CertiFi™

CRYPTO AGENCY

@certifiagency

Audit Report

May 5th, 2023

Defi Trading Club

www.certifiagency.com

Security Assessment

May 5, 2023

This security assessment report was prepared by CertiFi Agency, a USA-Based Crypto & Blockchain Agency.

Scan History

Critical High Medium Low Informational Gas

No	Date	Score	Scan Overview
1.	2023 05 03	4.35	3 4 4 25 45 47
2.	2023 05 13	4.65	0 0 4 25 45 47

Table of Contents.

- RETURN VALUE OF LOW LEVEL CALLS
- USE OF SAFEMATH LIBRARY
- UNNECESSARY DEFAULT VALUE INITIALIZATION
- FUNCTION SHOULD BE EXTERNAL
- UNUSED RECEIVE FALLBACK
- USING EXTCODESIZE TO CHECK FOR EXTERNALLY OWNED ACCOUNTS
- IN-LINE ASSEMBLY DETECTED
- VARIABLES SHOULD BE IMMUTABLE

Scan History

Disclaimer

Project Summary

This report has been prepared for Defi Trading Club (DTC) using CertiFi 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. CertiFi runs a comprehensive 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 CertiFi 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
Defi Trading Club

Contract Type
Smart Contract

Contract Address
0x7cB71D70FcAF9e2206916CBB0a18b33792C817bf

Contract Platform
bscscan

Contract Chain
mainnet

Contract URL
https://bscscan.com/address/0x7cB71D70FcAF9e2206916CBB0a18b33792C817bf

Language
Solidity

Website
www.defitradingclub.com

Date Published
May 13, 2023

Symbol
DTC

Project Contact Email
communitymanager@defitradingclub.com

Audit Methodology
Dynamic Scanning & Manual Review

Findings Summary

Defi Trading Club (DTC)

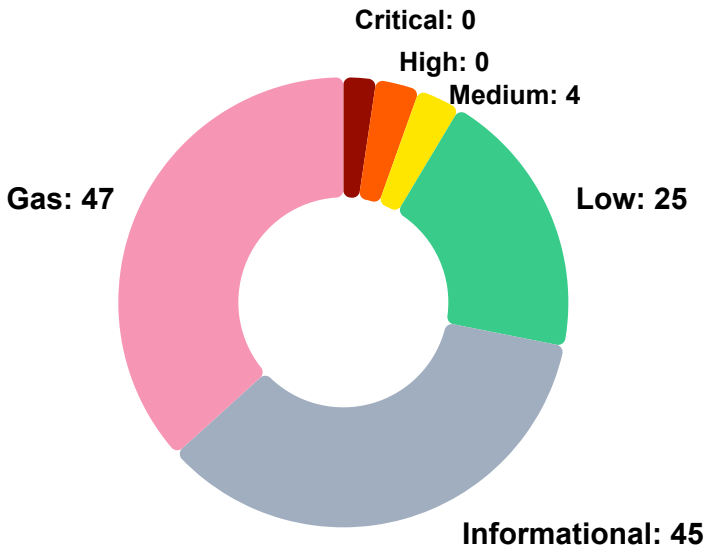
0x7cB71D70FcAF9e2206916CBB0a18b33792C817bf

Lines of Code **1420**

Score

4.65

Out of 5



Critical	0	Low	25
High	0	Informational	45
Medium	4	Gas	47

ACTION TAKEN

Fixed	False Positive
5	16
Won't Fix	**Pending Fix**
0	0

Bug ID	Severity	Bug Type	Status
SSB_43839_26	Medium	ACCOUNT EXISTENCE CHECK FOR LOW LEVEL CALLS	Pending Fix
SSB_43839_27	Medium	ACCOUNT EXISTENCE CHECK FOR LOW LEVEL CALLS	Pending Fix
SSB_43839_62	Informational	HARD-CODED ADDRESS DETECTED	Pending Fix
SSB_43839_63	Informational	HARD-CODED ADDRESS DETECTED	Pending Fix
SSB_43839_64	Informational	HARD-CODED ADDRESS DETECTED	Pending Fix
SSB_43839_58	Informational	BLOCK VALUES AS A PROXY FOR TIME	Pending Fix

SSB_43839_59	Informational	BLOCK VALUES AS A PROXY FOR TIME	Pending Fix
SSB_43839_60	Informational	BLOCK VALUES AS A PROXY FOR TIME	Pending Fix
SSB_43839_61	Informational	BLOCK VALUES AS A PROXY FOR TIME	Pending Fix
SSB_43839_61	Informational	BLOCK VALUES AS A PROXY FOR TIME	Pending Fix
SSB_43839_33	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSB_43839_34	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSB_43839_35	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSB_43839_36	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSB_43839_37	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSB_43839_38	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSB_43839_39	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSB_43839_40	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix

SSB_43839_41	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSB_43839_42	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSB_43839_43	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_43839_44	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_43839_45	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_43839_46	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_43839_47	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_43839_48	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_43839_49	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_43839_50	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_43839_51	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_43839_52	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix

SSB_43839_53	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_43839_54	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_43839_55	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_43839_31	Low	COMPILER VERSION TOO RECENT	Pending Fix
SSB_43839_30	Gas	CUSTOM ERRORS TO SAVE GAS	Pending Fix
SSB_43839_98	High	APPROVE FRONT-RUNNING ATTACK	FIXED
SSB_43839_99	High	APPROVE FRONT-RUNNING ATTACK	FIXED
SSB_43839_100	High	APPROVE FRONT-RUNNING ATTACK	FIXED
SSB_43839_101	High	APPROVE FRONT-RUNNING ATTACK	FIXED
SSB_43839_5	Gas	EXTRA GAS USAGE IN EMIT WITH LONG STRINGS	Pending Fix
SSB_43839_6	Gas	EXTRA GAS USAGE IN EMIT WITH LONG STRINGS	Pending Fix
SSB_43839_7	Gas	EXTRA GAS USAGE IN EMIT WITH LONG STRINGS	Pending Fix

SSB_43839_8	Gas	EXTRA GAS USAGE IN EMIT WITH LONG STRINGS	Pending Fix
SSB_43839_32	Gas	FUNCTION SHOULD RETURN STRUCT	Pending Fix
SSB_43839_21	Gas	INTERNAL FUNCTIONS NEVER USED	Pending Fix
SSB_43839_22	Gas	INTERNAL FUNCTIONS NEVER USED	Pending Fix
SSB_43839_23	Gas	INTERNAL FUNCTIONS NEVER USED	Pending Fix
SSB_43839_24	Gas	INTERNAL FUNCTIONS NEVER USED	Pending Fix
SSB_43839_65	Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_43839_66	Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_43839_67	Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_43839_68	Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_43839_74	Low	MISSING EVENTS	Pending Fix
SSB_43839_75	Low	MISSING EVENTS	Pending Fix

SSB_43839_76	Low	MISSING EVENTS	Pending Fix
SSB_43839_77	Low	MISSING EVENTS	Pending Fix
SSB_43839_78	Low	MISSING EVENTS	Pending Fix
SSB_43839_79	Low	MISSING EVENTS	Pending Fix
SSB_43839_80	Low	MISSING EVENTS	Pending Fix
SSB_43839_81	Low	MISSING EVENTS	Pending Fix
SSB_43839_82	Low	MISSING EVENTS	Pending Fix
SSB_43839_83	Low	MISSING EVENTS	Pending Fix
SSB_43839_84	Low	MISSING EVENTS	Pending Fix
SSB_43839_85	Low	MISSING EVENTS	Pending Fix
SSB_43839_86	Low	MISSING EVENTS	Pending Fix
SSB_43839_87	Low	MISSING EVENTS	Pending Fix

SSB_43839_88	Low	MISSING EVENTS	Pending Fix
SSB_43839_89	Low	MISSING EVENTS	Pending Fix
SSB_43839_90	Low	MISSING EVENTS	Pending Fix
SSB_43839_91	Low	MISSING EVENTS	Pending Fix
SSB_43839_92	Low	MISSING EVENTS	Pending Fix
SSB_43839_93	Low	MISSING EVENTS	Pending Fix
SSB_43839_94	Low	MISSING EVENTS	Pending Fix
SSB_43839_95	Low	MISSING EVENTS	Pending Fix
SSB_43839_96	Low	MISSING EVENTS	Pending Fix
SSB_43839_97	Low	MISSING EVENTS	Pending Fix
SSB_43839_69	Informational	MISSING INDEXED KEYWORDS IN EVENTS	Pending Fix
SSB_43839_70	Informational	MISSING INDEXED KEYWORDS IN EVENTS	Pending Fix

SSB_43839_71	Informational	MISSING INDEXED KEYWORDS IN EVENTS	Pending Fix
SSB_43839_72	Informational	MISSING INDEXED KEYWORDS IN EVENTS	Pending Fix
SSB_43839_73	Informational	MISSING INDEXED KEYWORDS IN EVENTS	Pending Fix
SSB_43839_2	Critical	MISSING PAYABLE IN CALL FUNCTION	FIXED
SSB_43839_3	Critical	MISSING PAYABLE IN CALL FUNCTION	FIXED
SSB_43839_4	Critical	MISSING PAYABLE IN CALL FUNCTION	FIXED
SSB_43839_105	Informational	PRESENCE OF OVERPOWERED ROLE	Pending Fix
SSB_43839_106	Informational	PRESENCE OF OVERPOWERED ROLE	Pending Fix
SSB_43839_107	Informational	PRESENCE OF OVERPOWERED ROLE	Pending Fix
SSB_43839_108	Informational	PRESENCE OF OVERPOWERED ROLE	Pending Fix
SSB_43839_109	Informational	PRESENCE OF OVERPOWERED ROLE	Pending Fix
SSB_43839_110	Informational	PRESENCE OF OVERPOWERED ROLE	Pending Fix

