SolidityScan | Powered by CRED SHIELDS

Security Assessment

8 Sept 2023

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

Table of Contents.

- NAME MAPPING PARAMETERS
- OPTIMIZING ADDRESS ID MAPPING
- PUBLIC CONSTANTS CAN BE PRIVATE
- REQUIRE WITH EMPTY MESSAGE
- USE OF SAFEMATH LIBRARY
- STORAGE VARIABLE CACHING IN MEMORY
- UNNECESSARY CHECKED ARITHMETIC IN LOOP
- FUNCTION SHOULD BE EXTERNAL
- USE OWNABLE2STEP
- VARIABLES DECLARED BUT NEVER USED

Scan History

Disclaimer

Project Summary

This report has been prepared for using SolidityScan to scan and discover vulnerabilities and safe coding practices in their smart contract including the libraries used by the contract that are not officially recognized. The SolidityScan tool runs a comprehensive static analysis on the Solidity code and finds vulnerabilities ranging from minor gas optimizations to major vulnerabilities leading to the loss of funds. The coverage scope pays attention to all the informational and critical vulnerabilities with over (130+) modules. The scanning and auditing process covers the following areas:

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

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

Audit Summary

Contract Name
EXZO

Contract Type
Smart Contract

Contract Address
0xb48344E930106cED3413BA69669CaD8b6DD4e1A5

Contract Platform
etherscan

Contract Chain
mainnet

Contract URL
https://etherscan.io/address/0xb48344E930106cED3413BA69669CaD8b6DD4e1A5

Language
Solidity

Date Published
08 Sep 2023

Audit Methodology

Static Scanning

Findings Summary

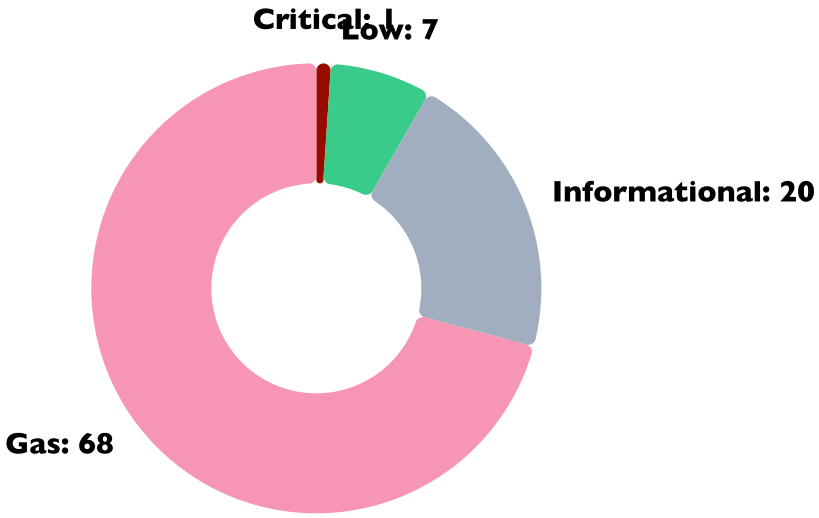
EXZO

0xb48344E930106cED3413BA69669CaD8b6DD4e1A5

Lines of Code **1046**

89.77

Security Score



Critical	1	Low	7
High	0	Informational	20
Medium	0	Gas	68

ACTION TAKEN

Fixed	False Positive
0	0
Won't Fix	**Pending Fix**
0	93

Bug ID	Severity	Bug Type	Status
SSB_57455_38	Informational	HARD-CODED ADDRESS DETECTED	Pending Fix
SSB_57455_39	Informational	HARD-CODED ADDRESS DETECTED	Pending Fix
SSB_57455_54	Gas	ARRAY LENGTH CACHING	Pending Fix
SSB_57455_35	Informational	BOOLEAN EQUALITY	Pending Fix
SSB_57455_36	Informational	BOOLEAN EQUALITY	Pending Fix
SSB_57455_57	Gas	CHEAPER CONDITIONAL OPERATORS	Pending Fix

