CERTIK

Harvest Finance

Security Assessment

November 15th, 2020

For :
Harvest Finance

By :
Alex Papageorgiou @ CertiK
alex.papageorgiou@certik.org

Angelos Apostolidis @ CertiK
angelos.apostolidis@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 indeed 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	Harvest Finance
Description	A yield farming protocol with multiple strategies.
Platform	Ethereum; Solidity, Yul
Codebase	GitHub Repository
Commit(s)	ffd95f02d19ddf878360059e4fa8a4cebb792c2a

Audit Summary

Delivery Date	Oct. 2nd, 2020
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	2
Timeline	Sep. 16, 2020 - Oct. 2 2020

Vulnerability Summary

Total Issues	74
Total Critical	0
Total Major	0
Total Minor	1
Total Informational	73

Executive Summary

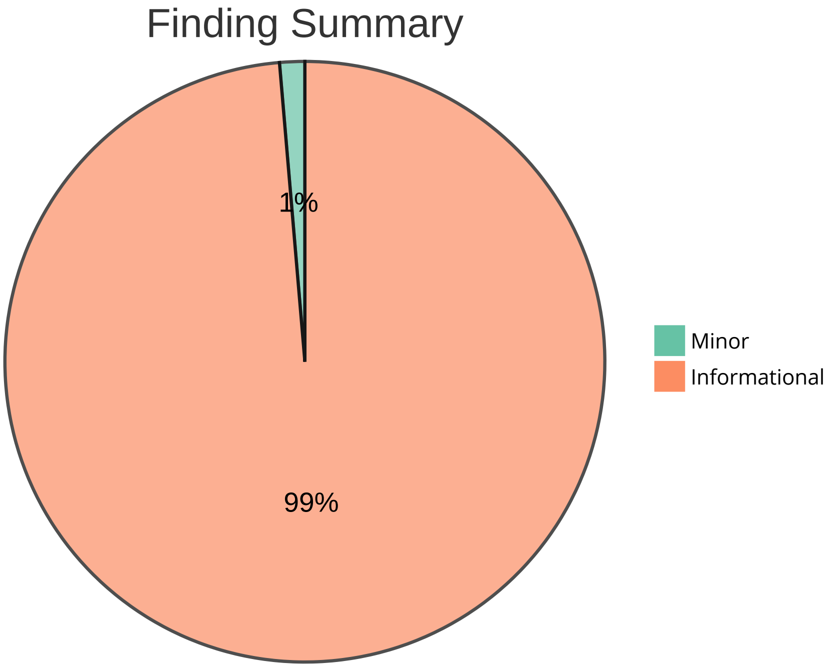
We were approached by Harvest to conduct an audit of their yield farming protocol, Harvest Finance. Our audit was able to pinpoint numerous sections where the codebase could be improved optimization-wise, however only a single minor vulnerability was pinpointed that was clarified by the developers to be desired functionality and should not result in an exploitable attack vector.

Files In Scope

ID	Contract	Location
STR	Storage.sol	contracts/Storage.sol
CTE	Controllable.sol	contracts/Controllable.sol
GVE	Governable.sol	contracts/Governable.sol
DMT	DelayMinter.sol	contracts/DelayMinter.sol
DPH	DepositHelper.sol	contracts/DepositHelper.sol
FRF	FeeRewardForwarder.sol	contracts/FeeRewardForwarder.sol
HRW	HardRewards.sol	contracts/HardRewards.sol
NHP	NotifyHelper.sol	contracts/NotifyHelper.sol
RWP	RewardPool.sol	contracts/RewardPool.sol
RWT	RewardToken.sol	contracts/RewardToken.sol
VLT	Vault.sol	contracts/Vault.sol
CTL	Controller.sol	contracts/Controller.sol
CTI	CTokenInterfaces.sol	contracts/compound/CTokenInterfaces.sol
COI	ComptrollerInterface.sol	contracts/compound/ComptrollerInterface.sol
IRM	InterestRateModel.sol	contracts/compound/InterestRateModel.sol
ICT	IController.sol	contracts/hardworkInterface/IController.sol
IRP	IRewardPool.sol	contracts/hardworkInterface/IRewardPool.sol
IST	IStrategy.sol	contracts/hardworkInterface/IStrategy.sol
IVT	IVault.sol	contracts/hardworkInterface/IVault.sol
WTH	WETH9.sol	contracts/weth/WETH9.sol
VDI	VaultDAI.sol	contracts/vaults/VaultDAI.sol
VUC	VaultUSDC.sol	contracts/vaults/VaultUSDC.sol
VUT	VaultUSDT.sol	contracts/vaults/VaultUSDT.sol
VYV	VaultYCRV.sol	contracts/vaults/VaultYCRV.sol
IWT	IWETH.sol	contracts/uniswap/interfaces/IWETH.sol
IUM	IUniswapV2Migrator.sol	contracts/uniswap/interfaces/IUniswapV2Migrator.sol
IUP	IUniswapV2Pair.sol	contracts/uniswap/interfaces/IUniswapV2Pair.sol
IUR	IUniswapV2Router01.sol, IUniswapV2Router02.sol	contracts/uniswap/interfaces/IUniswapV2Router01.sol, contracts/uniswap/interfaces/IUniswapV2Router02.sol
IUE	IUniswapV1Exchange.sol	contracts/uniswap/interfaces/V1/IUniswapV1Exchange.sol
IUF	IUniswapV1Factory.sol	contracts/uniswap/interfaces/V1/IUniswapV1Factory.sol
PNF	ProfitNotifier.sol	contracts/strategies/ProfitNotifier.sol
RPN	RewardTokenProfitNotifier.sol	contracts/strategies/RewardTokenProfitNotifier.sol
SRS	SNXRewardStrategy.sol	contracts/strategies/SNXRewards/SNXRewardStrategy.sol
SRU	SNXRewardUniLPStrategy.sol	contracts/strategies/SNXRewards/SNXRewardUniLPStrategy.sol
SRI	SNXRewardInterface.sol	contracts/strategies/SNXRewards/SNXRewardInterface.sol
CMI	CompoundInteractor.sol	contracts/strategies/compound/CompoundInteractor.sol
CCT	CompleteCToken.sol	contracts/strategies/compound/CompleteCToken.sol

