$ASTROCOWINU

$ACI 5% HOLDING REWARDS

$ASTROCOWINU ($ACI)
Hyper Deflationary Token with holders reward and automatic buyback
5% HOLDING REWARDS - TAX: 14% BUY/SELL
Holders reward 5% Buyback 6% Marketing 3%

https://www.grangefinance.app/ASTRO-COW-INU
https://t.me/grangefinance

GRANGE FINANCE

Solidity Scan

Powered by CRED SHIELDS

```
Website: https://www.grangefinance.app
Telegram: https://t.me/grangefinance
```

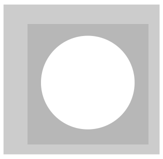
SolidityScan | Powered by CRED SHIELDS

Security Assessment

14 Dec 2022

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

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 (100+) 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	BuyBackToken
Contract Type	Smart Contract
Contract Address	0xe48DA8CF4Af5EbE6946F36CF1B7B19D41aCa2c26
Contract Platform	bscscan
	mainnet
	https://bscscan.com/address/0xe48DA8CF4Af5EbE6946F36CF1B7B19D

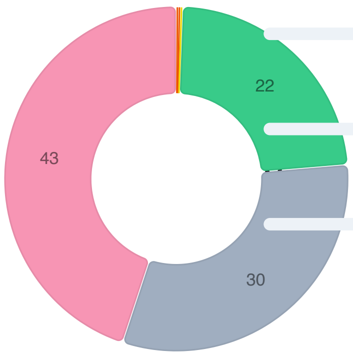
Contract Chain	Solidity
Contract URL	Static Scanning
Language	
Audit Methodology	

Findings Summary

BuyBackToken
0xe48DA8CF4Af5EbE6946F36CF1B7B19D41aCa2c26

Lines of Code
993

4.41
Score



Critical	0	Low	22
	0	Informational	30
	0	Gas	43

ACTION TAKEN

Fixed	False Positive	Won't Fix	Pending Fix
0	20	0	95

Bug ID	Severity	Bug Type	Status
SSB_1700_5	● Medium	ACCOUNT EXISTENCE CHECK FOR LOW LEVEL CALLS	False Positive
SSB_1700_95	● Informational	HARD-CODED ADDRESS DETECTED	False Positive
SSB_1700_96	● Informational	HARD-CODED ADDRESS DETECTED	False Positive
SSB_1700_97	● Informational	HARD-CODED ADDRESS DETECTED	False Positive
SSB_1700_39	● Informational	BLOCK VALUES AS A PROXY FOR TIME	Pending Fix
SSB_1700_40	● Informational	BLOCK VALUES AS A PROXY FOR TIME	Pending Fix
SSB_1700_41	● Informational	BLOCK VALUES AS A PROXY FOR TIME	Pending Fix
SSB_1700_42	● Informational	BLOCK VALUES AS A PROXY FOR TIME	Pending Fix
SSB_1700_43	● Informational	BLOCK VALUES AS A PROXY FOR TIME	Pending Fix
SSB_1700_42	● Informational	BLOCK VALUES AS A PROXY FOR TIME	Pending Fix
SSB_1700_8	● Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSB_1700_9	● Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix

SSB_1700_10	● Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSB_1700_11	● Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSB_1700_12	● Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSB_1700_13	● Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSB_1700_30	● Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_1700_31	● Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_1700_32	● Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_1700_33	● Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_1700_34	● Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_1700_35	● Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_1700_36	● Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_1700_37	● Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_1700_38	● Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_1700_14	● Gas	CUSTOM ERRORS TO SAVE GAS	False Positive
SSB_1700_46	● High	APPROVE FRONT-RUNNING ATTACK	False Positive

SSB_1700_47	● High	APPROVE FRONT-RUNNING ATTACK	False Positive
SSB_1700_48	● High	APPROVE FRONT-RUNNING ATTACK	False Positive
SSB_1700_49	● High	APPROVE FRONT-RUNNING ATTACK	False Positive
SSB_1700_2	● Gas	EXTRA GAS USAGE IN LOOPS	False Positive
SSB_1700_3	● Gas	EXTRA GAS USAGE IN LOOPS	False Positive
SSB_1700_29	● Low	USE OF FLOATING PRAGMA	False Positive
SSB_1700_50	● Gas	FUNCTION SHOULD RETURN STRUCT	False Positive
SSB_1700_4	● High	UNCHECKED ARRAY LENGTH	False Positive
SSB_1700_6	● Gas	GAS OPTIMIZATION IN INCREMENTS	False Positive
SSB_1700_7	● Gas	GAS OPTIMIZATION IN INCREMENTS	False Positive
SSB_1700_80	● Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_1700_81	● Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_1700_82	● Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_1700_83	● Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_1700_84	● Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix

SSB_1700_85	● Gas	LONG REQUIRE/REVERT STRINGS	ⓘ Pending Fix
SSB_1700_86	● Gas	LONG REQUIRE/REVERT STRINGS	ⓘ Pending Fix
SSB_1700_87	● Gas	LONG REQUIRE/REVERT STRINGS	ⓘ Pending Fix
SSB_1700_88	● Gas	LONG REQUIRE/REVERT STRINGS	ⓘ Pending Fix
SSB_1700_89	● Gas	LONG REQUIRE/REVERT STRINGS	ⓘ Pending Fix
SSB_1700_90	● Gas	LONG REQUIRE/REVERT STRINGS	ⓘ Pending Fix
SSB_1700_91	● Gas	LONG REQUIRE/REVERT STRINGS	ⓘ Pending Fix
SSB_1700_92	● Gas	LONG REQUIRE/REVERT STRINGS	ⓘ Pending Fix
SSB_1700_93	● Gas	LONG REQUIRE/REVERT STRINGS	ⓘ Pending Fix
SSB_1700_94	● Gas	LONG REQUIRE/REVERT STRINGS	ⓘ Pending Fix
SSB_1700_58	● Low	MISSING EVENTS	ⓘ Pending Fix
SSB_1700_59	● Low	MISSING EVENTS	ⓘ Pending Fix
SSB_1700_60	● Low	MISSING EVENTS	ⓘ Pending Fix
SSB_1700_61	● Low	MISSING EVENTS	ⓘ Pending Fix
SSB_1700_62	● Low	MISSING EVENTS	ⓘ Pending Fix

SSB_1700_63	● Low	MISSING EVENTS	ⓘ Pending Fix
SSB_1700_64	● Low	MISSING EVENTS	ⓘ Pending Fix
SSB_1700_65	● Low	MISSING EVENTS	ⓘ Pending Fix
SSB_1700_66	● Low	MISSING EVENTS	ⓘ Pending Fix
SSB_1700_67	● Low	MISSING EVENTS	ⓘ Pending Fix
SSB_1700_68	● Low	MISSING EVENTS	ⓘ Pending Fix
SSB_1700_69	● Low	MISSING EVENTS	ⓘ Pending Fix
SSB_1700_70	● Low	MISSING EVENTS	ⓘ Pending Fix
SSB_1700_71	● Low	MISSING EVENTS	ⓘ Pending Fix
SSB_1700_72	● Low	MISSING EVENTS	ⓘ Pending Fix
SSB_1700_73	● Low	MISSING EVENTS	ⓘ Pending Fix
SSB_1700_74	● Low	MISSING EVENTS	ⓘ Pending Fix
SSB_1700_75	● Low	MISSING EVENTS	ⓘ Pending Fix
SSB_1700_76	● Low	MISSING EVENTS	ⓘ Pending Fix
SSB_1700_77	● Low	MISSING EVENTS	ⓘ Pending Fix

SSB_1700_78	● Low	MISSING EVENTS	ⓘ Pending Fix
SSB_1700_79	● Low	MISSING EVENTS	ⓘ Pending Fix
SSB_1700_51	● Informational	MISSING INDEXED KEYWORDS IN EVENTS	ⓘ Pending Fix
SSB_1700_52	● Informational	MISSING INDEXED KEYWORDS IN EVENTS	ⓘ Pending Fix
SSB_1700_53	● Informational	MISSING INDEXED KEYWORDS IN EVENTS	ⓘ Pending Fix
SSB_1700_54	● Informational	MISSING INDEXED KEYWORDS IN EVENTS	ⓘ Pending Fix
SSB_1700_55	● Informational	MISSING INDEXED KEYWORDS IN EVENTS	ⓘ Pending Fix
SSB_1700_56	● Informational	MISSING INDEXED KEYWORDS IN EVENTS	ⓘ Pending Fix
SSB_1700_57	● Informational	MISSING INDEXED KEYWORDS IN EVENTS	ⓘ Pending Fix
SSB_1700_28	● Low	OUTDATED COMPILER VERSION	✓ False Positive
SSB_1700_98	● Informational	PRESENCE OF OVERPOWERED ROLE	ⓘ Pending Fix
SSB_1700_99	● Informational	PRESENCE OF OVERPOWERED ROLE	ⓘ Pending Fix
SSB_1700_100	● Informational	PRESENCE OF OVERPOWERED ROLE	ⓘ Pending Fix
SSB_1700_101	● Informational	PRESENCE OF OVERPOWERED ROLE	ⓘ Pending Fix
SSB_1700_102	● Informational	PRESENCE OF OVERPOWERED ROLE	ⓘ Pending Fix

