Security Assessment

Goldfinch

Aug 16th, 2021

Table of Contents

Summary

This report has been prepared for Goldfinch smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Majority of the findings are of informational nature with nine minor findings and two medium findings. The minor findings comprise lack of validation for function parameters. The medium findings comprise the missing of success status check for low level `call` and the possibility of a token owner setting more redeemed principal than the deposited principal in PoolTokens. All of the findings are remediated except a few informational findings as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

Overview

Project Summary

Project Name	Goldfinch
Description	A lending protocol focused on enabling crypto holders to earn yield from real-world, off-chain borrowers. This is achieved by allowing anyone to contribute USDC to the pool. Governance-approved underwriters can then extend credit lines to individual borrowers, who can draw down capital from the pool for use off-chain. The audit comprise the delta between commits `9c574d9715e924854d1c447b857c4afd53b88a78` and `223e7d96ceba84dacbca92daf469523a6142be1e`.
Platform	Ethereum
Language	Solidity
Codebase	• https://github.com/goldfinch-eng/goldfinch-protocol/tree/223e7d96ceba84dacbca92daf469523a6142be1e/contracts • https://github.com/goldfinch-eng/goldfinch-protocol/tree/1b64fcd2ff8c435d698e433482f635023cadbe19/contracts
Commit	223e7d96ceba84dacbca92daf469523a6142be1e 1b64fcd2ff8c435d698e433482f635023cadbe19

Audit Summary

Delivery Date	Aug 16, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	0	0	0	0	0	0
Medium	2	0	0	0	0	2
Minor	10	0	0	0	0	10
Informational	31	0	7	0	0	24
Discussion	0	0	0	0	0	0

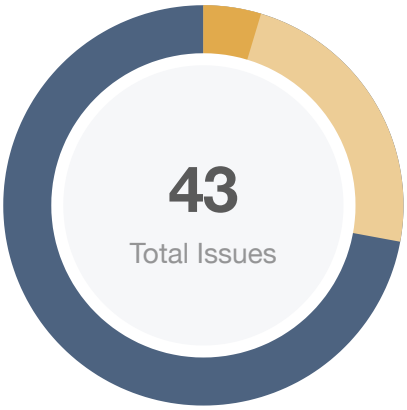
Audit Scope

ID	File	SHA256 Checksum
ERC	external/ERC721PresetMinterPauserAutoId.sol	4b53db152d6377601d3959ae8b72d94055753d1cb48aef25897fed3fae4693a0
FPG	external/FixedPoint.sol	44e32c06ba12cdd43c8dcafdada6224dfe15100c50533ef631e4312553726fea
IBG	interfaces/IBase.sol	3911dbc86751b269003137597aa652d288f4b3021b79a319778c9058b61ee3f1
IBN	interfaces/IBorrower.sol	ab0bd396cc39e354f34a1e65dd8210ad9d7340c064ebef5c79ee249011add8e7
ICU	interfaces/ICUSDCContract.sol	a10b73fd9235743baf21740015b7a32ef65df2b8274df38d82876802f8525b3f
ICD	interfaces/ICreditDesk.sol	1e0b4ef221dcb768ab45ea65dd9c13c2f0cf06ad615ba537c6c876c9314cb062
ICL	interfaces/ICreditLine.sol	a5776f2ded107fd76b4a6a822f787d3005febcb50c94519c408772a53963b5f8
IER	interfaces/IERC20withDec.sol	54ad14d05fbd2a0bf0c9f4cbd36a738f95b682d07bc9a3128d7bd38108d31871
IFG	interfaces/IFidu.sol	5d8f85d7eb58ea13f56731a799e3278520d50cb6722471fcedaccba62255fa6c
IFN	interfaces/IFund.sol	6715d4c0ad572cfbc3b1434973184aa91c21291543b7b570a3310e4259973f63
IFS	interfaces/IFundStrategy.sol	257a44454a508186de6e97f7f7548a246cf22ecec2b674a4ad215218e83c4386
IGC	interfaces/IGoldfinchConfig.sol	f70e25836cd186a35811567080813c72b3f21705350797e49e590810b69ff71d
IGF	interfaces/IGoldfinchFactory.sol	84af9b88a2d22f83d5f33f727e44afc9d8678490ac2fc9ec4ba3962655148f91
IMT	interfaces/IMigratedTranchedPool.sol	f8aa668317fb9e7c359f7080689c8db031a4eb256574e434f0696e970aec3a42
IPG	interfaces/IPool.sol	40b6797282ca06abfd5c8e0c370a428c257e77f6dab4e307e55650d8b4cf105b
IPT	interfaces/IPoolTokens.sol	6b0f05c9b6d9abb315f412ae43301574ed39613a35021eaa60f104d407bd8ed6