ID	Contract	Location
WCS	WETHCreamNoFoldStrategy.sol	contracts/strategies/compound/WETHCreamNoFoldStrategy.sol
PCV	PriceConvertor.sol	contracts/strategies/curve/PriceConvertor.sol
SST	CRVStrategyStable.sol	contracts/strategies/curve/CRVStrategyStable.sol
SYC	CRVStrategyYCRV.sol	contracts/strategies/curve/CRVStrategyYCRV.sol
SSW	CRVStrategySwerve.sol	contracts/strategies/curve/CRVStrategySwerve.sol
SWB	CRVStrategyWRenBTC.sol	contracts/strategies/curve/CRVStrategyWRenBTC.sol
SSM	CRVStrategyStableMainnet.sol	contracts/strategies/curve/CRVStrategyStableMainnet.sol
SRB	CRVStrategyRENBTCMainnet.sol	contracts/strategies/curve/CRVStrategyRENBTCMainnet.sol
SSD	CRVStrategySwerveDAIMainnet.sol	contracts/strategies/curve/CRVStrategySwerveDAIMainnet.sol
SSU	CRVStrategySwerveUSDCMainnet.sol	contracts/strategies/curve/CRVStrategySwerveUSDCMainnet.sol
SWM	CRVStrategyWBTCMainnet.sol	contracts/strategies/curve/CRVStrategyWBTCMainnet.sol
SYM	CRVStrategyYCRVMainnet.sol	contracts/strategies/curve/CRVStrategyYCRVMainnet.sol
GAU	Gauge.sol	contracts/strategies/curve/interfaces/Gauge.sol
ICF	ICurveFi.sol	contracts/strategies/curve/interfaces/ICurveFi.sol
ICW	ICurveFiWbtc.sol	contracts/strategies/curve/interfaces/ICurveFiWbtc.sol
IPC	IPriceConvertor.sol	contracts/strategies/curve/interfaces/IPriceConvertor.sol
ISF	ISwerveFi.sol	contracts/strategies/curve/interfaces/ISwerveFi.sol
YVL	yVault.sol	contracts/strategies/curve/interfaces/yVault.sol

Findings



ID	Title	Type	Severity
STR-01	Pull-over-Push Pattern	Language Specific	Informational
CTE-01	Consecutive External Calls	Optimization	Informational
DMT-01	Variable Mutability Specifiers	Optimization	Informational
DMT-02	Variable Visibility Specifiers	Language Specific	Informational
DMT-03	Event Optimization	Optimization	Informational
DMT-04	`require` Statement Optimization	Optimization	Informational
DMT-05	Redundant Assignment	Optimization	Informational
DMT-06	Mapping Lookup Optimization	Optimization	Informational
DMT-07	Checks-Effects-Interactions Pattern	Logical	Minor
DMT-08	Redundant SafeMath	Optimization	Informational
DMT-09	Typecasting Optimization	Optimization	Informational
DMT-10	Typecasting Optimization	Language Specific	Informational
DPH-01	Function Input Optimization	Language Specific	Informational
DPH-02	Assignment Optimization	Optimization	Informational
FRF-01	Function Input Optimization	Language Specific	Informational
FRF-02	Potential Underflow	Mathematical	Informational
FRF-03	Inefficient Conditional	Optimization	Informational
FRF-04	Function Side-Effect	Language Specific	Informational
HRW-01	Assignment Optimization	Optimization	Informational
HRW-02	Conditional Optimization	Optimization	Informational
HRW-03	Inefficient Greater-Than Comparison w/ Zero	Optimization	Informational
HRW-04	Redundant Syntax	Syntax	Informational
NHP-01	Function Input Optimization	Language Specific	Informational