SSB_43839_111	Informational	PRESENCE OF OVERPOWERED ROLE	Pending Fix
SSB_43839_112	Informational	PRESENCE OF OVERPOWERED ROLE	Pending Fix
SSB_43839_113	Informational	PRESENCE OF OVERPOWERED ROLE	Pending Fix
SSB_43839_114	Informational	PRESENCE OF OVERPOWERED ROLE	Pending Fix
SSB_43839_115	Informational	PRESENCE OF OVERPOWERED ROLE	Pending Fix
SSB_43839_116	Informational	PRESENCE OF OVERPOWERED ROLE	Pending Fix
SSB_43839_117	Informational	PRESENCE OF OVERPOWERED ROLE	Pending Fix
SSB_43839_118	Informational	PRESENCE OF OVERPOWERED ROLE	Pending Fix
SSB_43839_119	Informational	PRESENCE OF OVERPOWERED ROLE	Pending Fix
SSB_43839_120	Informational	PRESENCE OF OVERPOWERED ROLE	Pending Fix
SSB_43839_121	Informational	PRESENCE OF OVERPOWERED ROLE	Pending Fix
SSB_43839_122	Informational	PRESENCE OF OVERPOWERED ROLE	Pending Fix

SSB_43839_123	Informational	PRESENCE OF OVERPOWERED ROLE	Pending Fix
SSB_43839_124	Informational	PRESENCE OF OVERPOWERED ROLE	Pending Fix
SSB_43839_125	Informational	PRESENCE OF OVERPOWERED ROLE	Pending Fix
SSB_43839_126	Informational	PRESENCE OF OVERPOWERED ROLE	Pending Fix
SSB_43839_127	Informational	PRESENCE OF OVERPOWERED ROLE	Pending Fix
SSB_43839_29	Informational	REQUIRE WITH EMPTY MESSAGE	Pending Fix
SSB_43839_25	Medium	RETURN VALUE OF LOW LEVEL CALLS	Pending Fix
SSB_43839_1	Gas	USE OF SAFEMATH LIBRARY	Pending Fix
SSB_43839_102	Gas	UNNECESSARY DEFAULT VALUE INITIALIZATION	Pending Fix
SSB_43839_103	Gas	UNNECESSARY DEFAULT VALUE INITIALIZATION	Pending Fix
SSB_43839_14	Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSB_43839_15	Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix

SSB_43839_16	Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSB_43839_17	Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSB_43839_18	Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSB_43839_19	Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSB_43839_20	Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSB_43839_104	Informational	UNUSED RECEIVE FALLBACK	Pending Fix
SSB_43839_28	Medium	USING EXTCODESIZE TO CHECK FOR EXTERNALLY OWNED ACCOUNTS	Pending Fix
SSB_43839_56	Informational	IN-LINE ASSEMBLY DETECTED	Pending Fix
SSB_43839_57	Informational	IN-LINE ASSEMBLY DETECTED	Pending Fix
SSB_43839_9	Informational	VARIABLES SHOULD BE IMMUTABLE	Pending Fix
SSB_43839_10	Informational	VARIABLES SHOULD BE IMMUTABLE	Pending Fix
SSB_43839_11	Informational	VARIABLES SHOULD BE IMMUTABLE	Pending Fix

SSB_43839_12	Informational	VARIABLES SHOULD BE IMMUTABLE	Pending Fix
SSB_43839_13	Informational	VARIABLES SHOULD BE IMMUTABLE	Pending Fix

Vulnerability Details

Bug ID

SSB_43839_26

Severity

Medium

Confidence

Tentative

Line nos

60-60

Action Taken

Pending Fix

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. The remediation also largely depends on the contract logic since bypasses are possible during constructor calls

Bug ID

SSB_43839_27

Severity

Medium

Confidence

Tentative

Line nos

1068-1068

Action Taken

Pending Fix

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. The remediation also largely depends on the contract logic since bypasses are possible during constructor calls

Bug ID

SSB_43839_62

Severity

Informational

Confidence

Tentative

Line nos

336-336

Action Taken

Pending Fix

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 -
`0xde491C65E507d281B6a3688d11e8fC222eee0975`