SSB_1700_103	● Informational	PRESENCE OF OVERPOWERED ROLE	ⓘ Pending Fix
SSB_1700_104	● Informational	PRESENCE OF OVERPOWERED ROLE	ⓘ Pending Fix
SSB_1700_105	● Informational	PRESENCE OF OVERPOWERED ROLE	ⓘ Pending Fix
SSB_1700_106	● Informational	PRESENCE OF OVERPOWERED ROLE	ⓘ Pending Fix
SSB_1700_107	● Informational	PRESENCE OF OVERPOWERED ROLE	ⓘ Pending Fix
SSB_1700_108	● Informational	PRESENCE OF OVERPOWERED ROLE	ⓘ Pending Fix
SSB_1700_109	● Informational	PRESENCE OF OVERPOWERED ROLE	ⓘ Pending Fix
SSB_1700_110	● Informational	PRESENCE OF OVERPOWERED ROLE	ⓘ Pending Fix
SSB_1700_111	● Informational	PRESENCE OF OVERPOWERED ROLE	ⓘ Pending Fix
SSB_1700_112	● Informational	PRESENCE OF OVERPOWERED ROLE	ⓘ Pending Fix
SSB_1700_113	● Informational	PRESENCE OF OVERPOWERED ROLE	ⓘ Pending Fix
SSB_1700_114	● Informational	PRESENCE OF OVERPOWERED ROLE	ⓘ Pending Fix
SSB_1700_1	● Gas	USE OF SAFEMATH LIBRARY	False Positive
SSB_1700_15	● Gas	FUNCTION SHOULD BE EXTERNAL	ⓘ Pending Fix
SSB_1700_16	● Gas	FUNCTION SHOULD BE EXTERNAL	ⓘ Pending Fix

SSB_1700_17	● Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSB_1700_18	● Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSB_1700_19	● Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSB_1700_20	● Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSB_1700_21	● Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSB_1700_22	● Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSB_1700_23	● Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSB_1700_24	● Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSB_1700_25	● Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSB_1700_26	● Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSB_1700_27	● Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSB_1700_44	● Informational	IN-LINE ASSEMBLY DETECTED	False Positive
SSB_1700_45	● Informational	IN-LINE ASSEMBLY DETECTED	False Positive

Vulnerability Details.

Bug ID

SSB_1700_5

Severity	Confidence	Line nos	Action Taken
● Medium	Tentative	107-113	False Positive

Bug Type

ACCOUNT EXISTENCE CHECK FOR LOW LEVEL CALLS

File Location

contract.sol

Issue Description

The low-level calls such as the `delegatecall`, `call`, or `callcode`, do not validate prior to the call if the destination account exists or not. They will always return true even if the account is non-existent, therefore, giving invalid output.

Issue Remediation

It is recommended to have an account existence check before making these low-level calls to confirm the presence of an external account with some valid code. Eg: using `extcodesize`.

Bug ID

SSB_1700_95

Severity	Confidence	Line nos	Action Taken
Informational	Tentative	426-426	False Positive

Bug Type

HARD-CODED ADDRESS DETECTED

File Location

contract.sol

Issue Description

The contract contains an unknown hard-coded address. This address might be used for some malicious activity. Please check the hard-coded address and its usage.
These hard-coded addresses may be used everywhere throughout the code to define states and interact with the functions and external calls.
Therefore, it is extremely crucial to ensure the correctness of these token contracts as they define various important aspects of the protocol operation.
A misconfigured address mapping could lead to the potential loss of user funds or compromise of the contract owner depending on the function logic.
The following hard-coded addresses were found -
`0x000000000000000000000000000000000000dEaD`

Issue Remediation

It is required to check the address. Also, it is required to check the code of the called contract for vulnerabilities.
Ensure that the contract validates if there's an address or a code change or test cases to validate if the address is correct.

Bug ID

SSB_1700_96

Severity	Confidence	Line nos	Action Taken
Informational	Tentative	**509-509**	False Positive

Bug Type

HARD-CODED ADDRESS DETECTED

File Location

contract.sol

Issue Description

The contract contains an unknown hard-coded address. This address might be used for some malicious activity. Please check the hard-coded address and its usage.
These hard-coded addresses may be used everywhere throughout the code to define states and interact with the functions and external calls.

Therefore, it is extremely crucial to ensure the correctness of these token contracts as they define various important aspects of the protocol operation.
A misconfigured address mapping could lead to the potential loss of user funds or compromise of the contract owner depending on the function logic.
The following hard-coded addresses were found -
`0xEe89c2E6462141356c580f97BE3D5a35Abc3b27e`

Issue Remediation

It is required to check the address. Also, it is required to check the code of the called contract for vulnerabilities.
Ensure that the contract validates if there's an address or a code change or test cases to validate if the address is correct.

Bug ID

SSB_1700_97

Severity	Confidence	Line nos	Action Taken
Informational	Tentative	**528-528**	False Positive

Bug Type

HARD-CODED ADDRESS DETECTED

File Location

contract.sol

Issue Description

The contract contains an unknown hard-coded address. This address might be used for some malicious activity. Please check the hard-coded address and its usage.
These hard-coded addresses may be used everywhere throughout the code to define states and interact with the functions and external calls.
Therefore, it is extremely crucial to ensure the correctness of these token contracts as they define various important aspects of the protocol operation.
A misconfigured address mapping could lead to the potential loss of user funds or compromise of the contract owner depending on the function logic.
The following hard-coded addresses were found -
`0x10ED43C718714eb63d5aA57B78B54704E256024E`

Issue Remediation

It is required to check the address. Also, it is required to check the code of the called contract for vulnerabilities.

Ensure that the contract validates if there's an address or a code change or test cases to validate if the address is correct.

Bug ID

SSB_1700_39

Severity	Confidence	Line nos	Action Taken
Informational	Firm	191-191	Pending Fix

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

contract.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

Smart contracts should be written with the idea that block values are not precise, and their use can have unexpected results. Alternatively, oracles can be used.

Bug ID

SSB_1700_40

Severity	Confidence	Line nos	Action Taken

Informational	Firm	197-197	Pending Fix

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

contract.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

Smart contracts should be written with the idea that block values are not precise, and their use can have unexpected results. Alternatively, oracles can be used.

Bug ID

SSB_1700_41

Severity	Confidence	Line nos	Action Taken
Informational	Firm	203-203	Pending Fix

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

contract.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

Smart contracts should be written with the idea that block values are not precise, and their use can have unexpected results. Alternatively, oracles can be used.

Bug ID

SSB_1700_42

Severity	Confidence	Line nos	Action Taken
Informational	Firm	740-740	Pending Fix

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

contract.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

Smart contracts should be written with the idea that block values are not precise, and their use can have unexpected results. Alternatively, oracles can be used.

Bug ID

SSB_1700_43

Severity	Confidence	Line nos	Action Taken
Informational	Firm	757-757	Pending Fix

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

contract.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

Smart contracts should be written with the idea that block values are not precise, and their use can have unexpected results. Alternatively, oracles can be used.

Bug ID

SSB_1700_42

Severity	Confidence	Line nos	Action Taken
Informational	Firm	774-774	Pending Fix

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

contract.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

Smart contracts should be written with the idea that block values are not precise, and their use can have unexpected results. Alternatively, oracles can be used.

Bug ID

SSB_1700_8

Severity	Confidence	Line nos	Action Taken
● Gas	Firm	141-141	Pending Fix

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

contract.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_1700_9

Severity	Confidence	Line nos	Action Taken
• Gas	Firm	638-638	Pending Fix

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

contract.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_1700_10

Severity	Confidence	Line nos	Action Taken
• Gas	Firm	687-687	ⓘ Pending Fix

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

contract.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_1700_11

Severity	Confidence	Line nos	Action Taken
• Gas	Firm	689-689	ⓘ Pending Fix

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

contract.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_1700_12

Severity	Confidence	Line nos	Action Taken
● Gas	Firm	721-721	Pending Fix

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

contract.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_1700_13

Severity	Confidence	Line nos	Action Taken
● Gas	Firm	875-875	Pending Fix

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

contract.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_1700_30

Severity	Confidence	Line nos	Action Taken
● Gas	Firm	44-44	Pending Fix

Bug Type

CHEAPER INEQUALITIES IN REQUIRE()

File Location

contract.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.