ID	Title	Type	Severity
NHP-02	Inefficient Greater-Than Comparison w/ Zero	Optimization	Informational
NHP-03	Variable Re-use	Optimization	Informational
RWP-01	Library Consistency	Syntax	Informational
RWP-02	Inefficient Greater-Than Comparison w/ Zero	Optimization	Informational
VLT-01	Unconventional Syntax	Syntax	Informational
VLT-02	Inefficient Greater-Than Comparison w/ Zero	Optimization	Informational
VLT-03	Conditional Consistency	Codebase Consistency	Informational
VLT-04	Redundant SafeMath	Mathematical	Informational
VLT-05	Function Invocation Re-use	Optimization	Informational
CTL-01	Incorrect Error Message	Optimization	Informational
CTL-02	Contract Bytecode Optimization	Optimization	Informational
CTL-03	Inefficient Greater-Than Comparison w/ Zero	Optimization	Informational
PNF-01	`require` Consistency	Codebase Consistency	Informational
PNF-02	Typecasting Optimization	Optimization	Informational
RPN-01	Variable Mutability Specifiers	Optimization	Informational
RPN-02	`require` Consistency	Codebase Consistency	Informational
RPN-03	Inefficient Greater-Than Comparison w/ Zero	Optimization	Informational
SRS-01	Redundant SafeMath Operation	Optimization	Informational
SRS-02	Uniswap Conformity	Language Specific	Informational
SRS-03	Variable Re-use	Optimization	Informational
SRS-04	Inefficient Greater-Than Comparison w/ Zero	Optimization	Informational
SRU-01	Visibility Specifier Missing	Optimization	Informational

ID	Title	Type	Severity
SRU-02	Inefficient Greater-Than Comparison w/ Zero	Optimization	Informational
SRU-03	Redundant Typecasting	Optimization	Informational
SRU-04	Incorrect Comment	Documentation Conformity	Informational
CMI-01	Variable Mutability and Type	Optimization	Informational
CMI-02	Inefficient Greater-Than Comparison w/ Zero	Optimization	Informational
WCS-01	Inefficient Greater-Than Comparison w/ Zero	Optimization	Informational
PCV-01	Variable Mutability Specifier	Optimization	Informational
SSM-01	`revert` Statement	Syntactic	Informational
SST-01	Visibility Specifier Missing	Syntactic	Informational
SST-02	Variable Mutability Specifier	Optimization	Informational
SST-03	Inefficient Greater-Than Comparison w/ Zero	Optimization	Informational
SST-04	Unconventional Syntax	Syntactic	Informational
SST-05	Redundant SafeMath Operation	Optimization	Informational
SST-06	Variable Re-use	Optimization	Informational
SST-07	Variable Re-use	Optimization	Informational
SYC-01	Variable Mutability Specifier	Optimization	Informational
SYC-02	Variable Declaration Misuse	Optimization / Syntactical	Informational
SYC-03	Variable Re-use	Optimization	Informational
SYC-04	Inefficient Greater-Than Comparison w/ Zero	Optimization	Informational
SYC-05	Literal over Memory	Optimization	Informational
SSW-01	Variable Visibility Specifier	Optimization	Informational

ID	Title	Type	Severity
SSW-02	Variable Mutability Specifier	Optimization	Informational
SSW-03	Inefficient Greater-Than Comparison w/ Zero	Optimization	Informational
SSW-04	Unconventional Syntax	Syntactic	Informational
SWB-01	Variable Visibility Specifier	Optimization	Informational
SWB-02	Variable Mutability Specifier	Optimization	Informational
SWB-03	Inefficient Greater-Than Comparison w/ Zero	Optimization	Informational
SWB-04	Unconventional Syntax	Syntactic	Informational

STR-01: Pull-over-Push Pattern

Type	Severity	Location
Language Specific	Informational	[Storage.sol L17-L25]()

Description:

A secure Solidity pattern is to update sensitive variables utilizing a pull-over-push pattern whereby instead of overriding existing values, a proposed value is set and then the owner of the proposed value, usually the account an address points to, needs to accept the proposal for it to override the previous value. This prevents against mis-types and accidental transactions as software is prone to error and overriding sensitive variables, such as owners of contracts, is at times irreversible.

Recommendation:

We advise that the `set` prefixed functions for governance and the controller are instead replaced by propose and accept functions respectively.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

CTE-01: Consecutive External Calls

Type	Severity	Location
Optimization	Informational	Controllable.sol L15-L19

Description:

The conditional of L16 conducts to external calls on `store` by first calling `isController` and then calling `isGovernance`.

Recommendation:

As both calls are `view` functions that rely on the same input, `msg.sender`, it is possible to instead code a single function on `store` that retrieves whetehr the `address` is a controller or the governance as this would result in a single external function call rather than two.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

DMT-01: Variable Mutability Specifiers

Type	Severity	Location
Optimization	Informational	DelayMinter.sol L46-L49

Description:

The linked statements contain contract-level variable declarations and assignments, the variables of which are never assigned to elsewhere in the codebase.

Recommendation:

We advise that the mutability specifier `constant` is imposed on those variables to greatly reduce the gas cost incurred by utilizing them.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

DMT-02: Variable Visibility Specifiers

Type	Severity	Location
Language Specific	Informational	DelayMinter.sol L57

Description:

The linked variable declaration is missing a visibility specifier.

Recommendation:

We advise that a proper visibility specifier is set for the linked variable.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

DMT-03: Event Optimization

Type	Severity	Location
Optimization	Informational	DelayMinter.sol L62

Description:

The linked `MintingAnnounced` event declaration is missing any `indexed` variables.

Recommendation:

We advise that the `id` of the minting is `indexed` to greatly speed up lookup operations on blockchain nodes.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

DMT-04: `require` Statement Optimization

Type	Severity	Location
Optimization	Informational	DelayMinter.sol L71, L73, L75, L77

Description:

The linked `require` statements ensure that the input variables of the `constructor` are not zeroed out. However, they access the contract's newly stored variables instead of relying on the variables already in memory.