ID	File	SHA256 Checksum
ITP	interfaces/ITranchedPool.sol	02bdbb5ec489e1bc257e04d90301e95f18fff785597f05d4796aa78d81d34053
IVC	interfaces/IV1CreditLine.sol	39acc8a4caa3928aef3f60085d7a7c09d6f594cb2a353a4ba56f3fb4e76af5af
IVL	interfaces/IV2CreditLine.sol	b750eff87453fd0f4161e03df7ca777a08390a3a038a243d0102e4bfab7c36b1
SER	library/SafeERC20Transfer.sol	8e7893a11d6da3cb10f50332e510fa046514c59e4d2c08b0bf7338c65ba0479d
ACC	protocol/core/Accountant.sol	124d95807e24b417dff11809d585d04b5e056699978ce371fe9b8ab37f6b7014
BUP	protocol/core/BaseUpgradeablePausable.sol	6d6b1edea4f7db76da7c642aa6187cc71dee03771b2577e8e180ca10582be402
CHG	protocol/core/ConfigHelper.sol	354667d9a6e159cecf6e3cfd78944ae4eadc74169cdd58c3f5c93ffc0254e8b1
COG	protocol/core/ConfigOptions.sol	9b73a4bf7ba62cd344b09f4adc7c53e401b3c46dff2daa3c99089576c3fa80b8
CDG	protocol/core/CreditDesk.sol	e4c3f10db5768dae52165a88de80d76055ba7d47d7e3fdd49da5174ee1229398
CLG	protocol/core/CreditLine.sol	38d2962132ae1960391b8a486b5a27f69b79b7700e44cc6ec47ab88534fac954
FID	protocol/core/Fidu.sol	0054d6e7b6070037308d191a3ae50f58de17984dd3260d9256dc61444e890c15
FLR	protocol/core/FixedLeverageRatioStrategy.sol	982a375582c3376edc3942b022b4250286580ea775668093e0095e1b18dc73a2
GCG	protocol/core/GoldfinchConfig.sol	b7563b2a8b934ec62decdcab774c9925868cc085c674421569014d9774a586a8
GFG	protocol/core/GoldfinchFactory.sol	55db4c1c6bb198fe4beb8a265981e497f5029cec31f6071bbc1818cd656545a9
MTP	protocol/core/MigratedTranchedPool.sol	3e528644f2e01e99b6cb758594cc950b4e07b800724d561f83063b2d5ef6576e
PPG	protocol/core/PauserPausable.sol	16d6c7ae990b3efac99d05ef7b8f2e74f71f94e703dc1de10ba167e1f40f2994
POO	protocol/core/Pool.sol	a55c3fac20427dfb4bdb6a0bb8352c86df9c1ce62c191db029116221d412219b

ID	File	SHA256 Checksum
PTG	protocol/core/PoolTokens.sol	678aa1f24dfad264092cb69058c12f1751314531f418bfe10a4498790fb890f5
SFG	protocol/core/SeniorFund.sol	dd108b548725ea428aaa3674ac9d9ab9d024cad2bca8b83e2ad4e296dec48d20
TPG	protocol/core/TranchedPool.sol	4c734d653756812581d4fd8de25ca786613abe8be5294b1e1a6aab59723c2822
BOR	protocol/periphery/Borrower.sol	211a5044d38d21823b6cf8cd6a211397bbde391f8c001fd1981010e66979da09
TRV	protocol/periphery/TransferRestrictedVault.sol	837b94cbc476a0a426cea11d40df8d7c5c0afc7daee07dc87c987bd8f10c0853
VMG	protocol/periphery/V2Migrator.sol	d8a2fa416ada670c303f9e3eba644a496a4293b7eb077e07a2c0da4486d9a063