Bug ID

SSB_1700_31

Severity	Confidence	Line nos	Action Taken
● Gas	Firm	54-54	Pending Fix

Bug Type

CHEAPER INEQUALITIES IN REQUIRE()

File Location

contract.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.

Bug ID

SSB_1700_32

Severity	Confidence	Line nos	Action Taken

● Gas	Firm	108-108	ⓘ Pending Fix

Bug Type

CHEAPER INEQUALITIES IN REQUIRE()

File Location

contract.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.

Bug ID

SSB_1700_33

Severity	Confidence	Line nos	Action Taken
● Gas	Firm	129-129	ⓘ Pending Fix

Bug Type

CHEAPER INEQUALITIES IN REQUIRE()

File Location

contract.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.

Bug ID
SSB_1700_34

Severity	Confidence	Line nos	Action Taken
Gas	Firm	500-500	Pending Fix

Bug Type
CHEAPER INEQUALITIES IN REQUIRE()

File Location
contract.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.

Bug ID
SSB_1700_35

Severity	Confidence	Line nos	Action Taken
Gas	Firm	619-619	Pending Fix

Bug Type
CHEAPER INEQUALITIES IN REQUIRE()

File Location

contract.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.

Bug ID

SSB_1700_36

Severity	Confidence	Line nos	Action Taken
● Gas	Firm	630-630	Pending Fix

Bug Type

CHEAPER INEQUALITIES IN REQUIRE()

File Location

contract.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.

Bug ID

SSB_1700_37

Severity	Confidence	Line nos	Action Taken
• Gas	Firm	675-675	Pending Fix

Bug Type

CHEAPER INEQUALITIES IN REQUIRE()

File Location

contract.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.

Bug ID

SSB_1700_38

Severity	Confidence	Line nos	Action Taken
• Gas	Firm	946-946	Pending Fix

Bug Type

CHEAPER INEQUALITIES IN REQUIRE()

File Location

contract.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.

Bug ID

SSB_1700_14

Severity	Confidence	Line nos	Action Taken
● Gas	Certain	147-147	False Positive

Bug Type

CUSTOM ERRORS TO SAVE GAS

File Location

contract.sol

Issue Description

The contract was found to be using `revert()` statements. Since Solidity `v0.8.4`, custom errors have been introduced which are a better alternative to the revert. This allows the developers to pass custom errors with dynamic data while reverting the transaction and also making the whole implementation a bit cheaper than using `revert`.

Issue Remediation

It is recommended to replace all the instances of `revert()` statements with `error()` to save gas.

Bug ID

SSB_1700_46

Severity	Confidence	Line nos	Action Taken
● High	Tentative	571-574	False Positive

Bug Type

APPROVE FRONT-RUNNING ATTACK

File Location

contract.sol

Issue Description

The `approve()` method overrides current allowance regardless of whether the spender already used it or not, so there is no way to increase or decrease allowance by a certain value atomically unless the token owner is a smart contract, not an account.
This can be abused by a token receiver when they try to withdraw certain tokens from the sender's account.
Meanwhile, if the sender decides to change the amount and sends another `approve` transaction, the receiver can notice this transaction before it's mined and can extract tokens from both the transactions, therefore, ending up with tokens from both the transactions. This is a front-running attack affecting the `ERC20 Approve` function.
The function approve can be front-run by abusing the `_approve` function.

Issue Remediation

Only use the approve function of the ERC/BEP standard to change the allowed amount to 0 or from 0 (wait till transaction is mined and approved).
Token owner just needs to make sure that the first transaction actually changed allowance from N to 0, i.e., that the spender didn't manage to transfer some of N allowed tokens before the first transaction was mined. Such checking is possible using advanced blockchain explorers such as [Etherscan.io](https://etherscan.io/)
Another way to mitigate the threat is to approve token transfers only to smart contracts with verified source code that does not contain logic for performing attacks like described above, and to accounts owned by the people you may trust.

Bug ID

SSB_1700_47

Severity	Confidence	Line nos	Action Taken
● High	Tentative	576-580	False Positive

Bug Type

APPROVE FRONT-RUNNING ATTACK

File Location

contract.sol

Issue Description

The `transferFrom()` method overrides current allowance regardless of whether the spender already used it or not, so there is no way to increase or decrease allowance by a certain value atomically unless the token owner is a smart contract, not an account.
This can be abused by a token receiver when they try to withdraw certain tokens from the sender's account.
Meanwhile, if the sender decides to change the amount and sends another `approve` transaction, the receiver can notice this transaction before it's mined and can extract tokens from both the transactions, therefore, ending up with tokens from both the transactions. This is a front-running attack affecting the `ERC20 Approve` function.
The function transferFrom can be front-run by abusing the `_approve` function.

Issue Remediation

Only use the approve function of the ERC/BEP standard to change the allowed amount to 0 or from 0 (wait till transaction is mined and approved).
Token owner just needs to make sure that the first transaction actually changed allowance from N to 0, i.e., that the spender didn't manage to transfer some of N allowed tokens before the first transaction was mined. Such checking is possible using advanced blockchain explorers such as [Etherscan.io](https://etherscan.io/)
Another way to mitigate the threat is to approve token transfers only to smart contracts with verified source code that does not contain logic for performing attacks like described above, and to accounts owned by the people you may trust.

Bug ID

SSB_1700_48

Severity	Confidence	Line nos	Action Taken
● High	Tentative	726-744	False Positive

Bug Type

APPROVE FRONT-RUNNING ATTACK

File Location

contract.sol

Issue Description

The `swapTokensForEth()` method overrides current allowance regardless of whether the spender already used it or not, so there is no way to increase or decrease allowance by a certain value atomically unless the token owner is a smart contract, not an account.
This can be abused by a token receiver when they try to withdraw certain tokens from the sender's account.
Meanwhile, if the sender decides to change the amount and sends another `approve` transaction, the receiver can notice this transaction before it's mined and can extract tokens from both the transactions, therefore, ending up with tokens from both the transactions. This is a front-running attack affecting the `ERC20 Approve` function.
The function swapTokensForEth can be front-run by abusing the `_approve` function.

Issue Remediation

Only use the approve function of the ERC/BEP standard to change the allowed amount to 0 or from 0 (wait till transaction is mined and approved).
Token owner just needs to make sure that the first transaction actually changed allowance from N to 0, i.e., that the spender didn't manage to transfer some of N allowed tokens before the first transaction was mined. Such checking is possible using advanced blockchain explorers such as [Etherscan.io](https://etherscan.io/)
Another way to mitigate the threat is to approve token transfers only to smart contracts with verified source code that does not contain logic for performing attacks like described above, and to accounts owned by the people you may trust.

Bug ID

SSB_1700_49

Severity	Confidence	Line nos	Action Taken
● High	Tentative	763-776	False Positive

Bug Type

APPROVE FRONT-RUNNING ATTACK

File Location

contract.sol

Issue Description

The `addLiquidity()` method overrides current allowance regardless of whether the spender already used it or not, so there is no way to increase or decrease allowance by a certain value atomically unless the token owner is a smart contract, not an account.
This can be abused by a token receiver when they try to withdraw certain tokens from the sender's account.
Meanwhile, if the sender decides to change the amount and sends another `approve` transaction, the receiver can notice this transaction before it's mined and can extract tokens from both the transactions, therefore, ending up with tokens from both the transactions. This is a front-running attack affecting the `ERC20 Approve` function.
The function addLiquidity can be front-run by abusing the `_approve` function.

Issue Remediation

Only use the approve function of the ERC/BEP standard to change the allowed amount to 0 or from 0 (wait till transaction is mined and approved).
Token owner just needs to make sure that the first transaction actually changed allowance from N to 0, i.e., that the spender didn't manage to transfer some of N allowed tokens before the first transaction was mined. Such checking is possible using advanced blockchain explorers such as [Etherscan.io](https://etherscan.io/)
Another way to mitigate the threat is to approve token transfers only to smart contracts with verified source code that does not contain logic for performing attacks like described above, and to accounts owned by the people you may trust.

Bug ID
SSB_1700_2

Severity	Confidence	Line nos	Action Taken
● Gas	Firm	647-647	False Positive

Bug Type
EXTRA GAS USAGE IN LOOPS

File Location
contract.sol

Issue Description

State variables such as `.balance`, or `.length` of a non-memory array are used in the condition of `for` or `while` loop.

