Coti

cotitech-io

Security Assessment

March 29th, 2021

Audited By:
Adrian Hetman @ CertiK
adrian.hetman@certik.org
Reviewed By:
Alex Papageorgiou @ CertiK
alex.papageorgiou@certik.org

Disclaimer

CertiK reports are not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security review.

CertiK Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

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

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has completed a round of auditing with the intention to increase the quality of the company/product's IT infrastructure and or source code.

Overview

Project Summary

Project Name	cotitech-io - Coti
Description	The CVI (Cryptocurrency Volatility Index) platform is a decentralized solution to analyzing the market's expectation of future volatility and enabling users to open positions based on these predictions that can be subsequently liquidated or redeemed depending on whether the value remains above the liquidation threshold.
Platform	Ethereum; Solidity, Yul
Codebase	GitHub Repository
Commits	1. 7b76596b11a851e13df1277eb85a62b1306750b2 2. 0576c37797266aaaa98fc1111ac5461a3b75e0b4

Audit Summary

Delivery Date	March 29thnd, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	1
Timeline	March 16th, 2021 - March 22nd, 2021

Vulnerability Summary

Total Issues	14
Total Critical	0
Total Major	0
Total Medium	1
Total Minor	10
Total Informational	3

Executive Summary

We were tasked with auditing the changes of CVI implementation of Coti in Solidity. Our audit consisted of checking the differences between the previous audit we did and the version we got. We compared the version of contracts in V1 directory to contracts located at V2 and V3 directory.

We have found few minor issues that need to be addressed to raise the project's security. Still, we're glad to report we haven't found any critical vulnerabilities introduced with the changes. The findings below are mostly around lack of checks-effect pattern and lack of usage of nonReantrant modifier from OpenZeppelin to guard most crucial, publicly exposed functionality. The client was quick to fix these issues.
Most of the changes we've seen were refactoring the original codebase to have more concise and cleaner code. Some of the original audit issues that weren't resolved are now resolved in version 3, such as CVI-03, FCR-05, FCR-07, FCR-08 and PLA-01. Previously reported issues that were remediated remained fixed.

One thing we have found that we're recommending of changing is usage of sub-256 bit data types utilized across the project. We find these types redundant in usage, they introduces more gas costs to the operations of the protocol and is prone to more mathematical errors. We would advise to change all sub-256 bit data types to standard uint256. This still remains unaddressed.

Team added new version of Liquidation contract called LiquidationV2.sol which PlatformV2 now uses. As client states:"It accepts the CVI value at open position and the leverage, to calculate the liquidation without the impact of the CVI and with different percentages values for each leverage.". We haven't found this change to affect negatively the PlatformV2.sol.