Recommendation:

We advise that the `require` statements utilize the underscore (`_`) prefixed variable counterparts to ensure they do not access the contract's `storage` optimizing gas cost of deployment.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

DMT-05: Redundant Assignment

Type	Severity	Location
Optimization	Informational	DelayMinter.sol L78

Description:

In Solidity, all variables are by default initialized to their zeroed-out variable type, meaning that explicit assignment of such zeroed variables on a variable's initialization are redundant.

Recommendation:

We advise that the linked manual zeroing statements are omitted.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

DMT-06: Mapping Lookup Optimization

Type	Severity	Location
Optimization	Informational	DelayMinter.sol L102-L120

Description:

The function `executeMint` is utilizing the result of the `announcements[_id]` struct lookup thrice instead of storing it in an in-memory variable since all three members of the `MintingAnnouncement` struct are read.

Recommendation:

We advise that a `MintingAnnouncement memory` declaration is introduced at the beginning of the function that is subsequently read from for all purposes.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

DMT-07: Checks-Effects-Interactions Pattern

Type	Severity	Location
Logical	Minor	DelayMinter.sol L102-L120

Description:

The function `executeMint` is performing an arbitrary `mint` operation on an `ERC20Mintable` token. As certain tokens contain additional functionality that inform the recipient of a minting operation, it would be possible for a re-entrancy attack to be executed here as the minting announcements are deleted after the token is minted.

Recommendation:

We advise that the `delete` statement of L119 is moved before the first `mint` external invocation of L113.

Alleviation:

The reward token utilized in this context is $FARM, a token designed by Harvest Finance, and as such this attack vector is inexistent and the exhibit is nullified.

DMT-08: Redundant SafeMath

Type	Severity	Location
Optimization	Informational	DelayMinter.sol L112

Description:

The linked statement conducts logically-safe subtractions by wrapping them in the `SafeMath` library.

Recommendation:

These subtractions can be safely performed without being wrapped by the `SafeMath` library as the subtracted variables are guaranteed to be less than the `amount` due to the multiplication and divisions always resulting in a value less than `amount`. Additionally, the `div` operations that precede this statement can also be omitted and be conducted in their raw `/` format as `SafeMath`'s `div` simply ensures the divisor is not equal to `0` which is always the case on this contract.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

DMT-09: Typecasting Optimization

Type	Severity	Location
Optimization	Informational	DelayMinter.sol L51

Description:

The linked `token` declaration is never utilized as an `address` type and is instead always casted to an `ERC20Mintable` interface.

Recommendation:

We advise that the `address` is directly stored as the interface's type, `ERC20Mintable`.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

DMT-10: Typecasting Optimization

Type	Severity	Location
Language Specific	Informational	DelayMinter.sol L129-L139

Description:

A secure Solidity pattern is to update sensitive variables utilizing a pull-over-push pattern whereby instead of overriding existing values, a proposed value is set and then the owner of the proposed value, usually the account an address points to, needs to accept the proposal for it to override the previous value. This prevents against mis-types and accidental transactions as software is prone to error and overriding sensitive variables, such as owners of contracts, is at times irreversible.

Recommendation:

We advise that the `set` prefixed functions for the team and the operator addresses are instead replaced by propose and accept functions respectively.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

DPH-01: Function Input Optimization

Type	Severity	Location
Language Specific	Informational	DepositHelper.sol L38

Description:

When function calls are not utilized elsewhere internally in the codebase and contain array inputs, it is highly advisable to set them as `external` and instead store the input arrays in `calldata` rather than `memory` to greatly optimize the gas cost involved in invoking those functions.

Recommendation:

We advise the above pattern is applied on the linked function.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

DPH-02: Assignment Optimization

Type	Severity	Location
Optimization	Informational	DepositHelper.sol L44

Description:

The linked `require` statement conducts an external function call to the `controller` getter function of `store` on each iteration of the `for` loop.

Recommendation:

We advise that the result of this external call is instead stored outside the `for` loop to ensure only a single external call is conducted to `store` and the gas cost of the function is optimized.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

FRF-01: Function Input Optimization

Type	Severity	Location
Language Specific	Informational	FeeRewardForwarder.sol L54

Description:

When function calls are not utilized elsewhere internally in the codebase and contain array inputs, it is highly advisable to set them as `external` and instead store the input arrays in `calldata` rather than `memory` to greatly optimize the gas cost involved in invoking those functions.

Recommendation:

We advise the above pattern is applied on the linked function.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

FRF-02: Potential Underflow

Type	Severity	Location
Mathematical	Informational	FeeRewardForwarder.sol L54

Description:

The linked `require` conditional conducts a raw subtraction of the input array's `length` with the literal `1`. This can lead to arrays with a single item to execute the function "properly" and for arrays with no members to fail without a proper reason due to the overflow and subsequent out-of-bounds access of L58.

Recommendation:

We advise that a proper `require` check is imposed that ensures the length of the path is greater-than the literal `1`.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

FRF-03: Inefficient Conditional

Type	Severity	Location
Optimization	Informational	FeeRewardForwarder.sol L75

Description:

If FRF-02 is integrated, it is possible to convert this comparison to an inequality comparison with the literal `0` which is more optimal than greater-than comparisons.

