Page 1.

SolidityScan | Powered by CRED SHiELDS

SELF PUBLISHED AUDIT REPORT
PUBLISHED ON SOLIDITYSCAN

Security Assessment

26 Oct 2023

This security assessment report was prepared by SolidityScan.com, a cloud-based Smart Contract Scanner.

Self-published

This audit report was Self-published by the user. To learn more about our published reports click here.

Table of Contents.

Project Summary

This report has been prepared for using SolidityScan to scan and discover vulnerabilities and safe coding practices in their smart contract including the libraries used by the contract that are not officially recognized. The SolidityScan tool runs a comprehensive static analysis on the Solidity code and finds vulnerabilities ranging from minor gas optimizations to major vulnerabilities leading to the loss of funds. The coverage scope pays attention to all the informational and critical vulnerabilities with over (140+) modules. The scanning and auditing process covers the following areas:

Various common and uncommon attack vectors will be investigated to ensure that the smart contracts are secure from malicious actors. The scanner modules find and flag issues related to Gas optimizations that help in reducing the overall Gas cost It scans and evaluates the codebase against industry best practices and standards to ensure compliance It makes sure that the officially recognized libraries used in the code are secure and up to date

The SolidityScan Team recommends running regular audit scans to identify any vulnerabilities that are introduced after introduces new features or refactors the code.

Audit Summary

Contract Name

Libertatis

Contract Type

Smart Contract

Contract Address

0×32D36EC677F26cc9640a3113705baCB421090438

Contract Platform

etherscan

Contract Chain

mainnet

Contract URL

https://etherscan.io/address/0×32D36EC677F26cc9640a3113705baCB421090438

Language

Solidity

Website

https://libertatis.finance

Date Published

26 Oct 2023

Organization

Libertatis

Publishers/Owners Name

Libertatis Team

Audit Methodology

Static Scanning

Page 5.

Findings Summary

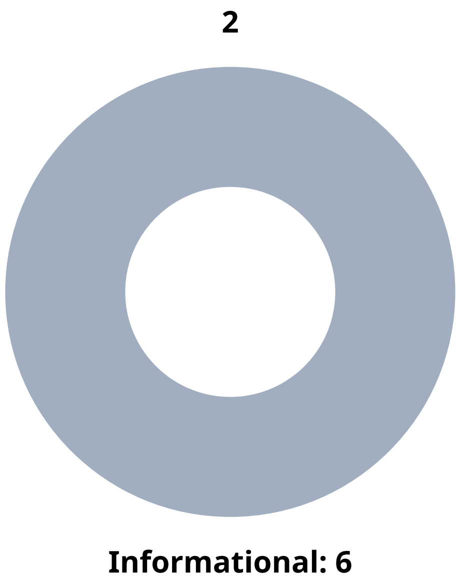
Libertatis

0×32D36EC677F26cc9640a3113705baCB421090438

Lines of Code **297**

98.00

Security Score



Critical	0	Low	0
High	0	Informational	6
Medium	0	Gas	0

Page 6.

ACTION TAKEN

Fixed ✓ 0	False Positive 18
Won't Fix 6	Pending Fix ⚠ 0

Bug ID	Severity	Bug Type	Status
SSB_63443_13	● Informational	BLOCK VALUES AS A PROXY FOR TIME	*False Positive*
SSB_63443_14	● Informational	BLOCK VALUES AS A PROXY FOR TIME	*False Positive*
SSB_63443_15	● Informational	BLOCK VALUES AS A PROXY FOR TIME	*False Positive*
SSB_63443_16	● Informational	BLOCK VALUES AS A PROXY FOR TIME	*False Positive*
SSB_63443_17	● Informational	BLOCK VALUES AS A PROXY FOR TIME	*False Positive*
SSB_63443_18	● Informational	BLOCK VALUES AS A PROXY FOR TIME	*False Positive*

Page 7.

SSB_63443_19	Informational	BLOCK VALUES AS A PROXY FOR TIME	*False Positive*
SSB_63443_10	Gas	CHEAPER INEQUALITIES IN REQUIRE()	*False Positive*
SSB_63443_7	Low	MISSING EVENTS	*False Positive*
SSB_63443_8	Low	MISSING EVENTS	*False Positive*
SSB_63443_9	Low	MISSING EVENTS	*False Positive*
SSB_63443_3	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	*Won't Fix*
SSB_63443_4	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	*Won't Fix*
SSB_63443_5	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	*Won't Fix*
SSB_63443_6	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	*Won't Fix*
SSB_63443_1	Informational	NAME MAPPING PARAMETERS	*Won't Fix*
SSB_63443_2	Informational	NAME MAPPING PARAMETERS	*Won't Fix*
SSB_63443_11	Gas	PUBLIC CONSTANTS CAN BE PRIVATE	*False Positive*

Page 8.