Findings



Critical	0 (0.00%)
Major	0 (0.00%)
Medium	2 (4.65%)
Minor	10 (23.26%)
Informational	31 (72.09%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
ACC-01	Inefficient storage read	Gas Optimization	Informational	Resolved
ACC-02	Explicitly returning local variable	Gas Optimization	Informational	Resolved
BOR-01	Lack of validation for function parameter	Logical Issue	Minor	Resolved
BOR-02	Data location for array parameters can be changed from `memory` to `calldata`	Gas Optimization	Informational	Resolved
CLG-01	Lack of validation for function parameter	Logical Issue	Minor	Resolved
CLG-02	Inefficient storage read	Gas Optimization	Informational	Resolved
ERC-01	Unlocked Compiler Version	Language Specific	Informational	Resolved
FLR-01	Unused function parameter	Coding Style	Informational	Declined
FLS-01	Mutability Specifiers Missing	Gas Optimization	Informational	Resolved
FPG-01	SPDX license identifier not provided	Language Specific	Informational	Resolved
GCG-01	Function Visibility Optimization	Gas Optimization	Informational	Resolved

ID	Title	Category	Severity	Status
GCG-02	Function Visibility Optimization	Gas Optimization	Informational	Resolved
GFG-01	Lack of validation for function parameter	Logical Issue	Minor	Resolved
GFG-02	Explicitly returning local variable	Gas Optimization	Informational	Resolved
MTP-01	Comparison with literal `false`	Gas Optimization	Informational	Resolved
MTP-02	Inefficient storage read	Gas Optimization	Informational	Resolved
POL-01	Returned value is not checked	Logical Issue	Minor	Resolved
POO-01	Lack of validation for function parameter	Logical Issue	Minor	Resolved
POO-02	Success status of low level `call` is not checked	Logical Issue	Medium	Resolved
PTG-01	Lack of validation for function parameter	Logical Issue	Minor	Resolved
PTG-02	Explicitly returning local variable	Gas Optimization	Informational	Resolved
PTN-01	Inefficient storage read	Gas Optimization	Informational	Resolved
PTN-02	Data location can be changed from `memory` to `calldata`	Gas Optimization	Informational	Resolved
PTN-03	Redeeming against a `tokenId` can exceed the principal deposited	Logical Issue	Medium	Resolved
SER-01	Incorrect function name	Coding Style	Informational	Resolved
SER-02	Code reusability is not observed	Coding Style	Informational	Resolved
SFG-01	Lack of validation for function parameter	Logical Issue	Minor	Resolved

ID	Title	Category	Severity	Status
SFG-02	Redundant `return` statements	Language Specific	Informational	Declined
SFG-03	Return Variable Utilization	Coding Style	Informational	Resolved
TPG-01	Lack of validation for function parameter	Logical Issue	Minor	Resolved
TPG-02	Possibility of `owner` being a zero address	Logical Issue	Minor	Resolved
TPG-03	Redundant `return` statements	Language Specific	Informational	Declined
TPG-04	Return Variable Utilization	Gas Optimization	Informational	Declined
TPG-05	Inefficient storage read	Gas Optimization	Informational	Resolved
TPG-06	Unnecessary storage read	Gas Optimization	Informational	Declined
TPG-07	Unnecessary addition assignment	Coding Style	Informational	Declined
TPG-08	Inefficient storage read	Gas Optimization	Informational	Resolved
TPG-09	Redundant Statements	Coding Style	Informational	Resolved
TRV-01	Lack of validation for function parameter	Logical Issue	Minor	Resolved
TRV-02	`require` statement can be substituted with modifier	Language Specific	Informational	Resolved
TRV-03	Inefficient storage read	Gas Optimization	Informational	Resolved
TRV-04	Redundant `return` statement	Language Specific	Informational	Declined
TRV-05	Ineffectual `approve` call	Volatile Code	Informational	Resolved

ACC-01 | Inefficient storage read

Category	Severity	Location	Status
Gas Optimization	Informational	../contracts/protocol/core/Accountant.sol (4f356a4): 66, 69	Resolved

Description

The aforementioned lines read storage variable `cl.termEndTime()` multiple times which can be optimized by storing the storage variable in a local variable and then utilizing it.