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_43839_63

Severity

Informational

Confidence

Tentative

Line nos

337-337

Action Taken

Pending Fix

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_43839_64

Severity

Informational

Confidence

Tentative

Line nos

407-407

Action Taken

Pending Fix

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_43839_58

Severity

Informational

Confidence

Firm

Line nos

687-687

Action Taken

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

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices

can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Bug ID

SSB_43839_59

Severity

Informational

Confidence

Firm

Line nos

1120-1120

Action Taken

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

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices

can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Bug ID

SSB_43839_60

Severity

Informational

Confidence

Firm

Line nos

1145-1145

Action Taken

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

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices

can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Bug ID

SSB_43839_61

Severity

Informational

Confidence

Firm

Line nos

1287-1287

Action Taken

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

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices

can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Bug ID

SSB_43839_61

Severity

Informational

Confidence

Firm

Line nos

1301-1301

Action Taken

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

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices

can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Bug ID

SSB_43839_33

Severity

Gas

Confidence

Firm

Line nos

513-513

Action Taken

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_43839_34

Severity

Gas

Confidence

Firm

Line nos

713-713

Action Taken

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_43839_35

Severity

Gas

Confidence

Firm

Line nos

719-719

Action Taken

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_43839_36

Severity

Gas

Confidence

Firm

Line nos

923-923

Action Taken

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_43839_37

Severity

Gas

Confidence

Firm

Line nos

1247-1247

Action Taken

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_43839_38

Severity

Gas

Confidence

Firm

Line nos

1253-1253

Action Taken

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_43839_39

Severity

Gas

Confidence

Firm

Line nos

1262-1262

Action Taken

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_43839_40

Severity

Gas

Confidence

Firm

Line nos

1369-1369

Action Taken

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_43839_41

Severity

Gas

Confidence

Firm

Line nos

1376-1376

Action Taken

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_43839_42

Severity

Gas

Confidence

Firm

Line nos

1386-1386

Action Taken

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_43839_43

Severity

Gas

Confidence

Firm

Line nos

32-32

Action Taken

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_43839_44

Severity

Gas

Confidence

Firm

Line nos

59-59

Action Taken

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_43839_45

Severity

Gas

Confidence

Firm

Line nos

77-77

Action Taken

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_43839_46

Severity

Gas

Confidence

Firm

Line nos

557-561

Action Taken

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_43839_47

Severity

Gas

Confidence

Firm

Line nos

586-590

Action Taken

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_43839_48

Severity

Gas

Confidence

Firm

Line nos

628-628

Action Taken

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_43839_49

Severity

Gas

Confidence

Firm

Line nos

629-629

Action Taken

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_43839_50

Severity

Gas

Confidence

Firm

Line nos

635-635

Action Taken

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_43839_51

Severity

Gas

Confidence

Firm

Line nos

636-636

Action Taken

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_43839_52

Severity

Gas

Confidence

Firm

Line nos

673-673

Action Taken

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_43839_53

Severity

Gas

Confidence

Firm

Line nos

1089-1089

Action Taken

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_43839_54

Severity

Gas

Confidence

Firm

Line nos

1165-1165

Action Taken

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_43839_55

Severity

Gas

Confidence

Firm

Line nos

1172-1172

Action Taken

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_43839_31

Severity

Low

Confidence

Certain

Line nos

6-6

Action Taken

Pending Fix

Bug Type

COMPILER VERSION TOO RECENT

File Location

contract.sol

Issue Description

The compiler version detected in the code is too recent. Therefore, it is not time-tested and may be susceptible to multiple bugs and vulnerabilities, both from the usage and security perspectives. The following compiler versions were detected which were too recent - ['contract.sol'] - 0.8.19

Issue Remediation

It is suggested to use a compiler version that is neither too recent nor too old i.e., Solidity `0.8.18`. A stable compiler version should be used that is time-tested by the community, which fixed vulnerabilities introduced in older compiler versions.

The code should be kept updated according to the compiler release cycle. It should be tested before going on the Mainnet to reduce the chances of new vulnerabilities being introduced.

Bug ID

SSB_43839_30

Severity

Gas

Confidence

Certain

Line nos

114-114

Action Taken

Pending Fix

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_43839_98

Severity

High

Confidence

Tentative

Line nos

1044-1047

Action Taken

FIXED

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_43839_99

Severity

High

Confidence

Tentative

Line nos

1061-1065

Action Taken

FIXED

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_43839_100

Severity

High

Confidence

Tentative

Line nos

1276-1289

Action Taken

FIXED

Bug Type

APPROVE FRONT-RUNNING ATTACK

File Location

contract.sol

Issue Description

The `swapTokensForBNB()` 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 swapTokensForBNB 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_43839_101

Severity

High

Confidence

Tentative

Line nos

1292-1303

Action Taken

FIXED

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_43839_5

Severity

Gas

Confidence

Firm

Line nos

859-877

Action Taken

Pending Fix

Bug Type

EXTRA GAS USAGE IN EMIT WITH LONG STRINGS

File Location

contract.sol

Issue Description

The only limits to how long a string argument to a function call can be is the block gas limit of the EVM, currently 30 million. If the function call arguments gets passed directly inside any emitted events, it will also affect the gas cost and refund. Gas refunds will include the gas price of emitting this event, which could potentially be very large.
The contract was passing parameter ['newOwner'] inside event OwnershipTransferred.