SSB_63443_12	● Gas	PUBLIC CONSTANTS CAN BE PRIVATE	*False Positive*
SSB_63443_20	● Gas	STORAGE VARIABLE CACHING IN MEMORY	*False Positive*
SSB_63443_21	● Gas	STORAGE VARIABLE CACHING IN MEMORY	*False Positive*
SSB_63443_22	● Gas	STORAGE VARIABLE CACHING IN MEMORY	*False Positive*
SSB_63443_23	● Gas	STORAGE VARIABLE CACHING IN MEMORY	*False Positive*
SSB_63443_23	● Gas	STORAGE VARIABLE CACHING IN MEMORY	*False Positive*
SSB_63443_24	● Gas	STORAGE VARIABLE CACHING IN MEMORY	*False Positive*
SSB_63443_24	● Gas	STORAGE VARIABLE CACHING IN MEMORY	*False Positive*

Vulnerability Details

Bug ID

SSB_63443_13

Severity	Confidence
● **Informational**	**Firm**
Line nos	Action Taken
85-85	*False Positive*

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

contracts/Libertatis.sol

Issue Description

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

For `block.number`, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.

Due to variable block times, `block.number` should not be relied on for precise calculations of time.

Issue Remediation

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Bug ID

SSB_63443_14

Severity	Confidence
● **Informational**	**Firm**

Line nos	Action Taken
171-171	*False Positive*

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

contracts/Libertatis.sol

Issue Description

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

For `block.number`, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.

Due to variable block times, `block.number` should not be relied on for precise calculations of time.

Issue Remediation

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Page 11.

Bug ID

SSB_63443_15

Severity	Confidence
● **Informational**	**Firm**
Line nos	Action Taken
204-204	*False Positive*

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

contracts/Libertatis.sol

Issue Description

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

For `block.number`, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.

Due to variable block times, `block.number` should not be relied on for precise calculations of time.

Issue Remediation

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Bug ID

SSB_63443_16

Severity

● **Informational**

Confidence

Firm

Line nos

242-242

Action Taken

False Positive

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

contracts/Libertatis.sol

Issue Description

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

For `block.number`, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.

Due to variable block times, `block.number` should not be relied on for precise calculations of time.

Issue Remediation

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Page 13.

Bug ID

SSB_63443_17

Severity	Confidence
● **Informational**	**Firm**

Line nos	Action Taken
251-251	*False Positive*

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

contracts/Libertatis.sol

Issue Description

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

For `block.number`, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.

Due to variable block times, `block.number` should not be relied on for precise calculations of time.

Issue Remediation

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Page 14.

Bug ID

SSB_63443_18

Severity

- **Informational**

Confidence

Firm

Line nos

252-252

Action Taken

False Positive

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

contracts/Libertatis.sol

Issue Description

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

For `block.number`, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.

Due to variable block times, `block.number` should not be relied on for precise calculations of time.

Issue Remediation

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Page 15.

Bug ID

SSB_63443_19

Severity

● **Informational**

Confidence

Firm

Line nos

271-271

Action Taken

False Positive

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

contracts/Libertatis.sol

Issue Description

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

For `block.number`, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.

Due to variable block times, `block.number` should not be relied on for precise calculations of time.

Issue Remediation

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Page 16.

Bug ID

SSB_63443_10

Severity	Confidence
• **Gas**	**Firm**

Line nos	Action Taken
201-201	*False Positive*

Bug Type

CHEAPER INEQUALITIES IN REQUIRE()

File Location

contracts/Libertatis.sol

Issue Description

The contract was found to be performing comparisons using inequalities inside the `require` statement. When inside the `require` statements, non-strict inequalities `(>=, <=)` are usually costlier than strict equalities `(>, <)`.

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the non-strict inequalities with the strict ones to save `~3` gas as long as the logic of the code is not affected.

Page 17.