Recommendation

We advise to store the storage variable in a local variable before usage to save gas cost.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

ACC-02 | Explicitly returning local variable

Category	Severity	Location	Status
Gas Optimization	Informational	../contracts/protocol/core/Accountant.sol (4f356a4): 125, 151	Resolved

Description

The function on the aforementioned line explicitly returns local variable which increases overall cost of gas.

Recommendation

Since named return variables can be declared in the signature of a function, consider refactoring to remove the local variable declaration and explicit return statement in order to reduce the overall cost of gas.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

BOR-01 | Lack of validation for function parameter

Category	Severity	Location	Status
Logical Issue	Minor	../contracts/protocol/periphery/Borrower.sol (223e7d9): 38	Resolved

Description

The function parameter `_config` on the aforementioned line is used to initialize the contract's state variable yet it is not validated against zero address value. If it is passed as zero value then it will result in unwanted state of the contract.

Recommendation

We advise to validate the function parameter `_config` against zero address.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

BOR-02 | Data location for array parameters can be changed from `memory` to `calldata`

Category	Severity	Location	Status
Gas Optimization	Informational	../contracts/protocol/periphery/Borrower.sol (223e7d9): 159, 173	Resolved

Description

The linked function is declared as `external` and contains array function arguments with `memory` as data location.

Recommendation

We advise that the data location array types parameters of the aforementioned function be changed from `memory` to `calldata` to save gas cost associated with copying of parameters from `memory` to `calldata`. Additionally, the function `swapOnOneInch` on `L226` can have the data location for its parameter `exchangeDistribution` from `memory` to `calldata` as this parameter is received from both of the aforementioned functions.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

CLG-01 | Lack of validation for function parameter

Category	Severity	Location	Status
Logical Issue	Minor	../contracts/protocol/core/CreditLine.sol (4f356a4): 49	Resolved

Description

The function parameter `_config` on the aforementioned line is used to initialize the contract's state variable yet it is not validated against zero address value. If it is passed as zero value then it will result in unwanted state of the contract.

Recommendation

We advise to validate the function parameter `_config` against zero address.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

CLG-02 | Inefficient storage read

Category	Severity	Location	Status
Gas Optimization	Informational	../contracts/protocol/core/CreditLine.sol (4f356a4): 293, 296, 298, 300	Resolved

Description

The aforementioned lines read storage variable `nextDueTime` multiple times which can be optimized by storing the storage variable in a local variable and then utilizing it.

Recommendation

We advise to store the storage variable in a local variable before usage to save gas cost.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

ERC-01 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	../contracts/external/ERC721PresetMinterPauserAutoId.sol (4f356a4): 13	Resolved

Description

The contract has unlocked compiler version. An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to an ambiguity when debugging as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

We advise that the compiler version is instead locked at the lowest version possible that the contract can be compiled at. For example, for `^0.6.12`, the version can be locked at `0.6.12`.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

FLR-01 | Unused function parameter

Category	Severity	Location	Status
Coding Style	Informational	../contracts/protocol/core/FixedLeverageRatioStrategy.sol (4f356a4): 27	Declined

Description

The function `invest` on the aforementioned line has unused parameter `fund` of type `IFund`.

Recommendation

We advise to either utilize this parameter or remove it from the function signature to increase code legibility.

Alleviation

No alleviations.

FLS-01 | Mutability Specifiers Missing

Category	Severity	Location	Status
Gas Optimization	Informational	../contracts/protocol/core/FixedLeverageRatioStrategy.sol (223e7d9) : 14	Resolved

Description

The linked variables are assigned to only once, either during their contract-level declaration or during the `constructor`'s execution.