In this case, every iteration of the loop consumes extra gas because state variables are being referred to.

Issue Remediation

If state variables such as `.balance`, or `.length` are used several times, holding its value in a local variable is more gas efficient. If `.length` of calldata-array is placed into a local variable, the optimization will be less significant.

Bug ID

SSB_1700_3

Severity	Confidence	Line nos	Action Taken
● Gas	Firm	870-870	False Positive

Bug Type

EXTRA GAS USAGE IN LOOPS

File Location

contract.sol

Issue Description

State variables such as `.balance`, or `.length` of a non-memory array are used in the condition of `for` or `while` loop.
In this case, every iteration of the loop consumes extra gas because state variables are being referred to.

Issue Remediation

If state variables such as `.balance`, or `.length` are used several times, holding its value in a local variable is more gas efficient. If `.length` of calldata-array is placed into a local variable, the optimization will be less significant.

Bug ID

SSB_1700_29

Severity	Confidence	Line nos	Action Taken

● Low	Certain	12-12	False Positive

Bug Type

USE OF FLOATING PRAGMA

File Location

contract.sol

Issue Description

Solidity source files indicate the versions of the compiler they can be compiled with using a pragma directive at the top of the solidity file. This can either be a floating pragma or a specific compiler version.
The contract was found to be using a floating pragma which is not considered safe as it can be compiled with all the versions described.
The following affected files were found to be using floating pragma:
`['contract.sol'] - ^0.8.5`

Issue Remediation

It is recommended to use a fixed pragma version, as future compiler versions may handle certain language constructions in a way the developer did not foresee.
Using a floating pragma may introduce several vulnerabilities if compiled with an older version.
The developers should always use the exact Solidity compiler version when designing their contracts as it may break the changes in the future.
Instead of `^0.8.5` use `pragma solidity 0.8.7`, which is a stable and recommended version right now.

Bug ID

SSB_1700_50

Severity	Confidence	Line nos	Action Taken
● Gas	Tentative	841-845	False Positive

Bug Type

FUNCTION SHOULD RETURN STRUCT

File Location

contract.sol

Issue Description

The function _getValues was detected to be returning multiple values.
Consider using a `struct` instead of multiple return values for the function. It can improve code readability.

Issue Remediation

Use `struct` for returning multiple values inside a function, which returns several parameters and improves code readability.

Bug ID

SSB_1700_4

Severity	Confidence	Line nos	Action Taken
● High	Tentative	647-647	False Positive

Bug Type

UNCHECKED ARRAY LENGTH

File Location

contract.sol

Issue Description

Ethereum is a very resource-constrained environment. Prices per computational step are orders of magnitude higher than with centralized providers. Moreover, Ethereum miners impose a limit on the total number of Gas consumed in a block. If `array.length` is large enough, the function exceeds the block gas limit, and transactions calling it will never be confirmed.
`for (uint256 i = 0; i < array.length ; i++) { cosltyFunc(); }`
This becomes a security issue, if an external actor influences `array.length`.

E.g., if an array enumerates all registered addresses, an adversary can register many addresses, causing the problem described above.

Issue Remediation

Either explicitly or just due to normal operation, the number of iterations in a loop can grow beyond the block gas limit, which can cause the complete contract to be stalled at a certain point. Therefore, loops with a bigger or unknown number of steps should always be avoided.

Bug ID

SSB_1700_6

Severity	Confidence	Line nos	Action Taken
• Gas	Tentative	647-647	False Positive

Bug Type

GAS OPTIMIZATION IN INCREMENTS

File Location

contract.sol

Issue Description

`++i` costs less gas compared to `i++` or `i += 1` for unsigned integers. In `i++`, the compiler has to create a temporary variable to store the initial value. This is not the case with `++i` in which the value is directly incremented and returned, thus, making it a cheaper alternative.

Issue Remediation

Consider changing the post-increments `(i++)` to pre-increments `(++i)` as long as the value is not used in any calculations or inside returns. Make sure that the logic of the code is not changed.

Bug ID

SSB_1700_7

Severity	Confidence	Line nos	Action Taken
• Gas	Tentative	870-870	False Positive

Bug Type

GAS OPTIMIZATION IN INCREMENTS

File Location

contract.sol

Issue Description

`++i` costs less gas compared to `i++` or `i += 1` for unsigned integers. In `i++`, the compiler has to create a temporary variable to store the initial value. This is not the case with `++i` in which the value is directly incremented and returned, thus, making it a cheaper alternative.

Issue Remediation

Consider changing the post-increments `(i++)` to pre-increments `(++i)` as long as the value is not used in any calculations or inside returns. Make sure that the logic of the code is not changed.

Bug ID

SSB_1700_80

Severity	Confidence	Line nos	Action Taken
● Gas	Tentative	66-66	Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

contract.sol

Issue Description

The `require()` and `revert()` functions take an input string to show errors if the validation fails.
This strings inside these functions that are longer than `32 bytes` require at least one additional `MSTORE`, along with additional overhead for computing memory offset, and other parameters.

Issue Remediation

It is recommended to short the strings passed inside `require()` and `revert()` to fit under `32 bytes`. This will decrease the gas usage at the time of deployment and at runtime when the validation condition is met.

Bug ID

SSB_1700_81

Severity	Confidence	Line nos	Action Taken
● Gas	Tentative	112-112	Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

contract.sol

Issue Description

The `require()` and `revert()` functions take an input string to show errors if the validation fails.
This strings inside these functions that are longer than `32 bytes` require at least one additional `MSTORE`, along with additional overhead for computing memory offset, and other parameters.

Issue Remediation

It is recommended to short the strings passed inside `require()` and `revert()` to fit under `32 bytes`. This will decrease the gas usage at the time of deployment and at runtime when the validation condition is met.

Bug ID

SSB_1700_82

Severity	Confidence	Line nos	Action Taken
● Gas	Tentative	129-129	Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

contract.sol

Issue Description

The `require()` and `revert()` functions take an input string to show errors if the validation fails.
This strings inside these functions that are longer than `32 bytes` require at least one additional `MSTORE`, along with additional overhead for computing memory offset, and other parameters.

Issue Remediation

It is recommended to short the strings passed inside `require()` and `revert()` to fit under `32 bytes`. This will decrease the gas usage at the time of deployment and at runtime when the validation condition is met.

Bug ID

SSB_1700_83

Severity	Confidence	Line nos	Action Taken
● Gas	Tentative	181-181	Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

contract.sol

Issue Description

The `require()` and `revert()` functions take an input string to show errors if the validation fails.
This strings inside these functions that are longer than `32 bytes` require at least one additional `MSTORE`, along with additional overhead for computing memory offset, and other parameters.

Issue Remediation

It is recommended to short the strings passed inside `require()` and `revert()` to fit under `32 bytes`. This will decrease the gas usage at the time of deployment and at runtime when the validation condition is met.

Bug ID

SSB_1700_84

Severity	Confidence	Line nos	Action Taken
● Gas	Tentative	202-202	Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

contract.sol

Issue Description

The `require()` and `revert()` functions take an input string to show errors if the validation fails.
This strings inside these functions that are longer than `32 bytes` require at least one additional `MSTORE`, along with additional overhead for computing memory offset, and other parameters.

Issue Remediation

It is recommended to short the strings passed inside `require()` and `revert()` to fit under `32 bytes`. This will decrease the gas usage at the time of deployment and at runtime when the validation condition is met.

Bug ID

SSB_1700_85

Severity	Confidence	Line nos	Action Taken
● Gas	Tentative	500-500	Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

contract.sol

Issue Description

The `require()` and `revert()` functions take an input string to show errors if the validation fails.
This strings inside these functions that are longer than `32 bytes` require at least one additional `MSTORE`, along with additional overhead for computing memory offset, and other parameters.

Issue Remediation

It is recommended to short the strings passed inside `require()` and `revert()` to fit under `32 bytes`. This will decrease the gas usage at the time of deployment and at runtime when the validation condition is met.

Bug ID

SSB_1700_86

Severity	Confidence	Line nos	Action Taken
● Gas	Tentative	610-610	Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

contract.sol

Issue Description

The `require()` and `revert()` functions take an input string to show errors if the validation fails.
This strings inside these functions that are longer than `32 bytes` require at least one additional `MSTORE`, along with additional overhead for computing memory offset, and other parameters.

Issue Remediation

It is recommended to short the strings passed inside `require()` and `revert()` to fit under `32 bytes`. This will decrease the gas usage at the time of deployment and at runtime when the validation condition is met.