Recommendation:

N/A.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

FRF-04: Function Side-Effect

Type	Severity	Location
Language Specific	Informational	FeeRewardForwarder.sol L77

Description:

The linked statement utilizes the balance of the contract rather than accepting an input variable.

Recommendation:

While this is secure in most circumstances whereby funds are never meant to remain at rest, the effects of this potentially unwanted accounting of accidental transfers should be evaluated.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

HRW-01: Assignment Optimization

Type	Severity	Location
Optimization	Informational	HardRewards.sol L59

Description:

The linked assignment of `block.number` to `lastReward[vault]` can be moved to the `if` block between L49 and L56 as, in any other case, it is redundant.

Recommendation:

L34 guarantees that the `blockReward` variable is greater-than zero and L46 would result in zero only if `lastReward[vault]` is already equal to `block.number`, so an assignment of `block.number` is only sensible if `lastReward[vault]` contains a different value which is the case only in the `if` clause of L49.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

HRW-02: Conditional Optimization

Type	Severity	Location
Optimization	Informational	HardRewards.sol L34, L40

Description:

The linked `if` clauses contain the same statements as the `else` clause of L56.

Recommendation:

As a result, it is advisable that the inverse conditions are checked on the `if` clause of L49 and the linked `if` clauses are omitted completely.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

HRW-03: Inefficient Greater-Than Comparison w/ Zero

Type	Severity	Location
Optimization	Informational	HardRewards.sol L49, L52, L77

Description:

The linked greater-than comparisons with zero compare variables that are restrained to the non-negative integer range, meaning that the comparator can be changed to an inequality one which is more gas efficient.

Recommendation:

We advise that the above paradigm is applied to the linked greater-than statements.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

HRW-04: Redundant Syntax

Type	Severity	Location
Syntax	Informational	HardRewards.sol L67

Description:

The linked statement conducts a `delete` operation on a parenthesis of a mapping lookup.

Recommendation:

The parenthesis can be safely omitted.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

NHP-01: Function Input Optimization

Type	Severity	Location
Language Specific	Informational	NotifyHelper.sol L14

Description:

When function calls are not utilized elsewhere internally in the codebase and contain array inputs, it is highly advisable to set them as `external` and instead store the input arrays in `calldata` rather than `memory` to greatly optimize the gas cost involved in invoking those functions.

Recommendation:

We advise the above pattern is applied on the linked function.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

NHP-02: Inefficient Greater-Than Comparison w/ Zero

Type	Severity	Location
Optimization	Informational	NotifyHelper.sol L17

Description:

The linked greater-than comparisons with zero compare variables that are restrained to the non-negative integer range, meaning that the comparator can be changed to an inequality one which is more gas efficient.

Recommendation:

We advise that the above paradigm is applied to the linked greater-than statements.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

NHP-03: Variable Re-use

Type	Severity	Location
Optimization	Informational	NotifyHelper.sol L22

Description:

The linked external call conducts a surplus type-casting which already exists under the variable `pool`.

Recommendation:

We advise that the already-declared variable `pool` is utilized instead.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

RWP-01: Library Consistency

Type	Severity	Location
Syntax	Informational	RewardPool.sol L57

Description:

The `Math` library contains a `max` and a `mint` function, the former using a loose greater-than comparison in contrast to the latter.

Recommendation:

We advise that a uniform comparison is utilized as both are equivalent.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

RWP-02: Inefficient Greater-Than Comparison w/ Zero

Type	Severity	Location
Optimization	Informational	RewardPool.sol L720, L726, L738

Description:

The linked greater-than comparisons with zero compare variables that are restrained to the non-negative integer range, meaning that the comparator can be changed to an inequality one which is more gas efficient.

Recommendation:

We advise that the above paradigm is applied to the linked greater-than statements.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

VLT-01: Unconventional Syntax

Type	Severity	Location
Syntax	Informational	Vault.sol L140

Description:

The linked representation of the maximum of `uint256` is unconventional.

Recommendation:

We advise that either `~uint256(0)` or `uint256(-1)` is utilized, the former of which we suggest. Additionally, it may be wise to store it in a contract-level `constant` declaration for ease-of-use.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

VLT-02: Inefficient Greater-Than Comparison w/ Zero

Type	Severity	Location
Optimization	Informational	Vault.sol L145, L172, L200, L201, L230

Description:

The linked greater-than comparisons with zero compare variables that are restrained to the non-negative integer range, meaning that the comparator can be changed to an inequality one which is more gas efficient.

Recommendation:

We advise that the above paradigm is applied to the linked greater-than statements.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

VLT-03: Conditional Consistency

Type	Severity	Location
Codebase Consistency	Informational	Vault.sol L46, L146

Description:

The `constructor` of the `Vault` contract ensures that the numerator is less-than-or-equal to the denominator, however the setter does not permit the equality case.

Recommendation:

We advise that one of the two conditionals is properly enforced on both `require` statements.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

VLT-04: Redundant SafeMath

Type	Severity	Location
Mathematical	Informational	Vault.sol L164, L213

Description:

The linked mathematical statements can be represented in their raw format rather than their `SafeMath` counterpart as the statements of L161 and L208 ensure their safety respectively.

Recommendation:

We advise that the SafeMath utilization is avoided here to optimize gas cost.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

VLT-05: Function Invocation Re-use

Type	Severity	Location
Optimization	Informational	Vault.sol L200, L202

Description:

The result of the `totalSupply()` function invocation is utilized twice.