System Analysis

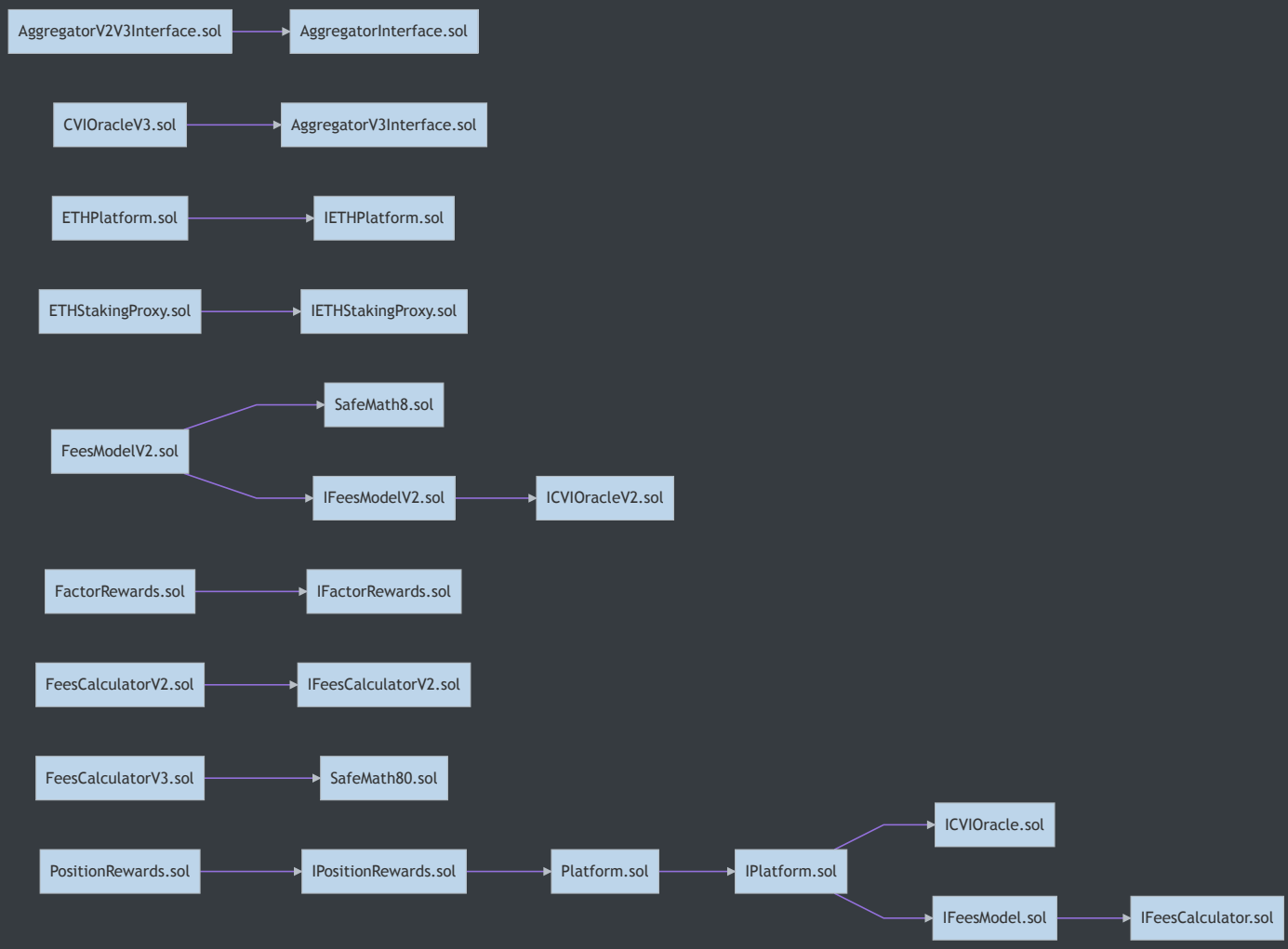
The owner of PlatformV2, FeesCalculatorV2/V3, FactorRewards, PositionRewards V1 and V2, is in full control of contract's normal functions and many behavioral contracts parameters can be changed by the owner of the contract at will.
In case of lost access to the private key of an account or mishandling security of private keys, an attacker could benefit from that and replace rewarder account or implementations of feesCalculator with their own implementation fo the contract would lead to stealing of funds.
Given the fact that the owner of the contract’s deployed can severely affect the system’s normal operation, we advise that a governance system or multi-signature wallet is utilized instead of a single account.