Bug ID

SSB_1700_87

Severity	Confidence	Line nos	Action Taken
● Gas	Tentative	630-630	Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

contract.sol

Issue Description

The `require()` and `revert()` functions take an input string to show errors if the validation fails.
This strings inside these functions that are longer than `32 bytes` require at least one additional `MSTORE`, along with additional overhead for computing memory offset, and other parameters.

Issue Remediation

It is recommended to short the strings passed inside `require()` and `revert()` to fit under `32 bytes`. This will decrease the gas usage at the time of deployment and at runtime when the validation condition is met.

Bug ID

SSB_1700_88

Severity	Confidence	Line nos	Action Taken
● Gas	Tentative	659-659	Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

contract.sol

Issue Description

The `require()` and `revert()` functions take an input string to show errors if the validation fails.
This strings inside these functions that are longer than `32 bytes` require at least one additional `MSTORE`, along with additional overhead for computing memory offset, and other parameters.

Issue Remediation

It is recommended to short the strings passed inside `require()` and `revert()` to fit under `32 bytes`. This will decrease the gas usage at the time of deployment and at runtime when the validation condition is met.

Bug ID

SSB_1700_89

Severity	Confidence	Line nos	Action Taken
● Gas	Tentative	660-660	Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

contract.sol

Issue Description

The `require()` and `revert()` functions take an input string to show errors if the validation fails.
This strings inside these functions that are longer than `32 bytes` require at least one additional `MSTORE`, along with additional overhead for computing memory offset, and other parameters.

Issue Remediation

It is recommended to short the strings passed inside `require()` and `revert()` to fit under `32 bytes`. This will decrease the gas usage at the time of deployment and at runtime when the validation condition is met.

Bug ID

SSB_1700_90

Severity	Confidence	Line nos	Action Taken
Gas	Tentative	671-671	Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

contract.sol

Issue Description

The `require()` and `revert()` functions take an input string to show errors if the validation fails.
This strings inside these functions that are longer than `32 bytes` require at least one additional `MSTORE`, along with additional overhead for computing memory offset, and other parameters.

Issue Remediation

It is recommended to short the strings passed inside `require()` and `revert()` to fit under `32 bytes`. This will decrease the gas usage at the time of deployment and at runtime when the validation condition is met.

Bug ID

SSB_1700_91

Severity	Confidence	Line nos	Action Taken
Gas	Tentative	672-672	Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

contract.sol

Issue Description

The `require()` and `revert()` functions take an input string to show errors if the validation fails.
This strings inside these functions that are longer than `32 bytes` require at least one additional `MSTORE`, along with additional overhead for computing memory offset, and other parameters.

Issue Remediation

It is recommended to short the strings passed inside `require()` and `revert()` to fit under `32 bytes`. This will decrease the gas usage at the time of deployment and at runtime when the validation condition is met.

Bug ID

SSB_1700_92

Severity	Confidence	Line nos	Action Taken
● Gas	Tentative	673-673	Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

contract.sol

Issue Description

The `require()` and `revert()` functions take an input string to show errors if the validation fails.
This strings inside these functions that are longer than `32 bytes` require at least one additional `MSTORE`, along with additional overhead for computing memory offset, and other parameters.

Issue Remediation

It is recommended to short the strings passed inside `require()` and `revert()` to fit under `32 bytes`. This will decrease the gas usage at the time of deployment and at runtime when the validation condition is met.

Bug ID

SSB_1700_93

Severity	Confidence	Line nos	Action Taken
● Gas	Tentative	675-675	Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

contract.sol

Issue Description

The `require()` and `revert()` functions take an input string to show errors if the validation fails.
This strings inside these functions that are longer than `32 bytes` require at least one additional `MSTORE`, along with additional overhead for computing memory offset, and other parameters.

Issue Remediation

It is recommended to short the strings passed inside `require()` and `revert()` to fit under `32 bytes`. This will decrease the gas usage at the time of deployment and at runtime when the validation condition is met.

Bug ID

SSB_1700_94

Severity	Confidence	Line nos	Action Taken
● Gas	Tentative	946-946	Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

contract.sol

Issue Description

The `require()` and `revert()` functions take an input string to show errors if the validation fails.
This strings inside these functions that are longer than `32 bytes` require at least one additional `MSTORE`, along with additional overhead for computing memory offset, and other parameters.

Issue Remediation

It is recommended to short the strings passed inside `require()` and `revert()` to fit under `32 bytes`. This will decrease the gas usage at the time of deployment and at runtime when the validation condition is met.

Bug ID

SSB_1700_58

Severity	Confidence	Line nos	Action Taken
● Low	Firm	107-113	Pending Fix

Bug Type

MISSING EVENTS

File Location

contract.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 Address was found to be missing these events on the function sendValue 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_1700_59

Severity	Confidence	Line nos	Action Taken
● Low	Firm	116-118	Pending Fix

Bug Type

MISSING EVENTS

File Location

contract.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 Address was found to be missing these events on the function functionCall 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_1700_60

Severity	Confidence	Line nos	Action Taken
● Low	Firm	120-122	Pending Fix

Bug Type

MISSING EVENTS

File Location

contract.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 Address was found to be missing these events on the function functionCall 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_1700_61

Severity	Confidence	Line nos	Action Taken
● Low	Firm	124-126	Pending Fix

Bug Type

MISSING EVENTS

File Location

contract.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 Address was found to be missing these events on the function functionCallWithValue 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_1700_62

Severity	Confidence	Line nos	Action Taken
● Low	Firm	128-131	Pending Fix

Bug Type

MISSING EVENTS

File Location

contract.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 Address was found to be missing these events on the function functionCallWithValue 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_1700_63

Severity	Confidence	Line nos	Action Taken
● Low	Firm	133-150	Pending Fix

Bug Type

MISSING EVENTS

File Location

contract.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 Address was found to be missing these events on the function _functionCallWithValue 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_1700_64

Severity	Confidence	Line nos	Action Taken
● **Low**	Firm	**608-615**	Pending Fix

Bug Type

MISSING EVENTS

File Location

contract.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 BuyBackToken was found to be missing these events on the function deliver 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_1700_65

Severity	Confidence	Line nos	Action Taken
● Low	Firm	635-643	Pending Fix

Bug Type

MISSING EVENTS

File Location

contract.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 BuyBackToken was found to be missing these events on the function excludeFromReward 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_1700_66

Severity	Confidence	Line nos	Action Taken
● Low	Firm	645-656	Pending Fix

Bug Type

MISSING EVENTS

File Location

contract.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 BuyBackToken was found to be missing these events on the function includeInReward 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_1700_67

Severity	Confidence	Line nos	Action Taken
● Low	Firm	836-839	Pending Fix

Bug Type

MISSING EVENTS

File Location

contract.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 BuyBackToken was found to be missing these events on the function _reflectFee 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_1700_68

Severity	Confidence	Line nos	Action Taken
● Low	Firm	879-885	Pending Fix

Bug Type

MISSING EVENTS

File Location

contract.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 BuyBackToken was found to be missing these events on the function _takeLiquidity 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_1700_69

Severity	Confidence	Line nos	Action Taken
● Low	Firm	899-911	Pending Fix

Bug Type

MISSING EVENTS

File Location

contract.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 BuyBackToken was found to be missing these events on the function removeAllFee 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_1700_70

Severity	Confidence	Line nos	Action Taken
● Low	Firm	913-918	Pending Fix

Bug Type

MISSING EVENTS

File Location

contract.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 BuyBackToken was found to be missing these events on the function restoreAllFee 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_1700_71

Severity	Confidence	Line nos	Action Taken
● Low	Firm	924-926	Pending Fix

Bug Type

MISSING EVENTS

File Location

contract.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 BuyBackToken was found to be missing these events on the function excludeFromFee 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_1700_72

Severity	Confidence	Line nos	Action Taken
● Low	Firm	928-930	Pending Fix

Bug Type

MISSING EVENTS

File Location

contract.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 BuyBackToken was found to be missing these events on the function includeInFee 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_1700_73

Severity	Confidence	Line nos	Action Taken
● Low	Firm	932-934	Pending Fix

Bug Type

MISSING EVENTS

File Location

contract.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 BuyBackToken was found to be missing these events on the function setTaxFee 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_1700_74

Severity	Confidence	Line nos	Action Taken
● **Low**	Firm	**936-939**	Pending Fix

Bug Type

MISSING EVENTS

File Location

contract.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 BuyBackToken was found to be missing these events on the function setBuybackFee 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_1700_75

Severity	Confidence	Line nos	Action Taken
● Low	Firm	941-943	Pending Fix

Bug Type

MISSING EVENTS

File Location

contract.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 BuyBackToken was found to be missing these events on the function setMaxTxAmount 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_1700_76

Severity	Confidence	Line nos	Action Taken
● Low	Firm	945-949	Pending Fix

Bug Type

MISSING EVENTS

File Location

contract.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 BuyBackToken was found to be missing these events on the function setMarketingFee 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_1700_77

Severity	Confidence	Line nos	Action Taken
● Low	Firm	951-953	Pending Fix

Bug Type

MISSING EVENTS

File Location

contract.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 BuyBackToken was found to be missing these events on the function setNumTokensSellToAddToLiquidity 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_1700_78

Severity	Confidence	Line nos	Action Taken
● Low	Firm	955-957	Pending Fix

Bug Type

MISSING EVENTS

File Location

contract.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 BuyBackToken was found to be missing these events on the function setBuybackUpperLimit 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_1700_79

Severity	Confidence	Line nos	Action Taken
● Low	Firm	959-961	Pending Fix

Bug Type

MISSING EVENTS

File Location

contract.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 BuyBackToken was found to be missing these events on the function setMarketingAddress 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_1700_51

Severity	Confidence	Line nos	Action Taken
Informational	Certain	260-260	Pending Fix

Bug Type

MISSING INDEXED KEYWORDS IN EVENTS

File Location

contract.sol

Issue Description

Events are essential for tracking off-chain data and when the event paraemters are `indexed` they can be used as filter options which will help getting only the specific data instead of all the logs.