Bug ID

SSB_63443_7

Severity	Confidence
● **Low**	**Firm**

Line nos	Action Taken
276-278	*False Positive*

Bug Type

MISSING EVENTS

File Location

contracts/Libertatis.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract Libertatis was found to be missing these events on the function _burn which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSB_63443_8

Severity	Confidence
● **Low**	**Firm**
Line nos	Action Taken
280-282	*False Positive*

Bug Type

MISSING EVENTS

File Location

contracts/Libertatis.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction’s log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract Libertatis was found to be missing these events on the function _afterTokenTransfer which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSB_63443_9

Severity	Confidence
● **Low**	**Firm**

Line nos	Action Taken
284-286	*False Positive*

Bug Type

MISSING EVENTS

File Location

contracts/Libertatis.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction’s log—a special data structure in the blockchain.

These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.

The contract Libertatis was found to be missing these events on the function _mint which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSB_63443_3

Severity	Confidence
● **Informational**	**Tentative**
Line nos	Action Taken
48-48	*Won't Fix*

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

contracts/Libertatis.sol

Issue Description

Solidity style guide suggests using underscores as the prefix for non-external functions and state variables (private or internal) but the contract was not found to be following the same.

Issue Remediation

It is recommended to use an underscore for internal and private variables and functions to be in accordance with the Solidity style guide which will also make the code much easier to read.

Comments

According to the style guide of the team we won't fix this informative issues

Bug ID

SSB_63443_4

Severity	Confidence
● **Informational**	**Tentative**

Line nos	Action Taken
49-49	*Won't Fix*

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

contracts/Libertatis.sol

Issue Description

Solidity style guide suggests using underscores as the prefix for non-external functions and state variables (private or internal) but the contract was not found to be following the same.

Issue Remediation

It is recommended to use an underscore for internal and private variables and functions to be in accordance with the Solidity style guide which will also make the code much easier to read.

Comments

According to the style guide of the team we won't fix this informative issues

Bug ID

SSB_63443_5

Severity	Confidence
• **Informational**	**Tentative**

Line nos	Action Taken
226-245	*Won't Fix*

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

contracts/Libertatis.sol

Issue Description

Solidity style guide suggests using underscores as the prefix for non-external functions and state variables (private or internal) but the contract was not found to be following the same.

Issue Remediation

It is recommended to use an underscore for internal and private variables and functions to be in accordance with the Solidity style guide which will also make the code much easier to read.

Comments

According to the style guide of the team we won't fix this informative issues

Page 23.

Bug ID

SSB_63443_6

Severity	Confidence
● **Informational**	**Tentative**

Line nos	Action Taken
262-265	*Won't Fix*

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

contracts/Libertatis.sol

Issue Description

Solidity style guide suggests using underscores as the prefix for non-external functions and state variables (private or internal) but the contract was not found to be following the same.

Issue Remediation

It is recommended to use an underscore for internal and private variables and functions to be in accordance with the Solidity style guide which will also make the code much easier to read.

Comments

According to the style guide of the team we won't fix this informative issues

Page 24.

Bug ID

SSB_63443_1

Severity

● **Informational**

Confidence

Tentative

Line nos

66-66

Action Taken

Won't Fix

Bug Type

NAME MAPPING PARAMETERS

File Location

contracts/Libertatis.sol

Issue Description

After Solidity 0.8.18, a feature was introduced to name mapping parameters. This helps in defining a purpose for each mapping and makes the code more descriptive.

Issue Remediation

It is recommended to name the mapping parameters if Solidity 0.8.18 and above is used.

Comments

According to the style guide of the team we won't fix this informative issues

Page 25.

Bug ID

SSB_63443_2

Severity

● **Informational**

Confidence

Tentative

Line nos

67-67

Action Taken

Won't Fix

Bug Type

NAME MAPPING PARAMETERS

File Location

contracts/Libertatis.sol

Issue Description

After Solidity 0.8.18, a feature was introduced to name mapping parameters. This helps in defining a purpose for each mapping and makes the code more descriptive.

Issue Remediation

It is recommended to name the mapping parameters if Solidity 0.8.18 and above is used.

Comments

According to the style guide of the team we won't fix this informative issues

Page 26.

Bug ID

SSB_63443_11

Severity	Confidence
• **Gas**	**Certain**
Line nos	Action Taken
58-58	*False Positive*

Bug Type

PUBLIC CONSTANTS CAN BE PRIVATE

File Location

contracts/Libertatis.sol

Issue Description

Public constant variables cost more gas because the EVM automatically creates getter functions for them and adds entries to the method ID table. The values can be read from the source code instead.
The following variable is affected: LIBERTATIS_AI_ROLE

Issue Remediation

If reading the values for the constants are not necessary, consider changing the `public` visibility to `private`.