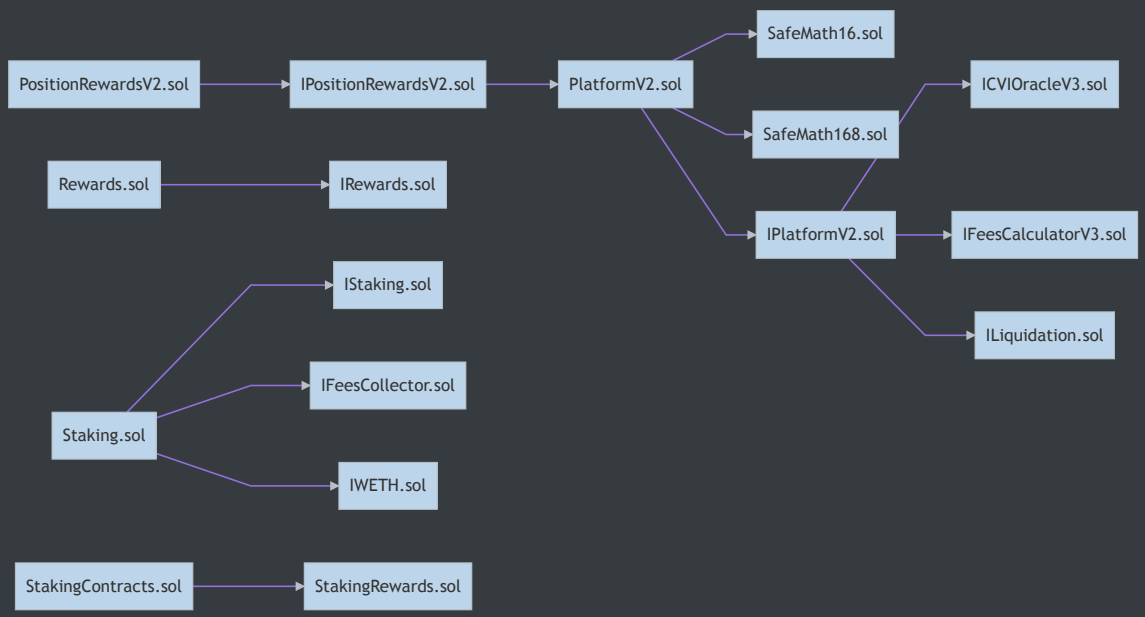
Files In Scope

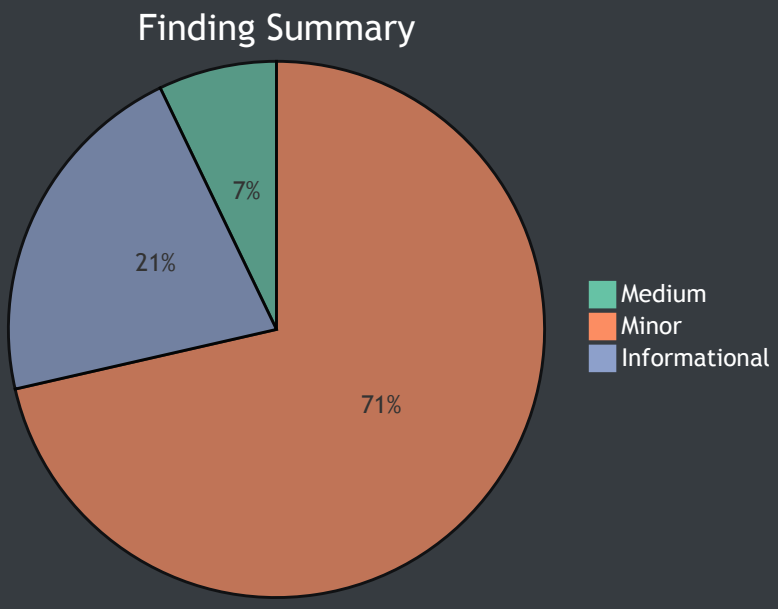
ID	Contract	Location
WET	WETH9.sol	contracts/external/WETH9.sol
CVI	CVIOracle.sol	contracts/v1/CVIOracle.sol
FCR	FeesCalculator.sol	contracts/v1/FeesCalculator.sol
FML	FeesModel.sol	contracts/v1/FeesModel.sol
GOV	GOVI.sol	contracts/v1/GOVI.sol
LIQ	Liquidation.sol	contracts/v1/Liquidation.sol
PLA	Platform.sol	contracts/v1/Platform.sol
REW	Rewards.sol	contracts/v1/Rewards.sol
STA	Staking.sol	contracts/v1/Staking.sol
CVO	CVIOracleV2.sol	contracts/v2/CVIOracleV2.sol
FCV	FeesCalculatorV2.sol	contracts/v2/FeesCalculatorV2.sol
FMV	FeesModelV2.sol	contracts/v2/FeesModelV2.sol
PRS	PositionRewards.sol	contracts/v2/PositionRewards.sol
CVV	CVIOracleV3.sol	contracts/v3/CVIOracleV3.sol
ETH	ETHPlatform.sol	contracts/v3/ETHPlatform.sol
ETS	ETHStakingProxy.sol	contracts/v3/ETHStakingProxy.sol
EET	ExtractETH.sol	contracts/v3/ExtractETH.sol
FRS	FactorRewards.sol	contracts/v3/FactorRewards.sol
CON	FeesCalculatorV3.sol	contracts/v3/FeesCalculatorV3.sol
PV2	PlatformV2.sol	contracts/v3/PlatformV2.sol
PRV	PositionRewardsV2.sol	contracts/v3/PositionRewardsV2.sol
GOI	GOVIAirdrop.sol	contracts/v1/distribution/GOVIAirdrop.sol
AIE	AggregatorInterface.sol	contracts/v1/interfaces/AggregatorInterface.sol
AVV	AggregatorV2V3Interface.sol	contracts/v1/interfaces/AggregatorV2V3Interface.sol