Issue Remediation

It is recommended to not pass user-controlled parameters directly inside emitted events. If it’s absolutely necessary, consider having input validations on the parameter.

Bug ID

SSB_43839_6

Severity

Gas

Confidence

Firm

Line nos

913-916

Action Taken

Pending Fix

Bug Type

EXTRA GAS USAGE IN EMIT WITH LONG STRINGS

File Location

contract.sol

Issue Description

The only limits to how long a string argument to a function call can be is the block gas limit of the EVM, currently 30 million. If the function call arguments gets passed directly inside any emitted events, it will also affect the gas cost and refund. Gas refunds will include the gas price of emitting this event, which could potentially be very large.
The contract was passing parameter ['true_or_false'] inside event updated_SwapAndLiquify_Enabled.

Issue Remediation

It is recommended to not pass user-controlled parameters directly inside emitted events. If it's absolutely necessary, consider having input validations on the parameter.

Bug ID

SSB_43839_7

Severity

Gas

Confidence

Firm

Line nos

1049-1054

Action Taken

Pending Fix

Bug Type

EXTRA GAS USAGE IN EMIT WITH LONG STRINGS

File Location

contract.sol

Issue Description

The only limits to how long a string argument to a function call can be is the block gas limit of the EVM, currently 30 million. If the function call arguments gets passed directly inside any emitted events, it will also affect the gas cost and refund. Gas refunds will include the gas price of emitting this event, which could potentially be very large.
The contract was passing parameter ['owner', 'spender', 'amount'] inside event Approval.

Issue Remediation

It is recommended to not pass user-controlled parameters directly inside emitted events. If it’s absolutely necessary, consider having input validations on the parameter.

Bug ID

SSB_43839_8

Severity

Gas

Confidence

Firm

Line nos

1319-1401

Action Taken

Pending Fix

Bug Type

EXTRA GAS USAGE IN EMIT WITH LONG STRINGS

File Location

contract.sol

Issue Description

The only limits to how long a string argument to a function call can be is the block gas limit of the EVM, currently 30 million. If the function call arguments gets passed directly inside any emitted events, it will also affect the gas cost and refund. Gas refunds will include the gas price of emitting this event, which could potentially be very large.
The contract was passing parameter ['sender', 'recipient'] inside event Transfer.

Issue Remediation

It is recommended to not pass user-controlled parameters directly inside emitted events. If it’s absolutely necessary, consider having input validations on the parameter.

Bug ID

SSB_43839_32

Severity

Gas

Confidence

Tentative

Line nos

485-531

Action Taken

Pending Fix

Bug Type

FUNCTION SHOULD RETURN STRUCT

File Location

contract.sol

Issue Description

The function Project_Information 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_43839_21

Severity

Gas

Confidence

Tentative

Line nos

27-28

Action Taken

Pending Fix

Bug Type

INTERNAL FUNCTIONS NEVER USED

File Location

contract.sol

Issue Description

The contract declared internal functions but was not using them in any of the functions or contracts.
Since internal functions can only be called from inside the contracts, it makes no sense to have them if they are not used. This uses up gas and causes issues for auditors when understanding the contract logic.

Issue Remediation

Having dead code in the contracts uses up unnecessary gas and increases the complexity of the overall smart contract.
It is recommended to remove the internal functions from the contracts if they are never used.

Bug ID

SSB_43839_22

Severity

Gas

Confidence

Tentative

Line nos

34-36

Action Taken

Pending Fix

Bug Type

INTERNAL FUNCTIONS NEVER USED

File Location

contract.sol

Issue Description

The contract declared internal functions but was not using them in any of the functions or contracts.
Since internal functions can only be called from inside the contracts, it makes no sense to have them if they are not used. This uses up gas and causes issues for auditors when understanding the contract logic.

Issue Remediation

Having dead code in the contracts uses up unnecessary gas and increases the complexity of the overall smart contract.
It is recommended to remove the internal functions from the contracts if they are never used.

Bug ID

SSB_43839_23

Severity

Gas

Confidence

Tentative

Line nos

43-46

Action Taken

Pending Fix

Bug Type

INTERNAL FUNCTIONS NEVER USED

File Location

contract.sol

Issue Description

The contract declared internal functions but was not using them in any of the functions or contracts.
Since internal functions can only be called from inside the contracts, it makes no sense to have them if they are not used. This uses up gas and causes issues for auditors when understanding the contract logic.

Issue Remediation

Having dead code in the contracts uses up unnecessary gas and increases the complexity of the overall smart contract.
It is recommended to remove the internal functions from the contracts if they are never used.

Bug ID

SSB_43839_24

Severity

Gas

Confidence

Tentative

Line nos

58-62

Action Taken

Pending Fix

Bug Type

INTERNAL FUNCTIONS NEVER USED

File Location

contract.sol

Issue Description

The contract declared internal functions but was not using them in any of the functions or contracts.
Since internal functions can only be called from inside the contracts, it makes no sense to have them if they are not used. This uses up gas and causes issues for auditors when understanding the contract logic.