Page 27.

Bug ID

SSB_63443_12

Severity

Gas

Confidence

Certain

Line nos

59-59

Action Taken

False Positive

Bug Type

PUBLIC CONSTANTS CAN BE PRIVATE

File Location

contracts/Libertatis.sol

Issue Description

Public constant variables cost more gas because the EVM automatically creates getter functions for them and adds entries to the method ID table. The values can be read from the source code instead.
The following variable is affected: LIBERTATIS_MODERATOR_ROLE

Issue Remediation

If reading the values for the constants are not necessary, consider changing the `public` visibility to `private`.

Page 28.

Bug ID

SSB_63443_20

Severity

Gas

Confidence

Tentative

Line nos

58-58

Action Taken

False Positive

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

contracts/Libertatis.sol

Issue Description

The contract `Libertatis` is using the state variable `LIBERTATIS_AI_ROLE` multiple times in the function `addAiAccount`.
`SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD` / `MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Page 29.

Bug ID

SSB_63443_21

Severity	Confidence
● **Gas**	**Tentative**

Line nos	Action Taken
59-59	*False Positive*

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

contracts/Libertatis.sol

Issue Description

The contract `Libertatis` is using the state variable `LIBERTATIS_MODERATOR_ROLE` multiple times in the function `addModeratorAccount`.
`SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD` / `MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Page 30.

Bug ID

SSB_63443_22

Severity

● **Gas**

Confidence

Tentative

Line nos

43-43

Action Taken

False Positive

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

contracts/Libertatis.sol

Issue Description

The contract `Libertatis` is using the state variable `currentPhase` multiple times in the function `moveToNextPhase`.
`SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD`/`MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Page 31.

Bug ID

SSB_63443_23

Severity	Confidence
• **Gas**	**Tentative**

Line nos	Action Taken
45-45	*False Positive*

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

contracts/Libertatis.sol

Issue Description

The contract `Libertatis` is using the state variable `phaseSupply` multiple times in the function `moveToNextPhase`.
`SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD` / `MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Page 32.

Bug ID

SSB_63443_23

Severity	Confidence
• **Gas**	**Tentative**

Line nos	Action Taken
45-45	*False Positive*

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

contracts/Libertatis.sol

Issue Description

The contract `Libertatis` is using the state variable `phaseSupply` multiple times in the function `buyLibertatisWithReferral`.
`SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD` / `MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Bug ID

SSB_63443_24

Severity	Confidence
● **Gas**	**Tentative**

Line nos	Action Taken
66-66	*False Positive*

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

contracts/Libertatis.sol

Issue Description

The contract `Libertatis` is using the state variable `stakes` multiple times in the function `registerNewStake`.
`SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD` / `MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Bug ID

SSB_63443_24

Severity

● **Gas**

Confidence

Tentative

Line nos

66-66

Action Taken

False Positive

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

contracts/Libertatis.sol

Issue Description

The contract `Libertatis` is using the state variable `stakes` multiple times in the function `claimStakes`.
`SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD` / `MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Scan History

● Critical ● High ● Medium ● Low ● Informational ● Gas

No	Date	Security Score	Scan Overview
1.	2023-10-26	**98.00**	● 0 ● 0 ● 0 ● 0 ● 6 ● 0

Disclaimer

The Reports neither endorse nor condemn any specific project or team, nor do they guarantee the security of any specific project. The contents of this report do not, and should not be interpreted as having any bearing on, the economics of tokens, token sales, or any other goods, services, or assets.

The security audit is not meant to replace functional testing done before a software release.

There is no warranty that all possible security issues of a particular smart contract(s) will be found by the tool, i.e., It is not guaranteed that there will not be any further findings based solely on the results of this evaluation.

Emerging technologies such as Smart Contracts and Solidity carry a high level of technical risk and uncertainty. There is no warranty or representation made by this report to any Third Party in regards to the quality of code, the business model or the proprietors of any such business model, or the legal compliance of any business.

In no way should a third party use these reports to make any decisions about buying or selling a token, product, service, or any other asset. It should be noted that this report is not investment advice, is not intended to be relied on as investment advice, and has no endorsement of this project or team. It does not serve as a guarantee as to the project's absolute security.

The assessment provided by SolidityScan 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. SolidityScan owes no duty to any third party by virtue of publishing these Reports.

As one audit-based assessment cannot be considered comprehensive, we always recommend proceeding with several independent manual audits including manual audit and a public bug bounty program to ensure the security of the smart contracts.