AVI	AggregatorV3Interface.sol	contracts/v1/interfaces/AggregatorV3Interface.sol
ICV	ICVIOracle.sol	contracts/v1/interfaces/ICVIOracle.sol
CON	IFeesCalculator.sol	contracts/v1/interfaces/IFeesCalculator.sol
IFC	IFeesCollector.sol	contracts/v1/interfaces/IFeesCollector.sol
IFM	IFeesModel.sol	contracts/v1/interfaces/IFeesModel.sol
ILN	ILiquidation.sol	contracts/v1/interfaces/ILiquidation.sol
IPM	IPlatform.sol	contracts/v1/interfaces/IPlatform.sol
IRS	IRewards.sol	contracts/v1/interfaces/IRewards.sol
ISG	IStaking.sol	contracts/v1/interfaces/IStaking.sol
IWE	IWETH.sol	contracts/v1/interfaces/IWETH.sol
SCS	StakingContracts.sol	contracts/v1/staking/StakingContracts.sol
SRS	StakingRewards.sol	contracts/v1/staking/StakingRewards.sol
SM6	SafeMath16.sol	contracts/v1/utils/SafeMath16.sol
SM8	SafeMath8.sol	contracts/v1/utils/SafeMath8.sol
SM0	SafeMath80.sol	contracts/v1/utils/SafeMath80.sol
ICI	ICVIOracleV2.sol	contracts/v2/interfaces/ICVIOracleV2.sol
CON	IFeesCalculatorV2.sol	contracts/v2/interfaces/IFeesCalculatorV2.sol
IFV	IFeesModelV2.sol	contracts/v2/interfaces/IFeesModelV2.sol
IPR	IPositionRewards.sol	contracts/v2/interfaces/IPositionRewards.sol
ICO	ICVIOracleV3.sol	contracts/v3/interfaces/ICVIOracleV3.sol
IET	IETHPlatform.sol	contracts/v3/interfaces/IETHPlatform.sol
IEH	IETHStakingProxy.sol	contracts/v3/interfaces/IETHStakingProxy.sol
IFR	IFactorRewards.sol	contracts/v3/interfaces/IFactorRewards.sol
CON	IFeesCalculatorV3.sol	contracts/v3/interfaces/IFeesCalculatorV3.sol
IPV	IPlatformV2.sol	contracts/v3/interfaces/IPlatformV2.sol
IRV	IPositionRewardsV2.sol	contracts/v3/interfaces/IPositionRewardsV2.sol
SM1	SafeMath168.sol	contracts/v3/utils/SafeMath168.sol

File Dependency Graph




PositionRewardsV2.sol
IPositionRewardsV2.sol
PlatformV2.sol
SafeMath16.sol
SafeMath168.sol
ICVIOracleV3.sol
Rewards.sol
IRewards.sol
IPlatformV2.sol
IFeesCalculatorV3.sol
ILiquidation.sol
IStaking.sol
IFeesCollector.sol
Staking.sol
IWETH.sol
StakingContracts.sol
StakingRewards.sol


Finding Summary
7%
21%
71%
Medium
Minor
Informational

Manual Review Findings

ID	Title	Type	Severity	Resolved
PLA-01	safeApprove will prevent setting the same fee collector	Volatile Code	Minor	✓
FCV-01	Minimum Oracle Heartbeat Period	Volatile Code	Minor	⟳
FCV-02	Variable Tight-Packing	Gas Optimization	Informational	✓
FMV-01	Redundant Statements	Dead Code	Informational	✓
ETH-01	Possibility of re-entrancy attack	Volatile Code	Minor	✓
ETH-02	Function name doesn't match the logic	Logical Issue	Informational	✓
FRS-01	Claimed Position Units can be counted twice.	Mathematical Operations	Medium	✓
CON-01	Wrong value assignment to `updatedTurbulenceIndicatorPercent`	Volatile Code	Minor	✓
CON-02	Lack of safemath usage	Mathematical Operations	Minor	✓
CON-03	Potential for overflow	Mathematical Operations	Minor	✓
PV2-01	Possibility of re-entrancy attack	Volatile Code	Minor	✓
PV2-02	Checks-effects-pattern not used	Volatile Code	Minor	✓
PV2-03	Fees collector doesn't decrease allowance when new fees collector is	Volatile Code	Minor	✓

	introduced			
PRV-01	clainReward() can be called twice during 24h period.	Volatile Code	Minor	