Issue Remediation

Consider adding `indexed` keyword to crucial event parameters that could be used in off-chain tracking. Do remember that the `indexed` keyword costs more gas.

Bug ID

SSB_1700_52

Severity	Confidence	Line nos	Action Taken
Informational	Certain	**473-473**	Pending Fix

Bug Type

MISSING INDEXED KEYWORDS IN EVENTS

File Location

contract.sol

Issue Description

Events are essential for tracking off-chain data and when the event paraemters are `indexed` they can be used as filter options which will help getting only the specific data instead of all the logs.

Issue Remediation

Consider adding `indexed` keyword to crucial event parameters that could be used in off-chain tracking. Do remember that the `indexed` keyword costs more gas.

Bug ID

SSB_1700_53

Severity	Confidence	Line nos	Action Taken
Informational	Certain	474-474	Pending Fix

Bug Type

MISSING INDEXED KEYWORDS IN EVENTS

File Location

contract.sol

Issue Description

Events are essential for tracking off-chain data and when the event paraemters are `indexed` they can be used as filter options which will help getting only the specific data instead of all the logs.

Issue Remediation

Consider adding `indexed` keyword to crucial event parameters that could be used in off-chain tracking. Do remember that the `indexed` keyword costs more gas.

Bug ID

SSB_1700_54

Severity	Confidence	Line nos	Action Taken
Informational	Certain	475-475	Pending Fix

Bug Type

MISSING INDEXED KEYWORDS IN EVENTS

File Location

contract.sol

Issue Description

Events are essential for tracking off-chain data and when the event paraemters are `indexed` they can be used as filter options which will help getting only the specific data instead of all the logs.

Issue Remediation

Consider adding `indexed` keyword to crucial event parameters that could be used in off-chain tracking. Do remember that the `indexed` keyword costs more gas.

Bug ID

SSB_1700_55

Severity	Confidence	Line nos	Action Taken
Informational	Certain	**476-480**	Pending Fix

Bug Type

MISSING INDEXED KEYWORDS IN EVENTS

File Location

contract.sol

Issue Description

Events are essential for tracking off-chain data and when the event paraemters are `indexed` they can be used as filter options which will help getting only the specific data instead of all the logs.

Issue Remediation

Consider adding `indexed` keyword to crucial event parameters that could be used in off-chain tracking. Do remember that the `indexed` keyword costs more gas.

Bug ID

SSB_1700_56

Severity	Confidence	Line nos	Action Taken
Informational	Certain	**482-485**	Pending Fix

Bug Type

MISSING INDEXED KEYWORDS IN EVENTS

File Location

contract.sol

Issue Description

Events are essential for tracking off-chain data and when the event paraemters are `indexed` they can be used as filter options which will help getting only the specific data instead of all the logs.

Issue Remediation

Consider adding `indexed` keyword to crucial event parameters that could be used in off-chain tracking. Do remember that the `indexed` keyword costs more gas.

Bug ID

SSB_1700_57

Severity	Confidence	Line nos	Action Taken
Informational	Certain	**487-490**	Pending Fix

Bug Type

MISSING INDEXED KEYWORDS IN EVENTS

File Location

contract.sol

Issue Description

Events are essential for tracking off-chain data and when the event paraemters are `indexed` they can be used as filter options which will help getting only the specific data instead of all the logs.

Issue Remediation

Consider adding `indexed` keyword to crucial event parameters that could be used in off-chain tracking. Do remember that the `indexed` keyword costs more gas.

Bug ID

SSB_1700_28

Severity	Confidence	Line nos	Action Taken
● Low	Certain	12-12	False Positive

Bug Type

OUTDATED COMPILER VERSION

File Location

contract.sol

Issue Description

Using an outdated compiler version can be problematic especially if there are publicly disclosed bugs and issues that affect the current compiler version.
The following outdated versions were detected:
`['contract.sol']` - `^0.8.5`

Issue Remediation

It is recommended to use a recent version of the Solidity compiler that should not be the most recent version, and it should not be an outdated version as well. Using very old versions of Solidity prevents the benefits of bug fixes and newer security checks. Consider using the solidity version `0.8.7`, which patches most solidity vulnerabilities.

Bug ID

SSB_1700_98

Severity	Confidence	Line nos	Action Taken
Informational	Firm	175-178	Pending Fix

Bug Type

PRESENCE OF OVERPOWERED ROLE

File Location

contract.sol

Issue Description

The overpowered owner (i.e., the person who has too much power) is a project design where the contract is tightly coupled to their owner (or owners); only they can manually invoke critical functions.
Due to the fact that this function is only accessible from a single address, the system is heavily dependent on the address of the owner. In this case, there are scenarios that may lead to undesirable consequences for investors, e.g., if the private key of this address is compromised, then an attacker can take control of the contract.

Issue Remediation

We recommend designing contracts in a trust-less manner. For instance, this functionality can be implemented in the contract's constructor. Another option is to use a MultiSig wallet for this address. For systems that are provisioned for a single user, you can use [Ownable.sol].
For systems that require provisioning users in a group, you can use [@openzeppelin/Roles.sol] or [@hq20/Whitelist.sol].

Bug ID

SSB_1700_99

Severity	Confidence	Line nos	Action Taken
Informational	Firm	180-184	Pending Fix

Bug Type

PRESENCE OF OVERPOWERED ROLE

File Location

contract.sol

Issue Description

The overpowered owner (i.e., the person who has too much power) is a project design where the contract is tightly coupled to their owner (or owners); only they can manually invoke critical functions.
Due to the fact that this function is only accessible from a single address, the system is heavily dependent on the address of the owner. In this case, there are scenarios that may lead to undesirable consequences for investors, e.g., if the private key of this address is compromised, then an attacker can take control of the contract.

Issue Remediation

We recommend designing contracts in a trust-less manner. For instance, this functionality can be implemented in the contract's constructor. Another option is to use a MultiSig wallet for this address. For systems that are provisioned for a single user, you can use [Ownable.sol].
For systems that require provisioning users in a group, you can use [@openzeppelin/Roles.sol] or [@hq20/Whitelist.sol].

Bug ID

SSB_1700_100

Severity	Confidence	Line nos	Action Taken
Informational	Firm	**194-199**	ⓘ Pending Fix

Bug Type

PRESENCE OF OVERPOWERED ROLE

File Location

contract.sol

Issue Description

The overpowered owner (i.e., the person who has too much power) is a project design where the contract is tightly coupled to their owner (or owners); only they can manually invoke critical functions.
Due to the fact that this function is only accessible from a single address, the system is heavily dependent on the address of the owner. In this case, there are scenarios that may lead to undesirable consequences for investors, e.g., if the private key of this address is compromised, then an attacker can take control of the contract.

Issue Remediation

We recommend designing contracts in a trust-less manner. For instance, this functionality can be implemented in the contract's constructor. Another option is to use a MultiSig wallet for this address. For systems that are provisioned for a single user, you can use [Ownable.sol].
For systems that require provisioning users in a group, you can use [@openzeppelin/Roles.sol] or [@hq20/Whitelist.sol].