Recommendation:

As its result won't change across invocations, it is more optimal to store the result of the invocation to an in-memory variable.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

CTL-01: Incorrect Error Message

Type	Severity	Location
Optimization	Informational	Controller.sol L60

Description:

The linked `require` statement checks whether the `msg.sender` is a hard worker or the governance, however the error message differs.

Recommendation:

We advise that the error message is synced with what the conditionals represent.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

CTL-02: Contract Bytecode Optimization

Type	Severity	Location
Optimization	Informational	Controller.sol L70-L91

Description:

The linked code block contains two sets of `add` and `remove` prefixed functions that occupy surplus bytecode size.

Recommendation:

We advise that each set is assimilated to a single `setter` function that sets the hard workers and grey lists status respectively.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

CTL-03: Inefficient Greater-Than Comparison w/ Zero

Type	Severity	Location
Optimization	Informational	Controller.sol L145

Description:

The linked greater-than comparisons with zero compare variables that are restrained to the non-negative integer range, meaning that the comparator can be changed to an inequality one which is more gas efficient.

Recommendation:

We advise that the above paradigm is applied to the linked greater-than statements.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

PNF-01: `require` Consistency

Type	Severity	Location
Codebase Consistency	Informational	ProfitNotifier.sol L32

Description:

The linked numerator and denominator `require` conditional should be synced with the result of VLT-03.

Recommendation:

N/A.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

PNF-02: Typecasting Optimization

Type	Severity	Location
Optimization	Informational	ProfitNotifier.sol L15

Description:

The linked `underlying` declaration is never utilized as an `address` type and is instead always casted to an `IERC20` interface.

Recommendation:

We advise that the `address` is directly stored as the interface's type, `IERC20`.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

RPN-01: Variable Mutability Specifiers

Type	Severity	Location
Optimization	Informational	RewardTokenProfitNotifier.sol L13-L14

Description:

The linked contract variables are assigned to only once during the contract's `constructor` and the assignment contains literal values.

Recommendation:

If the literal values remain, it is advisable to instead declare those two variables as `constant` optimizing gas cost.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

RPN-02: `require` Consistency

Type	Severity	Location
Codebase Consistency	Informational	RewardTokenProfitNotifier.sol L25

Description:

The linked numerator and denominator `require` conditional should be synced with the result of VLT-03.

Recommendation:

N/A.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

RPN-03: Inefficient Greater-Than Comparison w/ Zero

Type	Severity	Location
Optimization	Informational	RewardTokenProfitNotifier.sol L31

Description:

The linked greater-than comparisons with zero compare variables that are restrained to the non-negative integer range, meaning that the comparator can be changed to an inequality one which is more gas efficient.

Recommendation:

We advise that the above paradigm is applied to the linked greater-than statements.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SRS-01: Redundant SafeMath Operation

Type	Severity	Location
Optimization	Informational	SNXRewardStrategy.sol L162, L239

Description:

The linked mathematical statements can be represented in their raw format rather than their `SafeMath` counterpart as the statements of L161 and L236 ensure their safety respectively.

Recommendation:

We advise that the SafeMath utilization is avoided here to optimize gas cost.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SRS-02: Uniswap Conformity

Type	Severity	Location
Language Specific	Informational	SNXRewardStrategy.sol L170-L178

Description:

The Uniswap implementation of a route defines a set of point to point segments that are meant to direct the contract how to swap each token pair to the next one. The current function implementation allows for no routes to be defined as it is possible for the route array to be valid and be composed of a single element.

Recommendation:

We advise that a `require` check is imposed that ensures the uniswap route has a length greater-than-or-equal (`>=`) to `2` as the current function allows the inclusion of a single address array which should not be the case.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SRS-03: Variable Re-use

Type	Severity	Location
Optimization	Informational	SNXRewardStrategy.sol L236-L241

Description:

The linked code segment evaluates the external call `underlying.balanceOf(address(this))` twice. The comment beneath the `if` statement dictates that the result of this invocation may change between the `if` block invocation and the `sub` invocation that immediately follows it.

Recommendation:

We advise that the result of the `underlying.balanceOf(address(this))` call is actually stored to an in-memory variable as it is not meant to change between those two invocations since `balanaceOf` is a `view` function that cannot possibly alter `storage`. Thus, those two results are guaranteed to be the same.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SRS-04: Inefficient Greater-Than Comparison w/ Zero

Type	Severity	Location
Optimization	Informational	SNXRewardStrategy.sol L185, L213

Description:

The linked greater-than comparisons with zero compare variables that are restrained to the non-negative integer range, meaning that the comparator can be changed to an inequality one which is more gas efficient.

Recommendation:

We advise that the above paradigm is applied to the linked greater-than statements.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SRU-01: Visibility Specifier Missing

Type	Severity	Location
Optimization	Informational	SNXRewardUniLPStrategy.sol L58

Description:

The visibility specifier of the linked variable is missing.

Recommendation:

We advise that the visibility specifier for the linked variable is properly set.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SRU-02: Inefficient Greater-Than Comparison w/ Zero

Type	Severity	Location
Optimization	Informational	SNXRewardUniLPStrategy.sol L153, L216

Description:

The linked greater-than comparisons with zero compare variables that are restrained to the non-negative integer range, meaning that the comparator can be changed to an inequality one which is more gas efficient.