PLA-01: safeApprove will prevent setting the same fee collector

Type	Severity	Location
Volatile Code	Minor	Platform.sol L112

Description:

safeApprove function is non standard and internally evaluates that the approval is zero when set to a non-zero value. As it is never set to zero, setting the same fee collector will fail.

Recommendation:

We suggest using standard approve method in this case and firstly set the feeCollector to 0, and then _newCollector to `uint256(-1)`. This way same fee collector can be set twice.

Alleviation:

Issue has been resolved in PlatformV2.sol file. Platform.sol is already deployed and new version of Platform fixes this issue.

FCV-01: Minimum Oracle Heartbeat Period

Type	Severity	Location
Volatile Code	Minor	FeesCalculatorV2.sol L104-L106

Description:

A minimum oracle heartbeat period is not guaranteed by the contract's code.

Recommendation:

We advise that a require check is imposed to ensure that the heartbeat period is within certain sensible bounds and cannot be maliciously altered so.

Alleviation:

The cotitech-io - Coti development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase due to time constraints.

FCV-02: Variable Tight-Packing

Type	Severity	Location
Gas Optimization	Informational	FeesCalculatorV2.sol L26-L30, L33-L38, L41-L45

Description:

The linked variable declaration is a uint16 declaration that can be tightly packed with other variables it is simultaneously read with.

Recommendation:

We advise to relocated linked variables into single localization so they can be tight-packed.

Alleviation:

Issue has been resolved in FeesCalculatorV3.sol. FeesCalculatorV2 is already deployed and no changes will be made to this contract.

FMV-01: Redundant Statements

Type	Severity	Location
Dead Code	Informational	FeesModelV2.sol L73-L74, L80-L81, L91-L92

Description:

The linked statements do not affect the functionality of the codebase and appear to be either leftovers from test code or older functionality.

Recommendation:

We advise that they are removed to better prepare the code for production environments.

Alleviation:

Issue not resolved.
Client's comment:
"Won't be fixed as code already deployed and newer version is clean"

ETH-01: Possibility of re-entrancy attack

Type	Severity	Location
Volatile Code	Minor	ETHPlatform.sol L20, L24

Description:

Linked functions have calls to externals contracts/addresses that could trigger an re-entrancy attack by malicious ERC20 transfer function or plain ETH transfer.

Recommendation:

It is recommended to follow checks-effects-interactions pattern for cases like this.
Another protection which we would also recommend is usage of `nonReantrant` modifier from `ReentrancyGuard.sol` OpenZeppelin framework.

Alleviation:

Issue resolved. The team opted for `nonReantrant` modifier.

ETH-02: Function name doesn't match the logic

Type	Severity	Location
Logical Issue	Informational	ETHPlatform.sol L29-L31

Description:

`transferTokens()` function do not conform to the actual logic of what function name suggest. Instead of sending tokens it does send ether.

Recommendation:

We would suggest renaming the function to something more appropriate to the actual logic.

Alleviation:

Issue resolved. Function name was changed to `transferFunds`.

FRS-01: Claimed Position Units can be counted twice.

Type	Severity	Location
Mathematical Operations	Medium	FactorRewards.sol L61

Description:

With the current logic of calculating the `claimedPositionUnits`, difference between new positionUnitAmount and currClaimedPositionsUnits is not added, only the full amount of new positionUnitAmount. This can lead to misscalculations of claimed positions.

Recommendation:

We would recommend in this case calculations done in this way to avoid any issues mentioned above:
`claimedPositionUnits[msg.sender][originalCreationTimestamp] = currClaimedPositionUnits.add(positionUnitsAmount - currClaimedPositionUnits)`

Alleviation:

Issue resolved. The team used solution recommended in our exhibit.