Recommendation

For the former, we advise that the `constant` keyword is introduced in the variable declaration to greatly optimize the gas cost involved in utilizing the variable. For the latter, we advise that the `immutable` mutability specifier is set at the variable's contract-level declaration to greatly optimize the gas cost of utilizing the variables. Please note that the `immutable` keyword only works in Solidity versions `v0.6.5` and up.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

FPG-01 | SPDX license identifier not provided

Category	Severity	Location	Status
Language Specific	Informational	../contracts/external/FixedPoint.sol (223e7d9): 1	Resolved

Description

SPDX license identifier not provided in the aforementioned file.

Recommendation

Consider adding one before deployment.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

GCG-01 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	../contracts/protocol/core/GoldfinchConfig.sol (4f356a4): 108	Resolved

Description

The linked function is declared as `public`, contains array function arguments and is not invoked in any of the contract's contained within the project's scope.

Recommendation

We advise that the functions' visibility specifiers are set to `external` and the array-based arguments change their data location from `memory` to `calldata`, optimizing the gas cost of the function.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

GCG-02 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	../contracts/protocol/core/GoldfinchConfig.sol (4f356a4): 118	Resolved

Description

The linked function is declared as `public`, contains array function arguments and is not invoked in any of the contract's contained within the project's scope.

Recommendation

We advise that the functions' visibility specifiers are set to `external` and the array-based arguments change their data location from `memory` to `calldata`, optimizing the gas cost of the function.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

GFG-01 | Lack of validation for function parameter

Category	Severity	Location	Status
Logical Issue	Minor	../contracts/protocol/core/GoldfinchFactory.sol (4f356a4): 26	Resolved

Description

The function parameters `owner` and `_config` on the aforementioned line is used to initialize the contract's state variables yet they are not validated against zero address value. If they are passed as zero address value then it will result in unwanted state of the contract.

Recommendation

We advise to validate the function parameters `owner` and `_config` against zero address.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

GFG-02 | Explicitly returning local variable

Category	Severity	Location	Status
Gas Optimization	Informational	../contracts/protocol/core/GoldfinchFactory.sol (4f356a4): 64, 97	Resolved

Description

The functions on the aforementioned lines explicitly return local variable which increases overall cost of gas.

Recommendation

Since named return variables can be declared in the signature of a function, consider refactoring to remove the local variable declaration and explicit return statement in order to reduce the overall cost of gas.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

MTP-01 | Comparison with literal `false`

Category	Severity	Location	Status
Gas Optimization	Informational	../contracts/protocol/core/MigratedTranchedPool.sol (4f356a4): 23	Resolved

Description

The comparison with literal `false` on the aforementioned can be substituted with negation of expression to increase legibility of codebase.

Recommendation

We advise to substitute the comparison with literal `false` with the negation of expression.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

MTP-02 | Inefficient storage read

Category	Severity	Location	Status
Gas Optimization	Informational	../contracts/protocol/core/MigratedTranchedPool.sol (4f356a4): 25~32, 38	Resolved

Description

The aforementioned lines read storage variable `creditLine` multiple times which can be optimized by storing the storage variable in a local variable and then utilizing it.

Recommendation

We advise to store the storage variable in a local variable before usage to save gas cost.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

POL-01 | Returned value is not checked

Category	Severity	Location	Status
Logical Issue	Minor	../contracts/protocol/core/Pool.sol (223e7d9): 194	Resolved

Description

The returned value of the function call `doUSDCTransfer` on the aforementioned line is not asserted to be `true`.

Recommendation

We advise to introduce a check ensuring that the returned value of the function call `doUSDCTransfer` is `true`.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

POO-01 | Lack of validation for function parameter

Category	Severity	Location	Status
Logical Issue	Minor	../contracts/protocol/core/Pool.sol (4f356a4): 35	Resolved

Description

The function parameters `owner` and `_config` on the aforementioned line is used to initialize the contract's state variables yet they are not validated against zero address value. If they are passed as zero address value then it will result in unwanted state of the contract.