Recommendation:

We advise that the above paradigm is applied to the linked greater-than statements.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SRU-03: Redundant Typecasting

Type	Severity	Location
Optimization	Informational	SNXRewardUniLPStrategy.sol L154, L155, L170

Description:

The linked statements contain redundant castings of `address` variables to the `address` type.

Recommendation:

As such, they can be safely omitted.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SRU-04: Incorrect Comment

Type	Severity	Location
Documentation Conformity	Informational	SNXRewardUniLPStrategy.sol L279

Description:

The comment of the linked function states that "Note that although `onlyNotPausedInvesting` is not added here" which is invalid as it is actually used as seen on L283.

Recommendation:

We advise the linked comment be revised.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

CMI-01: Variable Mutability and Type

Type	Severity	Location
Optimization	Informational	CompoundInteractor.sol L20

Description:

The linked variable is only assigned to once at its contract-level declaration and it is redundantly casted to the `IERC20` type whilst it is only used after being again cast to the `WETH9` type.

Recommendation:

We advise that its mutability specifier is set to `constant` and that it is stored as a `WETH9` variable rather than an `IERC20` variable.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

CMI-02: Inefficient Greater-Than Comparison w/ Zero

Type	Severity	Location
Optimization	Informational	CompoundInteractor.sol L110, L119

Description:

The linked greater-than comparisons with zero compare variables that are restrained to the non-negative integer range, meaning that the comparator can be changed to an inequality one which is more gas efficient.

Recommendation:

We advise that the above paradigm is applied to the linked greater-than statements.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

WCS-01: Inefficient Greater-Than Comparison w/ Zero

Type	Severity	Location
Optimization	Informational	WETHCreamNoFoldStrategy.sol L170

Description:

The linked greater-than comparisons with zero compare variables that are restrained to the non-negative integer range, meaning that the comparator can be changed to an inequality one which is more gas efficient.

Recommendation:

We advise that the above paradigm is applied to the linked greater-than statements.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

PCV-01: Variable Mutability Specifier

Type	Severity	Location
Optimization	Informational	PriceConvertor.sol L11

Description:

The linked variable is only assigned to once at its contract-level declaration.

Recommendation:

We advise that its mutability is set to `constant`, optimizing the gas cost involved in utilizing it.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SSM-01: `revert` Statement

Type	Severity	Location
Syntactic	Informational	CRVStrategyStableMainnet.sol L11

Description:

The linked code block contains a `revert` statement which is ill-advised.

Recommendation:

While it makes complete sense to keep the code as is, for it to conform to the latest standards an alternative would be to utilize a local variable assigned to the maximum of `uint256` that is then assigned in the chained `if-else` statements and consequently used in a `require` statement that ensures it has changed.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SST-01: Visibility Specifier Missing

Type	Severity	Location
Syntactic	Informational	CRVStrategyStable.sol L38

Description:

The linked variable has no visibility specifier set.

Recommendation:

We advise that an explicit visibility specifier is set to aid in the legibility of the codebase.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SST-02: Variable Mutability Specifier

Type	Severity	Location
Optimization	Informational	CRVStrategyStable.sol L99

Description:

The linked variable is only assigned to once during the contract's `constructor` and is done so to a value literal rather than an input variable.

Recommendation:

We advise that the variable is set to a `constant` greatly optimizing the gas cost involved in utilizing it and moving its assignment to its declaration.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SST-03: Inefficient Greater-Than Comparison w/ Zero

Type	Severity	Location
Optimization	Informational	CRVStrategyStable.sol L124, L141, L190, L208, L222

Description:

The linked greater-than comparisons with zero compare variables that are restrained to the non-negative integer range, meaning that the comparator can be changed to an inequality one which is more gas efficient.

Recommendation:

We advise that the above paradigm is applied to the linked greater-than statements.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SST-04: Unconventional Syntax

Type	Severity	Location
Syntactic	Informational	CRVStrategyStable.sol L158, L206

Description:

The linked representation of the maximum of `uint256` is unconventional.

Recommendation:

We advise that either `~uint256(0)` or `uint256(-1)` is utilized, the former of which we suggest. Additionally, it may be wise to store it in a contract-level `constant` declaration for ease-of-use.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SST-05: Redundant SafeMath Operation

Type	Severity	Location
Optimization	Informational	CRVStrategyStable.sol L173

Description:

The linked mathematical statement can be represented in its raw format rather than its `SafeMath` counterpart as the statement of L171 ensures its safety.

Recommendation:

We advise that the SafeMath utilization is avoided here to optimize gas cost.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SST-06: Variable Re-use

Type	Severity	Location
Optimization	Informational	CRVStrategyStable.sol L260

Description:

The linked in-memory variable declaration should be omitted as `ycrvUnit` can be used instead with no extra gas cost.

Recommendation:

Included above.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SST-07: Variable Re-use

Type	Severity	Location
Optimization	Informational	CRVStrategyStable.sol L272, L289

Description:

The linked numeric literals should be omitted as `ycrvUnit` can be used instead with no extra gas cost.

Recommendation:

Included above.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SYC-01: Variable Mutability Specifier

Type	Severity	Location
Optimization	Informational	CRVStrategyYCRV.sol L42

Description:

The linked variable is only assigned to once during its contract-level declaration.