CON-01: Wrong value assignment to `updatedTurbulenceIndicatorPercent`

Type	Severity	Location
Volatile Code	Minor	FeesCalculatorV3.sol L147

Description:

In the linked if statement, `updatedTurbulenceIndicatorPercent` should be assigned `turbulenceFeeMinPercentThreshold` if it's current value is lower than it, not 0.

Recommendation:

We would advise to assign `turbulenceFeeMinPercentThreshold` to `updatedTurbulenceIndicatorPercent` when linked if is resolved positively.

Alleviation:

Issue resolved. The team changed the name of the parameter to better reflect intended behaviour.
Client's comment:
"This is by design, we changed the name of the parameter to better reflect it"

CON-02: Lack of safemath usage

Type	Severity	Location
Mathematical Operations	Minor	FeesCalculatorV3.sol L213

Description:

Linked lines should have been using SafeMath library as current implementation could lead to underflow in edge scenario.

Recommendation:

We recommend using `SafeMath` for these mathematical operations to ensure edge cases are accounted for.

Alleviation:

Issue resolved. The team used SafeMath library.

CON-03: Potential for overflow

Type	Severity	Location
Mathematical Operations	Minor	FeesCalculatorV3.sol L193

Description:

When `exponent` value is higher or equal than 256 it will cause overflow and `fundingFeeRatePercents` would produce a wrong calculation.

Recommendation:

We recommend using `SafeMath` for these mathematical operations to ensure edge cases are accounted for. Also we would recommend to create a require or if/else statement when exponent value is higher or equal 256 to cover the case of overflow.

Alleviation:

Issue isn't resolved. The team added require check for `exponent` variable but it is still checking if it's lower-or-equal to 256 where as it should check if it's only lower than 256.
When `(2**exponent)` is higher than `PRECISION_DECIMALS`, current calculation will evaluate to 0 + FUNDING_FEE_MIN_RATE.

PV2-01: Possibility of re-entrancy attack

Type	Severity	Location
Volatile Code	Minor	PlatformV2.sol L71, L75, L79, L99, L103

Description:

Linked functions have calls to externals contracts/addresses that could trigger an re-entrancy attack by malicious ERC20 transfer function or plain ETH transfer.

Recommendation:

It is recommended to follow checks-effects-interactions pattern for cases like this.
Another protection which we would also recommend is usage of `nonReantrant` modifier from `ReentrancyGuard.sol` OpenZeppelin framework.

Alleviation:

Issue has been resolved. The team opted for `nonReantrant` modifier.

PV2-02: Checks-effects-pattern not used

Type	Severity	Location
Volatile Code	Minor	PlatformV2.sol L360-L364, L178-179, L329-L333

Description:

State variables are changed after `_burn`, `collectProfit`, `_mint`, `collectTokens` and `transferTokens` functions calls.

Recommendation:

It is recommended to follow checks-effects-interactions pattern for cases like this.
It shields public functions from re-entrancy attacks. It's always a good practie to follow this pattern. `checks-effects-interaction` pattern also applies to ERC20 tokens as they can inform the recipient of a transfer in certain implementations.

Alleviation:

Issue has been resolved. The team used checks-effects-pattern a top of `nonReantrant` modifier.

PV2-03: Fees collector doesn't decrease allowance when new fees collector is introduced

Type	Severity	Location
Volatile Code	Minor	PlatformV2.sol L182-L192

Description:

When new Fees collector contract is updated and the new instance is used in Deposit Manager, the old Fees collector contract is still approved for max uint256 value to spend ERC20 token

Recommendation:

Upon changing the deployed version and updating fees collector, ERC20 token should have decreased the allowance of old fees collector instances to avoid any unexpected issues.

Alleviation:

Issue has been resolved. Please check the PLA-01 findings for more explanation.

PRV-01: clainReward() can be called twice during 24h period.

Type	Severity	Location
Volatile Code	Minor	PositionRewardsV2.sol L104-L106

Description:

The current implementation of claim period allows one to submit a transaction before the day ends and right after to perform a valid claim reward call which shouldn't be expected.

Recommendation:

We wold advise to compare block timestamp with creationTimestamp and see if it's bigger than 1 day.
`block.timestamp - creationTimestamp > 1 days`.

Alleviation:

The cotitech-io - Coti development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase due to time constraints.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Dead Code

Code that otherwise does not affect the functionality of the codebase and can be safely omitted.