Issue Remediation

Having dead code in the contracts uses up unnecessary gas and increases the complexity of the overall smart contract.
It is recommended to remove the internal functions from the contracts if they are never used.

Bug ID

SSB_43839_65

Severity

Gas

Confidence

Certain

Line nos

61-61

Action Taken

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_43839_66

Severity

Gas

Confidence

Certain

Line nos

1050-1050

Action Taken

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_43839_67

Severity

Gas

Confidence

Certain

Line nos

1051-1051

Action Taken

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_43839_68

Severity

Gas

Confidence

Certain

Line nos

1109-1109

Action Taken

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_43839_74

Severity

Low

Confidence

Firm

Line nos

58-62

Action Taken

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_43839_75

Severity

Low

Confidence

Firm

Line nos

64-66

Action Taken

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_43839_76

Severity

Low

Confidence

Firm

Line nos

68-70

Action Taken

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_43839_77

Severity

Low

Confidence

Firm

Line nos

72-74

Action Taken

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_43839_78

Severity

Low

Confidence

Firm

Line nos

76-81

Action Taken

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_43839_79

Severity

Low

Confidence

Firm

Line nos

94-96

Action Taken

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 functionDelegateCall 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_43839_80

Severity

Low

Confidence

Firm

Line nos

98-102

Action Taken

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 functionDelegateCall 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_43839_81

Severity

Low

Confidence

Firm

Line nos

534-544

Action Taken

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 STANDARD_TOKEN was found to be missing these events on the function Contract_SetUp_01__Presale_Address 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_43839_82

Severity

Low

Confidence

Firm

Line nos

666-678

Action Taken

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 STANDARD_TOKEN was found to be missing these events on the function Contract_SetUp_05__Bot_Protection 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_43839_83

Severity

Low

Confidence

Firm

Line nos

681-691

Action Taken

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 STANDARD_TOKEN was found to be missing these events on the function Contract_SetUp_06__Open_Trade 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_43839_84

Severity

Low

Confidence

Firm

Line nos

694-703

Action Taken

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 STANDARD_TOKEN was found to be missing these events on the function Contract_SetUp_07__Blacklist_Bots 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_43839_85

Severity

Low

Confidence

Firm

Line nos

706-724

Action Taken

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 STANDARD_TOKEN was found to be missing these events on the function Contract_SetUp_08__Deactivate_Launch_Mode 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_43839_86

Severity

Low

Confidence

Firm

Line nos

734-755

Action Taken

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 STANDARD_TOKEN was found to be missing these events on the function Contract_SetUp_09__Update_Wallets 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_43839_87

Severity

Low

Confidence

Firm

Line nos

765-777

Action Taken

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 STANDARD_TOKEN was found to be missing these events on the function Contract_SetUp_10__Update_Links 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_43839_88

Severity

Low

Confidence

Firm

Line nos

795-799

Action Taken

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 STANDARD_TOKEN was found to be missing these events on the function Contract__Options__Burn_From_Supply 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_43839_89

Severity

Low

Confidence

Firm

Line nos

816-820

Action Taken

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 STANDARD_TOKEN was found to be missing these events on the function Contract__Options__Free_Wallet_Transfers 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_43839_90

Severity

Low

Confidence

Firm

Line nos

823-832

Action Taken

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 STANDARD_TOKEN was found to be missing these events on the function Maintenance__Add_Liquidity_Pair 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_43839_91

Severity

Low

Confidence

Firm

Line nos

889-902

Action Taken

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 STANDARD_TOKEN was found to be missing these events on the function Maintenance__Remove_Contract_Fee 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_43839_92

Severity

Low

Confidence

Firm

Line nos

931-934

Action Taken

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 STANDARD_TOKEN was found to be missing these events on the function Processing__Swap_Trigger_Count 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_43839_93

Severity

Low

Confidence

Firm

Line nos

958-966

Action Taken

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 STANDARD_TOKEN was found to be missing these events on the function Wallet_Settings__Exclude_From_Fees 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_43839_94

Severity

Low

Confidence

Firm

Line nos

969-976

Action Taken

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 STANDARD_TOKEN was found to be missing these events on the function Wallet_Settings__Exempt_From_Limits 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_43839_95

Severity

Low

Confidence

Firm

Line nos

979-986

Action Taken

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 STANDARD_TOKEN was found to be missing these events on the function Wallet_Settings__Pre_Launch_Access 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_43839_96

Severity

Low

Confidence

Firm

Line nos

989-995

Action Taken

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 STANDARD_TOKEN was found to be missing these events on the function Wallet_Settings__Remove_Early_Buyer_Tag 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_43839_97

Severity

Low

Confidence

Firm

Line nos

1067-1069

Action Taken

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 STANDARD_TOKEN was found to be missing these events on the function send_BNB 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_43839_69

