whitepaper. It should also be noted that the results of the automated analysis were verified manually.

7.4 Gas Consumption

In this step, we checked the behaviour of all smart contracts in production. Checks were completed to understand how much gas gets consumed, along with the possibilities of optimisation of code to reduce gas consumption.

7.5 Tools and Platforms used for Audit

VSCode, Remix IDE, Truffle, Truffle Team, Ganache, Solhint, Mythril, Manticore, Slither.

7.6 Checked Vulnerabilities

We have scanned Tower Finance smart contracts for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC-20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

8.0 Limitations on Disclosure and Use of this Report

This report contains information concerning potential details of Tower Finance and methods for exploiting them. Entersoft recommends that special precautions be taken to protect the confidentiality of both this document and the information contained herein. Security Assessment is an uncertain process, based on past experiences, currently available information, and known threats. All information security systems, which by their nature are dependent on human beings, are vulnerable to some degree. Therefore, while Entersoft considers the major security vulnerabilities of the analyzed systems to have been identified, there can be no assurance that any exercise of this nature will identify all possible vulnerabilities or propose exhaustive and operationally viable recommendations to mitigate those exposures. In addition, the analysis set forth herein is based on the technologies and known threats as of the date of this report. As technologies and risks change over time, the vulnerabilities associated with the operation of the Smart Contract described in this report, as well as the actions necessary to reduce the exposure to such vulnerabilities will also change. Entersoft makes no undertaking to supplement or update this report based on changed circumstances or facts of which Entersoft becomes aware after the date hereof, absent a specific written agreement to perform the supplemental or updated analysis. This report may recommend that Entersoft use certain software or hardware products manufactured or maintained by other vendors. Entersoft bases these recommendations upon its prior experience with the capabilities of those products. Nonetheless, Entersoft does not and cannot warrant that a particular product will work as advertised by the vendor, nor that it will operate in the manner intended. This report was prepared by Entersoft for the exclusive benefit of Tower Finance and is proprietary information. The Non-Disclosure Agreement (NDA) in effect between Entersoft and Tower Finance govern the disclosure of this report to all other parties including product vendors and suppliers.