Bug ID

SSB_1700_101

Severity	Confidence	Line nos	Action Taken
Informational	Firm	635-643	Pending Fix

Bug Type

PRESENCE OF OVERPOWERED ROLE

File Location

contract.sol

Issue Description

The overpowered owner (i.e., the person who has too much power) is a project design where the contract is tightly coupled to their owner (or owners); only they can manually invoke critical functions.
Due to the fact that this function is only accessible from a single address, the system is heavily dependent on the address of the owner. In this case, there are scenarios that may lead to undesirable consequences for investors, e.g., if the private key of this address is compromised, then an attacker can take control of the contract.

Issue Remediation

We recommend designing contracts in a trust-less manner. For instance, this functionality can be implemented in the contract's constructor. Another option is to use a MultiSig wallet for this address. For systems that are provisioned for a single user, you can use [Ownable.sol].
For systems that require provisioning users in a group, you can use [@openzeppelin/Roles.sol] or [@hq20/Whitelist.sol].

Bug ID

SSB_1700_102

Severity	Confidence	Line nos	Action Taken
Informational	Firm	645-656	Pending Fix

Bug Type

PRESENCE OF OVERPOWERED ROLE

File Location

contract.sol

Issue Description

The overpowered owner (i.e., the person who has too much power) is a project design where the contract is tightly coupled to their owner (or owners); only they can manually invoke critical functions.
Due to the fact that this function is only accessible from a single address, the system is heavily dependent on the address of the owner. In this case, there are scenarios that may lead to undesirable consequences for investors, e.g., if the private key of this address is compromised, then an attacker can take control of the contract.

Issue Remediation

We recommend designing contracts in a trust-less manner. For instance, this functionality can be implemented in the contract's constructor. Another option is to use a MultiSig wallet for this address. For systems that are provisioned for a single user, you can use [Ownable.sol].
For systems that require provisioning users in a group, you can use [@openzeppelin/Roles.sol] or [@hq20/Whitelist.sol].

Bug ID

SSB_1700_103

Severity	Confidence	Line nos	Action Taken
Informational	Firm	**924-926**	ⓘ Pending Fix

Bug Type

PRESENCE OF OVERPOWERED ROLE

File Location

contract.sol

Issue Description

The overpowered owner (i.e., the person who has too much power) is a project design where the contract is tightly coupled to their owner (or owners); only they can manually invoke critical functions.
Due to the fact that this function is only accessible from a single address, the system is heavily dependent on the address of the owner. In this case, there are scenarios that may lead to undesirable consequences for investors, e.g., if the private key of this address is compromised, then an attacker can take control of the contract.

Issue Remediation

We recommend designing contracts in a trust-less manner. For instance, this functionality can be implemented in the contract's constructor. Another option is to use a MultiSig wallet for this address. For systems that are provisioned for a single user, you can use [Ownable.sol].
For systems that require provisioning users in a group, you can use [@openzeppelin/Roles.sol] or [@hq20/Whitelist.sol].

Bug ID

SSB_1700_104

Severity	Confidence	Line nos	Action Taken
Informational	Firm	928-930	Pending Fix

Bug Type

PRESENCE OF OVERPOWERED ROLE

File Location

contract.sol

Issue Description

The overpowered owner (i.e., the person who has too much power) is a project design where the contract is tightly coupled to their owner (or owners); only they can manually invoke critical functions.
Due to the fact that this function is only accessible from a single address, the system is heavily dependent on the address of the owner. In this case, there are scenarios that may lead to undesirable consequences for investors, e.g., if the private key of this address is compromised, then an attacker can take control of the contract.

Issue Remediation

We recommend designing contracts in a trust-less manner. For instance, this functionality can be implemented in the contract's constructor. Another option is to use a MultiSig wallet for this address. For systems that are provisioned for a single user, you can use [Ownable.sol].
For systems that require provisioning users in a group, you can use [@openzeppelin/Roles.sol] or [@hq20/Whitelist.sol].

Bug ID

SSB_1700_105

Severity	Confidence	Line nos	Action Taken
Informational	Firm	932-934	Pending Fix

Bug Type

PRESENCE OF OVERPOWERED ROLE

File Location

contract.sol

Issue Description

The overpowered owner (i.e., the person who has too much power) is a project design where the contract is tightly coupled to their owner (or owners); only they can manually invoke critical functions.
Due to the fact that this function is only accessible from a single address, the system is heavily dependent on the address of the owner. In this case, there are scenarios that may lead to undesirable consequences for investors, e.g., if the private key of this address is compromised, then an attacker can take control of the contract.

Issue Remediation

We recommend designing contracts in a trust-less manner. For instance, this functionality can be implemented in the contract's constructor. Another option is to use a MultiSig wallet for this address. For systems that are provisioned for a single user, you can use [Ownable.sol].
For systems that require provisioning users in a group, you can use [@openzeppelin/Roles.sol] or [@hq20/Whitelist.sol].

Bug ID

SSB_1700_106

Severity	Confidence	Line nos	Action Taken
Informational	Firm	936-939	Pending Fix

Bug Type

PRESENCE OF OVERPOWERED ROLE

File Location

contract.sol

Issue Description

The overpowered owner (i.e., the person who has too much power) is a project design where the contract is tightly coupled to their owner (or owners); only they can manually invoke critical functions.
Due to the fact that this function is only accessible from a single address, the system is heavily dependent on the address of the owner. In this case, there are scenarios that may lead to undesirable consequences for investors, e.g., if the private key of this address is compromised, then an attacker can take control of the contract.

Issue Remediation

We recommend designing contracts in a trust-less manner. For instance, this functionality can be implemented in the contract's constructor. Another option is to use a MultiSig wallet for this address. For systems that are provisioned for a single user, you can use [Ownable.sol].
For systems that require provisioning users in a group, you can use [@openzeppelin/Roles.sol] or [@hq20/Whitelist.sol].

Bug ID

SSB_1700_107

Severity	Confidence	Line nos	Action Taken
Informational	Firm	**941-943**	Pending Fix

Bug Type

PRESENCE OF OVERPOWERED ROLE

File Location

contract.sol

Issue Description

The overpowered owner (i.e., the person who has too much power) is a project design where the contract is tightly coupled to their owner (or owners); only they can manually invoke critical functions.
Due to the fact that this function is only accessible from a single address, the system is heavily dependent on the address of the owner. In this case, there are scenarios that may lead to undesirable consequences for investors, e.g., if the private key of this address is compromised, then an attacker can take control of the contract.

Issue Remediation

We recommend designing contracts in a trust-less manner. For instance, this functionality can be implemented in the contract's constructor. Another option is to use a MultiSig wallet for this address. For systems that are provisioned for a single user, you can use [Ownable.sol].
For systems that require provisioning users in a group, you can use [@openzeppelin/Roles.sol] or [@hq20/Whitelist.sol].

Bug ID

SSB_1700_108

Severity	Confidence	Line nos	Action Taken
Informational	Firm	945-949	Pending Fix

Bug Type

PRESENCE OF OVERPOWERED ROLE

File Location

contract.sol

Issue Description

The overpowered owner (i.e., the person who has too much power) is a project design where the contract is tightly coupled to their owner (or owners); only they can manually invoke critical functions.
Due to the fact that this function is only accessible from a single address, the system is heavily dependent on the address of the owner. In this case, there are scenarios that may lead to undesirable consequences for investors, e.g., if the private key of this address is compromised, then an attacker can take control of the contract.

Issue Remediation

We recommend designing contracts in a trust-less manner. For instance, this functionality can be implemented in the contract's constructor. Another option is to use a MultiSig wallet for this address. For systems that are provisioned for a single user, you can use [Ownable.sol].
For systems that require provisioning users in a group, you can use [@openzeppelin/Roles.sol] or [@hq20/Whitelist.sol].

Bug ID

SSB_1700_109

Severity	Confidence	Line nos	Action Taken
Informational	Firm	951-953	Pending Fix

Bug Type

PRESENCE OF OVERPOWERED ROLE

File Location

contract.sol

Issue Description

The overpowered owner (i.e., the person who has too much power) is a project design where the contract is tightly coupled to their owner (or owners); only they can manually invoke critical functions.
Due to the fact that this function is only accessible from a single address, the system is heavily dependent on the address of the owner. In this case, there are scenarios that may lead to undesirable consequences for investors, e.g., if the private key of this address is compromised, then an attacker can take control of the contract.

Issue Remediation

We recommend designing contracts in a trust-less manner. For instance, this functionality can be implemented in the contract's constructor. Another option is to use a MultiSig wallet for this address. For systems that are provisioned for a single user, you can use [Ownable.sol].
For systems that require provisioning users in a group, you can use [@openzeppelin/Roles.sol] or [@hq20/Whitelist.sol].