Severity

Informational

Confidence

Certain

Line nos

157-157

Action Taken

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_43839_70

Severity

Informational

Confidence

Certain

Line nos

441-441

Action Taken

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_43839_71

Severity

Informational

Confidence

Certain

Line nos

442-442

Action Taken

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_43839_72

Severity

Informational

Confidence

Certain

Line nos

443-443

Action Taken

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_43839_73

Severity

Informational

Confidence

Certain

Line nos

445-445

Action Taken

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_43839_2

Severity

Critical

Confidence

Firm

Line nos

60-60

Action Taken

FIXED

Bug Type

MISSING PAYABLE IN CALL FUNCTION

File Location

contract.sol

Issue Description

The contract is using a `.call()` method to make external calls along with passing some Ether as `msg.value`. Since the function sendValue is not marked as `payable`, the transaction will fail.

Issue Remediation

If the function needs to pass some Ether as `msg.value` inside a function, make sure to set that function as `payable`. No changes are required if the use case is to send Ether from the contract's balance.

Bug ID

SSB_43839_3

Severity

Critical

Confidence

Firm

Line nos

79-79

Action Taken

FIXED

Bug Type

MISSING PAYABLE IN CALL FUNCTION

File Location

contract.sol

Issue Description

The contract is using a `.call()` method to make external calls along with passing some Ether as `msg.value`. Since the function functionCallWithValue is not marked as `payable`, the transaction will fail.

Issue Remediation

If the function needs to pass some Ether as `msg.value` inside a function, make sure to set that function as `payable`. No changes are required if the use case is to send Ether from the contract's balance.

Bug ID

SSB_43839_4

Severity

Critical

Confidence

Firm

Line nos

1068-1068

Action Taken

FIXED

Bug Type

MISSING PAYABLE IN CALL FUNCTION

File Location

contract.sol

Issue Description

The contract is using a `.call()` method to make external calls along with passing some Ether as `msg.value`. Since the function send_BNB is not marked as `payable`, the transaction will fail.

Issue Remediation

If the function needs to pass some Ether as `msg.value` inside a function, make sure to set that function as `payable`. No changes are required if the use case is to send Ether from the contract's balance.

Bug ID

SSB_43839_105

Severity

Informational

Confidence

Firm

Line nos

534-544

Action Taken

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_43839_106

Severity

Informational

Confidence

Firm

Line nos

547-573

Action Taken

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_43839_107

Severity

Informational

Confidence

Firm

Line nos

576-603

Action Taken

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_43839_108

Severity

Informational

Confidence

Firm

Line nos

618-646

Action Taken

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_43839_109

Severity

Informational

Confidence

Firm

Line nos

666-678

Action Taken

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_43839_110

Severity

Informational

Confidence

Firm

Line nos

681-691

Action Taken

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_43839_111

Severity

Informational

Confidence

Firm

Line nos

694-703

Action Taken

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_43839_112

Severity

Informational

Confidence

Firm

Line nos

706-724

Action Taken

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_43839_113

Severity

Informational

Confidence

Firm

Line nos

734-755

Action Taken

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_43839_114

Severity

Informational

Confidence

Firm

Line nos

765-777

Action Taken

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_43839_115

Severity

Informational

Confidence

Firm

Line nos

795-799

Action Taken

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_43839_116

Severity

Informational

Confidence

Firm

Line nos

816-820

Action Taken

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_43839_117

Severity

Informational

Confidence

Firm

Line nos

823-832

Action Taken

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_43839_118

Severity

Informational

Confidence

Firm

Line nos

845-856

Action Taken

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_43839_119

Severity

Informational

Confidence

Firm

Line nos

859-877

Action Taken

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_43839_120

Severity

Informational

Confidence

Firm

Line nos

913-916

Action Taken

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_43839_121

Severity

Informational

Confidence

Firm

Line nos

919-928

Action Taken

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_43839_122

Severity

Informational

Confidence

Firm

Line nos

931-934

Action Taken

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_43839_123

Severity

Informational

Confidence

Firm

Line nos

937-947

Action Taken

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_43839_124

Severity

Informational

Confidence

Firm

Line nos

958-966

Action Taken

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_43839_125

Severity

Informational

Confidence

Firm

Line nos

969-976

Action Taken

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_43839_126

Severity

Informational

Confidence

Firm

Line nos

979-986

Action Taken

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_43839_127

Severity

Informational

Confidence

Firm

Line nos

989-995

Action Taken

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_43839_29

Severity

Informational

Confidence

Certain

Line nos

684-684

Action Taken

Pending Fix

Bug Type

REQUIRE WITH EMPTY MESSAGE

File Location

contract.sol

Issue Description

A `require` statement was detected with an empty message. It takes two parameters and the message part is optional. This is shown to the user when and if the `require` statement evaluates to `false`. This message gives more information about the statement and why it gave a `false` response.