Recommendation

We advise to validate the function parameters `owner` and `_config` against zero address.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

POO-02 | Success status of low level `call` is not checked

Category	Severity	Location	Status
Logical Issue	Medium	../contracts/protocol/core/Pool.sol (4f356a4): 202, 212	Resolved

Description

The low level `call` function execution on the aforementioned lines return the status of the call as first variable in the returned tuple. The status of the `call` is not asserted to be `true` which will treat the low level call as success even in the event it reverts.

Recommendation

We advise to check the status of the low level `call` to be `true`.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

PTG-01 | Lack of validation for function parameter

Category	Severity	Location	Status
Logical Issue	Minor	../contracts/protocol/core/PoolTokens.sol (4f356a4): 59	Resolved

Description

The function parameters `owner` and `_config` on the aforementioned line is used to initialize the contract's state variables yet they are not validated against zero address value. If they are passed as zero address value then it will result in unwanted state of the contract.

Recommendation

We advise to validate the function parameters `owner` and `_config` against zero address.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

PTG-02 | Explicitly returning local variable

Category	Severity	Location	Status
Gas Optimization	Informational	../contracts/protocol/core/PoolTokens.sol (4f356a4): 83, 163	Resolved

Description

The functions on the aforementioned lines explicitly return local variable which increases overall cost of gas.

Recommendation

Since named return variables can be declared in the signature of a function, consider refactoring to remove the local variable declaration and explicit return statement in order to reduce the overall cost of gas.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

PTN-01 | Inefficient storage read

Category	Severity	Location	Status
Gas Optimization	Informational	../contracts/protocol/core/PoolTokens.sol (223e7d9): 109	Resolved

Description

The aforementioned lines read storage variable `token.pool` multiple times which can be optimized by storing the storage variable in a local variable and then utilizing it.

Recommendation

We advise to store the storage variable in a local variable before usage to save gas cost.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

PTN-02 | Data location can be changed from `memory` to `calldata`

Category	Severity	Location	Status
Gas Optimization	Informational	../contracts/protocol/core/PoolTokens.sol (223e7d9): 163	Resolved

Description

The function `createToken` on the aforementioned line specify data location of parameter `params` as `memory`. Since this parameter is only received from the function `mint` which specifies its data location as `calldata`. The data location on `L163` can be changed to `calldata` to save gas cost associated with unnecessary copying of the parameter from `calldata` to `memory`.

Recommendation

We advise to change the data location of parameter `params` from `memory` to `calldata`.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

PTN-03 | Redeeming against a `tokenId` can exceed the principal deposited

Category	Severity	Location	Status
Logical Issue	Medium	../contracts/protocol/core/PoolTokens.sol (223e7d9): 103	Resolved

Description

The function `redeem` on the aforementioned line updates a token with the principal and interest that is redeemed. The function does not ensure that the principal amount that is being redeemed against a particular `tokenId` does not exceed the principal amount that is deposited against the same `tokenId`. Although, the `redeemableInterestAndPrincipal` function in TranchedPool contract ensures the redeemed amounts do not exceed the principal deposit, we still suggest to introduce this check within the `redeem` function observing separation of concerns design principal.

Recommendation

We advise to introduce a check to ensure that the total redeemed amount does not exceed the principal deposited against a `tokenId`.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

SER-01 | Incorrect function name

Category	Severity	Location	Status
Coding Style	Informational	../contracts/library/SafeERC20Transfer.sol (4f356a4): 33, 45	Resolved

Description

The functions on the aforementioned lines are named `safeTransfer` yet the underlying code of function performs `transferFrom` operation on the provided `ERC20` contract.

Recommendation

We advise to rename the functions on the aforementioned lines to `safeTransferFrom` to increase the legibility of codebase.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

SER-02 | Code reusability is not observed

Category	Severity	Location	Status
Coding Style	Informational	../contracts/library/SafeERC20Transfer.sol (4f356a4): 55, 64	Resolved

Description

The function `safeApprove` on `L55` can utilize the function on `L64` by passing it the error message instead of directly calling `approve` function on the `ERC20` contract to observe code reusability.