Recommendation:

We advise that the variable is set to a `constant` greatly optimizing the gas cost involved in utilizing it.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SYC-02: Variable Declaration Misuse

Type	Severity	Location
Optimization / Syntactical	Informational	CRVStrategyYCRV.sol L50

Description:

The linked variable is only assigned to once during its contract-level declaration, has no visibility specifier and conforms to an unusual syntax.

Recommendation:

We advise that the variable is set to a `constant` greatly optimizing the gas cost involved in utilizing it, its visibilit specifier is explicitly set and that it is represented either by `~uint256(0)` or `uint256(-1)`, the former of which we advise.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SYC-03: Variable Re-use

Type	Severity	Location
Optimization	Informational	CRVStrategyYCRV.sol L88

Description:

The linked array declaration utilises 3 `storage` declarations whilst they are readily available in memory.

Recommendation:

We advise that the in-memory variables are used instead optimizing the deployment cost of the contract.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SYC-04: Inefficient Greater-Than Comparison w/ Zero

Type	Severity	Location
Optimization	Informational	CRVStrategyYCRV.sol L140, L156, L169, L198, L204

Description:

The linked greater-than comparisons with zero compare variables that are restrained to the non-negative integer range, meaning that the comparator can be changed to an inequality one which is more gas efficient.

Recommendation:

We advise that the above paradigm is applied to the linked greater-than statements.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SYC-05: Literal over Memory

Type	Severity	Location
Optimization	Informational	CRVStrategyYCRV.sol L209

Description:

The linked variable utilization serves no purpose apart from explicitly representing its purpose.

Recommendation:

As memory declarations cost gas, we advise that a literal is utilized here instead that is properly documented in the form of comments rather than memory variable names.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SSW-01: Variable Visibility Specifier

Type	Severity	Location
Optimization	Informational	CRVStrategySwerve.sol L34

Description:

The linked variable contains no visibility specifier.

Recommendation:

We advise that an explicit visibility specifier is set for it.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SSW-02: Variable Mutability Specifier

Type	Severity	Location
Optimization	Informational	CRVStrategySwerve.sol L123

Description:

The linked variable is only assigned to once during the contract's `constructor` and is done so to a value literal rather than an input variable.

Recommendation:

We advise that the variable is set to a `constant` greatly optimizing the gas cost involved in utilizing it and moving its assignment to its declaration.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SSW-03: Inefficient Greater-Than Comparison w/ Zero

Type	Severity	Location
Optimization	Informational	CRVStrategySwerve.sol L148, L205, L223, L237

Description:

The linked greater-than comparisons with zero compare variables that are restrained to the non-negative integer range, meaning that the comparator can be changed to an inequality one which is more gas efficient.

Recommendation:

We advise that the above paradigm is applied to the linked greater-than statements.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SSW-04: Unconventional Syntax

Type	Severity	Location
Syntactic	Informational	CRVStrategySwerve.sol L220

Description:

The linked representation of the maximum of `uint256` is unconventional.

Recommendation:

We advise that either `~uint256(0)` or `uint256(-1)` is utilized, the former of which we suggest. Additionally, it may be wise to store it in a contract-level `constant` declaration for ease-of-use.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SWB-01: Variable Visibility Specifier

Type	Severity	Location
Optimization	Informational	CRVStrategyWRenBTC.sol L33

Description:

The linked variable contains no visibility specifier.

Recommendation:

We advise that an explicit visibility specifier is set for it.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SWB-02: Variable Mutability Specifier

Type	Severity	Location
Optimization	Informational	CRVStrategyWRenBTC.sol L119

Description:

The linked variable is only assigned to once during the contract's `constructor` and is done so to a value literal rather than an input variable.

Recommendation:

We advise that the variable is set to a `constant` greatly optimizing the gas cost involved in utilizing it and moving its assignment to its declaration.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SWB-03: Inefficient Greater-Than Comparison w/ Zero

Type	Severity	Location
Optimization	Informational	CRVStrategyWRenBTC.sol L144, L201, L219, L233

Description:

The linked greater-than comparisons with zero compare variables that are restrained to the non-negative integer range, meaning that the comparator can be changed to an inequality one which is more gas efficient.

Recommendation:

We advise that the above paradigm is applied to the linked greater-than statements.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

SWB-04: Unconventional Syntax

Type	Severity	Location
Syntactic	Informational	CRVStrategyWRenBTC.sol L216

Description:

The linked representation of the maximum of `uint256` is unconventional.

Recommendation:

We advise that either `~uint256(0)` or `uint256(-1)` is utilized, the former of which we suggest. Additionally, it may be wise to store it in a contract-level `constant` declaration for ease-of-use.

Alleviation:

The Harvest team has decided to not apply this exhibit due to it being an optimizational benefit rather than a security concern.

Appendix

Icons explanation

✓ : Issue resolved

❗: Issue not resolved / Acknowledged. The team will be fixing the issues in the own timeframe.

❗✓: Issue partially resolved. Not all instances of an issue was resolved.

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.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

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

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a `struct` assignment operation affecting an in-memory `struct` rather than an in-storage one.

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 and comment on how to make the codebase more legible and as a result easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a `constructor` assignment imposing different `require` statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as `constant` contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Dead Code

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