Issue Remediation

It is recommended to add a descriptive message, no longer than `32 bytes`, inside the `require` statement to give more detail to the user about why the condition failed.

Bug ID

SSB_43839_25

Severity

Medium

Confidence

Certain

Line nos

1068-1068

Action Taken

Pending Fix

Bug Type

RETURN VALUE OF LOW LEVEL CALLS

File Location

contract.sol

Issue Description

The functions do not check the return value of low-level calls. This can lock Ether in the contract if the call fails or may compromise the contract if the ownership is being changed. The following calls were detected without return value validations - call

Issue Remediation

Ensure return value is checked using conditional statements for low-level calls. We should also ensure that we log failed calls using events.

Bug ID

SSB_43839_1

Severity

Gas

Confidence

Certain

Line nos

328-328

Action Taken

Pending Fix

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_43839_102

Severity

Gas

Confidence

Certain

Line nos

478-478

Action Taken

Pending Fix

Bug Type

UNNECESSARY DEFAULT VALUE INITIALIZATION

File Location

contract.sol

Issue Description

The contract was found to be initializing the value of bool to it's default value, i.e., false. This is redundant and not required.

Issue Remediation

It's not recommended to initialize the data types to their default values unless there's a use-case because it's unnecessary and costs around ~3 gas.

Bug ID

SSB_43839_103

Severity

Gas

Confidence

Certain

Line nos

479-479

Action Taken

Pending Fix

Bug Type

UNNECESSARY DEFAULT VALUE INITIALIZATION

File Location

contract.sol

Issue Description

The contract was found to be initializing the value of bool to it's default value, i.e., false. This is redundant and not required.

Issue Remediation

It's not recommended to initialize the data types to their default values unless there's a use-case because it's unnecessary and costs around ~3 gas.

Bug ID

SSB_43839_14

Severity

Gas

Confidence

Certain

Line nos

1061-1065

Action Taken

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_43839_15

Severity

Gas

Confidence

Certain

Line nos

1039-1042

Action Taken

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_43839_16

Severity

Gas

Confidence

Certain

Line nos

859-877

Action Taken

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_43839_17

Severity

Gas

Confidence

Certain

Line nos

1044-1047

Action Taken

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_43839_18

Severity

Gas

Confidence

Certain

Line nos

1034-1037

Action Taken

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_43839_19

Severity

Gas

Confidence

Certain

Line nos

816-820

Action Taken

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_43839_20

Severity

Gas

Confidence

Certain

Line nos

845-856

Action Taken

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_43839_104

Severity

Informational

Confidence

Tentative

Line nos

1404-1404

Action Taken

Pending Fix

Bug Type

UNUSED RECEIVE FALLBACK

File Location

contract.sol

Issue Description

NOT DEFINED YET

Issue Remediation

NOT DEFINED YET

Bug ID

SSB_43839_28

Severity

Medium

Confidence

Tentative

Line nos

54-54

Action Taken

Pending Fix

Bug Type

USING EXTCODESIZE TO CHECK FOR EXTERNALLY OWNED ACCOUNTS

File Location

contract.sol

Issue Description

`extcodesize` is used to check if a contract is an externally owned account or another contract.
`extcodesize` returns `0` for externally owned accounts but there's a specific condition here that when an `extcodesize` check is made to a contract that is still under construction or when the contract's constructor is running, extcodesize for its address returns zero.
This may give erroneous outputs for checking externally owned contracts.

Issue Remediation

It is recommended to manually check and validate at compile-time that the contract/account address being checked inside `extcodesize` does not return improper values due to the external contract's construction.

Bug ID

SSB_43839_56

Severity

Informational

Confidence

Certain

Line nos

54-54

Action Taken

Pending Fix

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_43839_57

Severity

Informational

Confidence

Certain

Line nos

109-112

Action Taken

Pending Fix

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_43839_9

Severity

Informational

Confidence

Tentative

Line nos

385-385

Action Taken

Pending Fix

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

contract.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSB_43839_10

Severity

Informational

Confidence

Tentative

Line nos

386-386

Action Taken

Pending Fix

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

contract.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSB_43839_11

Severity

Informational

Confidence

Tentative

Line nos

387-387

Action Taken

Pending Fix

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

contract.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSB_43839_12

Severity

Informational

Confidence

Tentative

Line nos

410-410

Action Taken

Pending Fix

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

contract.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSB_43839_13

Severity

Informational

Confidence

Tentative

Line nos

411-411

Action Taken

Pending Fix

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

contract.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Scan History

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

No	Date	Score	Scan Overview
1.	2023 05 03	4.35	● 3 ● 4 ● 4 ● 25 ● 45 ● 47
2.	2023 05 13	4.65	● 0 ● 0 ● 4 ● 25 ● 45 ● 47

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, 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 CertiFi 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. CertiFi 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.