Recommendation

We advise to rectify the function `safeApprove` on `L55` such that it utilizes the `safeApprove` function on `L64`.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

SFG-01 | Lack of validation for function parameter

Category	Severity	Location	Status
Logical Issue	Minor	../contracts/protocol/core/SeniorFund.sol (223e7d9): 38	Resolved

Description

The function parameters `owner` and `_config` on the aforementioned line is used to initialize the contract's state variables yet they are not validated against zero address value. If they are passed as zero address value then it will result in unwanted state of the contract.

Recommendation

We advise to validate the function parameters `owner` and `_config` against zero address.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

SFG-02 | Redundant `return` statements

Category	Severity	Location	Status
Language Specific	Informational	../contracts/protocol/core/SeniorFund.sol (223e7d9): 72, 354	Declined

Description

The `return` statements on the aforementioned lines are redundant as the functions make use of named returning variables in their signatures.

Recommendation

We recommend to remove the redundant `return` statements on the aforementioned lines to increase the legibility of codebase.

Alleviation

No alleviations were considered.

SFG-03 | Return Variable Utilization

Category	Severity	Location	Status
Coding Style	Informational	../contracts/protocol/core/SeniorFund.sol (223e7d9): 89, 113, 273	Resolved

Description

The linked function declarations contain explicitly named `return` variables that are not utilized within the function's code block.

Recommendation

We advise that the linked variables are either utilized or omitted from the declaration.

Alleviation

No alleviations were considered.

TPG-01 | Lack of validation for function parameter

Category	Severity	Location	Status
Logical Issue	Minor	../contracts/protocol/core/TranchedPool.sol (4f356a4): 57	Resolved

Description

The function parameters `_borrower` and `_config` on the aforementioned line is used to initialize the contract's state variables yet they are not validated against zero address value. If they are passed as zero address value then it will result in unwanted state of the contract.

Recommendation

We advise to validate the function parameters `_borrower` and `_config` against zero address.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

TPG-02 | Possibility of `owner` being a zero address

Category	Severity	Location	Status
Logical Issue	Minor	../contracts/protocol/core/TranchedPool.sol (4f356a4): 68~69	Resolved

Description

The `owner` address retrieved from the `config` contract has a possibility of being a zero address if it is not set in the config. It can result in unwanted state of the contract as the retrieved `owner` address is set the as the `owner` of the `TranchedPool` contract on `L69`.

Recommendation

We advise to validate `owner` variable against zero address.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

TPG-03 | Redundant `return` statements

Category	Severity	Location	Status
Language Specific	Informational	../contracts/protocol/core/TranchedPool.sol (4f356a4): 117, 466, 574	Declined

Description

The `return` statements on the aforementioned lines are redundant as the functions make use of named returning variables in their signatures.

Recommendation

We recommend to remove the redundant `return` statements on the aforementioned lines to increase the legibility of codebase.

Alleviation

No alleviations were considered.

TPG-04 | Return Variable Utilization

Category	Severity	Location	Status
Gas Optimization	Informational	../contracts/protocol/core/TranchedPool.sol (4f356a4): 127, 145, 165, 409, 428	Declined

Description

The linked function declarations contain explicitly named `return` variables that are not utilized within the function's code block.

Recommendation

We advise that the linked variables are either utilized or omitted from the declaration.

Alleviation

No alleviations were considered.

TPG-05 | Inefficient storage read

Category	Severity	Location	Status
Gas Optimization	Informational	../contracts/protocol/core/TranchedPool.sol (4f356a4): 303~313, 318	Resolved

Description

The aforementioned lines read storage variable `creditLine` multiple times which can be optimized by storing the storage variable in a local variable and then utilizing it.

Recommendation

We advise to store the storage variable in a local variable before usage to save gas cost.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

TPG-06 | Unnecessary storage read

Category	Severity	Location	Status
Gas Optimization	Informational	../contracts/protocol/core/TranchedPool.sol (4f356a4): 347	Declined

Description

The aforementioned line reads `creditLine` from contract's storage which can be substituted with the utilization of local variable `newCl` to save gas cost.