Bug ID

SSB_1700_110

Severity	Confidence	Line nos	Action Taken
Informational	Firm	955-957	Pending Fix

Bug Type

PRESENCE OF OVERPOWERED ROLE

File Location

contract.sol

Issue Description

The overpowered owner (i.e., the person who has too much power) is a project design where the contract is tightly coupled to their owner (or owners); only they can manually invoke critical functions.
Due to the fact that this function is only accessible from a single address, the system is heavily dependent on the address of the owner. In this case, there are scenarios that may lead to undesirable consequences for investors, e.g., if the private key of this address is compromised, then an attacker can take control of the contract.

Issue Remediation

We recommend designing contracts in a trust-less manner. For instance, this functionality can be implemented in the contract's constructor. Another option is to use a MultiSig wallet for this address. For systems that are provisioned for a single user, you can use [Ownable.sol].
For systems that require provisioning users in a group, you can use [@openzeppelin/Roles.sol] or [@hq20/Whitelist.sol].

Bug ID

SSB_1700_111

Severity	Confidence	Line nos	Action Taken
Informational	Firm	**959-961**	Pending Fix

Bug Type

PRESENCE OF OVERPOWERED ROLE

File Location

contract.sol

Issue Description

The overpowered owner (i.e., the person who has too much power) is a project design where the contract is tightly coupled to their owner (or owners); only they can manually invoke critical functions.
Due to the fact that this function is only accessible from a single address, the system is heavily dependent on the address of the owner. In this case, there are scenarios that may lead to undesirable consequences for investors, e.g., if the private key of this address is compromised, then an attacker can take control of the contract.

Issue Remediation

We recommend designing contracts in a trust-less manner. For instance, this functionality can be implemented in the contract's constructor. Another option is to use a MultiSig wallet for this address. For systems that are provisioned for a single user, you can use [Ownable.sol].
For systems that require provisioning users in a group, you can use [@openzeppelin/Roles.sol] or [@hq20/Whitelist.sol].

Bug ID

SSB_1700_112

Severity	Confidence	Line nos	Action Taken
Informational	Firm	963-966	Pending Fix

Bug Type

PRESENCE OF OVERPOWERED ROLE

File Location

contract.sol

Issue Description

The overpowered owner (i.e., the person who has too much power) is a project design where the contract is tightly coupled to their owner (or owners); only they can manually invoke critical functions.
Due to the fact that this function is only accessible from a single address, the system is heavily dependent on the address of the owner. In this case, there are scenarios that may lead to undesirable consequences for investors, e.g., if the private key of this address is compromised, then an attacker can take control of the contract.

Issue Remediation

We recommend designing contracts in a trust-less manner. For instance, this functionality can be implemented in the contract's constructor. Another option is to use a MultiSig wallet for this address. For systems that are provisioned for a single user, you can use [Ownable.sol].
For systems that require provisioning users in a group, you can use [@openzeppelin/Roles.sol] or [@hq20/Whitelist.sol].

Bug ID

SSB_1700_113

Severity	Confidence	Line nos	Action Taken
Informational	Firm	968-971	Pending Fix

Bug Type

PRESENCE OF OVERPOWERED ROLE

File Location

contract.sol

Issue Description

The overpowered owner (i.e., the person who has too much power) is a project design where the contract is tightly coupled to their owner (or owners); only they can manually invoke critical functions.
Due to the fact that this function is only accessible from a single address, the system is heavily dependent on the address of the owner. In this case, there are scenarios that may lead to undesirable consequences for investors, e.g., if the private key of this address is compromised, then an attacker can take control of the contract.

Issue Remediation

We recommend designing contracts in a trust-less manner. For instance, this functionality can be implemented in the contract's constructor. Another option is to use a MultiSig wallet for this address. For systems that are provisioned for a single user, you can use [Ownable.sol].
For systems that require provisioning users in a group, you can use [@openzeppelin/Roles.sol] or [@hq20/Whitelist.sol].

Bug ID

SSB_1700_114

Severity	Confidence	Line nos	Action Taken
Informational	Firm	973-984	Pending Fix

Bug Type

PRESENCE OF OVERPOWERED ROLE

File Location

contract.sol

Issue Description

The overpowered owner (i.e., the person who has too much power) is a project design where the contract is tightly coupled to their owner (or owners); only they can manually invoke critical functions.
Due to the fact that this function is only accessible from a single address, the system is heavily dependent on the address of the owner. In this case, there are scenarios that may lead to undesirable consequences for investors, e.g., if the private key of this address is compromised, then an attacker can take control of the contract.

Issue Remediation

We recommend designing contracts in a trust-less manner. For instance, this functionality can be implemented in the contract's constructor. Another option is to use a MultiSig wallet for this address. For systems that are provisioned for a single user, you can use [Ownable.sol].
For systems that require provisioning users in a group, you can use [@openzeppelin/Roles.sol] or [@hq20/Whitelist.sol].

Bug ID

SSB_1700_1

Severity	Confidence	Line nos	Action Taken
● Gas	Certain	421-421	False Positive

Bug Type

USE OF SAFEMATH LIBRARY

File Location

contract.sol

Issue Description

`SafeMath` library is found to be used in the contract. This increases gas consumption than traditional methods and validations if done manually.
Also, Solidity `0.8.0` includes checked arithmetic operations by default, and this renders `SafeMath` unnecessary.

Issue Remediation

We do not recommend using `SafeMath` library for all arithmetic operations. It is good practice to use explicit checks where it is really needed and to avoid extra checks where overflow/underflow is impossible.
The compiler should be upgraded to Solidity version `0.8.0+` which automatically checks for overflows and underflows.

Bug ID

SSB_1700_15

Severity	Confidence	Line nos	Action Taken
● Gas	Certain	201-206	Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

contract.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSB_1700_16

Severity	Confidence	Line nos	Action Taken
● Gas	Certain	194-199	Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

contract.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSB_1700_17

Severity	Confidence	Line nos	Action Taken
● Gas	Certain	180-184	Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

contract.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSB_1700_18

Severity	Confidence	Line nos	Action Taken
● Gas	Certain	175-178	Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

contract.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSB_1700_19

Severity	Confidence	Line nos	Action Taken
● Gas	Certain	968-971	Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

contract.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSB_1700_20

Severity	Confidence	Line nos	Action Taken
● Gas	Certain	608-615	Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

contract.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSB_1700_21

Severity	Confidence	Line nos	Action Taken
● Gas	Certain	635-643	Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

contract.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSB_1700_22

Severity	Confidence	Line nos	Action Taken
● Gas	Certain	576-580	Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

contract.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSB_1700_23

Severity	Confidence	Line nos	Action Taken
• Gas	Certain	587-590	Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

contract.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSB_1700_24

Severity	Confidence	Line nos	Action Taken
• Gas	Certain	571-574	Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

contract.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSB_1700_25

Severity	Confidence	Line nos	Action Taken
• Gas	Certain	928-930	Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

contract.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSB_1700_26

Severity	Confidence	Line nos	Action Taken
● Gas	Certain	924-926	Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

contract.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSB_1700_27

Severity	Confidence	Line nos	Action Taken
● Gas	Certain	582-585	Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

contract.sol

Issue Description

A function with `public` visibility modifier was detected that is not called internally. `public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Issue Remediation

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

Bug ID

SSB_1700_44

Severity	Confidence	Line nos	Action Taken
Informational	Certain	103-103	False Positive

Bug Type

IN-LINE ASSEMBLY DETECTED

File Location

contract.sol

Issue Description

Inline assembly is a way to access the Ethereum Virtual Machine at a low level. This bypasses several important safety features and checks of Solidity. This should only be used for tasks that need it and if there is confidence in using it.

Multiple vulnerabilities have been detected previously when the assembly is not properly used within the Solidity code; therefore, caution should be exercised while using them.

Issue Remediation

Avoid using inline assembly instructions if possible because it might introduce certain issues in the code if not dealt with properly because it bypasses several safety features that are already implemented in Solidity.

Bug ID

SSB_1700_45

Severity	Confidence	Line nos	Action Taken
Informational	Certain	**142-145**	False Positive

Bug Type

IN-LINE ASSEMBLY DETECTED

File Location

contract.sol

Issue Description

Inline assembly is a way to access the Ethereum Virtual Machine at a low level. This bypasses several important safety features and checks of Solidity. This should only be used for tasks that need it and if there is confidence in using it.

Multiple vulnerabilities have been detected previously when the assembly is not properly used within the Solidity code; therefore, caution should be exercised while using them.

Issue Remediation

Avoid using inline assembly instructions if possible because it might introduce certain issues in the code if not dealt with properly because it bypasses several safety features that are already implemented in Solidity.

Scan History

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

No	Date	Score	Scan Overview					
1.	2022-12-14	4.41	● 0	● 0	● 0	● 22	● 30	● 43

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.