Recommendation

We advise to substitute the storage read of `creditLine` with local variable `newCl`.

Alleviation

No alleviations were considered.

TPG-07 | Unnecessary addition assignment

Category	Severity	Location	Status
Coding Style	Informational	../contracts/protocol/core/TranchedPool.sol (4f356a4): 515	Declined

Description

The aforementioned line performs addition assignment of `totalReserveAmount = totalReserveAmount.add(reserveDeduction);` which can be substituted with direct assignment to increase code legibility as `totalReserveAmount` is guaranteed to be zero before the aforementioned addition assignment.

Recommendation

We advise to substitute the addition assignment with plain assignment on the aforementioned line.

Alleviation

No alleviations were considered.

TPG-08 | Inefficient storage read

Category	Severity	Location	Status
Gas Optimization	Informational	../contracts/protocol/core/TranchedPool.sol (4f356a4): 650	Resolved

Description

The aforementioned lines read storage variable `tranche.principalSharePrice` and `tranche.interestSharePrice` multiple times which can be optimized by storing the storage variable in a local variable and then utilizing it.

Recommendation

We advise to store the storage variable in a local variable before usage to save gas cost.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

TPG-09 | Redundant Statements

Category	Severity	Location	Status
Coding Style	Informational	../contracts/protocol/core/TranchedPool.sol (4f356a4): 730~733	Resolved

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

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

TRV-01 | Lack of validation for function parameter

Category	Severity	Location	Status
Logical Issue	Minor	../contracts/protocol/periphery/TransferRestrictedVault.sol (223e7d9): 52	Resolved

Description

The function parameters `owner` and `_config` on the aforementioned line is used to initialize the contract's state variables yet they are not validated against zero address value. If they are passed as zero address value then it will result in unwanted state of the contract.

Recommendation

We advise to validate the function parameters `owner` and `_config` against zero address.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

TRV-02 | `require` statement can be substituted with modifier

Category	Severity	Location	Status
Language Specific	Informational	../contracts/protocol/periphery/TransferRestrictedVault.sol (223e7d9): 130, 143, 158	Resolved

Description

The `require` statements on the aforementioned lines can be substituted with modifier to increase the legibility of codebase.

Recommendation

We advise to substitute `require` statements on the aforementioned lines with modifier.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

TRV-03 | Inefficient storage read

Category	Severity	Location	Status
Gas Optimization	Informational	../contracts/protocol/periphery/TransferRestrictedVault.sol (223e7d9): 135, 137, 146, 148	Resolved

Description

The aforementioned lines read storage variable `fiduPosition.amount` multiple times which can be optimized by storing the storage variable in a local variable and then utilizing it.

Recommendation

We advise to store the storage variable in a local variable before usage to save gas cost.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

TRV-04 | Redundant `return` statement

Category	Severity	Location	Status
Language Specific	Informational	../contracts/protocol/periphery/TransferRestrictedVault.sol (223e7d9): 171	Declined

Description

The `return` statement on the aforementioned line is redundant as the function makes use of named returning variables in its signatures.

Recommendation

We recommend to remove the redundant `return` statement on the aforementioned line to increase the legibility of codebase.

Alleviation

No alleviations were considered.

TRV-05 | Ineffectual `approve` call

Category	Severity	Location	Status
Volatile Code	Informational	../contracts/protocol/periphery/TransferRestrictedVault.sol (223e7d9): 218	Resolved

Description

The aforementioned line performs approval for an address which receives the token on `L219`. As the transfer of token on `L219` clears the approval and sets it to address zero rendering the approval on `L218` ineffectual.

Recommendation

We advise to remove the ineffectual `approve` call on the aforementioned line.

Alleviation

Alleviations are applied as of commit hash `1b64fcd2ff8c435d698e433482f635023cadbe19`.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings 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.

Logical Issue

Logical Issue findings 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.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

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.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.