SSB_57455_58	Gas	CHEAPER CONDITIONAL OPERATORS	Pending Fix
SSB_57455_59	Gas	CHEAPER CONDITIONAL OPERATORS	Pending Fix
SSB_57455_31	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSB_57455_32	Gas	CHEAPER INEQUALITIES IN IF()	Pending Fix
SSB_57455_42	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_57455_43	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_57455_44	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_57455_45	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_57455_46	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_57455_47	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_57455_48	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_57455_49	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix

SSB_57455_50	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_57455_51	Gas	CHEAPER INEQUALITIES IN REQUIRE()	Pending Fix
SSB_57455_4	Gas	DEFINE CONSTRUCTOR AS PAYABLE	Pending Fix
SSB_57455_5	Gas	DEFINE CONSTRUCTOR AS PAYABLE	Pending Fix
SSB_57455_26	Low	USE OF FLOATING PRAGMA	Pending Fix
SSB_57455_52	Gas	GAS OPTIMIZATION FOR STATE VARIABLES	Pending Fix
SSB_57455_53	Gas	GAS OPTIMIZATION FOR STATE VARIABLES	Pending Fix
SSB_57455_13	Gas	GAS OPTIMIZATION IN INCREMENTS	Pending Fix
SSB_57455_34	Critical	INCORRECT ACCESS CONTROL	Pending Fix
SSB_57455_6	Gas	INTERNAL FUNCTIONS NEVER USED	Pending Fix
SSB_57455_60	Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_57455_61	Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix

SSB_57455_62	Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_57455_63	Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_57455_64	Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_57455_65	Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_57455_66	Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_57455_67	Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_57455_68	Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_57455_69	Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_57455_70	Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_57455_71	Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_57455_72	Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_57455_73	Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix

SSB_57455_74	Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_57455_75	Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_57455_76	Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_57455_77	Gas	LONG REQUIRE/REVERT STRINGS	Pending Fix
SSB_57455_2	Low	MISCONFIGURED BEFORETOKENTRANSFER	Pending Fix
SSB_57455_3	Low	MISCONFIGURED BEFORETOKENTRANSFER	Pending Fix
SSB_57455_27	Low	MISSING EVENTS	Pending Fix
SSB_57455_28	Low	MISSING EVENTS	Pending Fix
SSB_57455_29	Low	MISSING EVENTS	Pending Fix
SSB_57455_55	Informational	MISSING INDEXED KEYWORDS IN EVENTS	Pending Fix
SSB_57455_56	Informational	MISSING INDEXED KEYWORDS IN EVENTS	Pending Fix
SSB_57455_14	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	Pending Fix

SSB_57455_15	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	Pending Fix
SSB_57455_16	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	Pending Fix
SSB_57455_17	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	Pending Fix
SSB_57455_18	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	Pending Fix
SSB_57455_19	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	Pending Fix
SSB_57455_20	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	Pending Fix
SSB_57455_21	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	Pending Fix
SSB_57455_8	Informational	NAME MAPPING PARAMETERS	Pending Fix
SSB_57455_9	Informational	NAME MAPPING PARAMETERS	Pending Fix
SSB_57455_10	Informational	NAME MAPPING PARAMETERS	Pending Fix
SSB_57455_11	Informational	NAME MAPPING PARAMETERS	Pending Fix
SSB_57455_12	Informational	NAME MAPPING PARAMETERS	Pending Fix

SSB_57455_78	Gas	OPTIMIZING ADDRESS ID MAPPING	Pending Fix
SSB_57455_79	Gas	OPTIMIZING ADDRESS ID MAPPING	Pending Fix
SSB_57455_80	Gas	OPTIMIZING ADDRESS ID MAPPING	Pending Fix
SSB_57455_81	Gas	OPTIMIZING ADDRESS ID MAPPING	Pending Fix
SSB_57455_82	Gas	OPTIMIZING ADDRESS ID MAPPING	Pending Fix
SSB_57455_37	Gas	PUBLIC CONSTANTS CAN BE PRIVATE	Pending Fix
SSB_57455_33	Informational	REQUIRE WITH EMPTY MESSAGE	Pending Fix
SSB_57455_1	Gas	USE OF SAFEMATH LIBRARY	Pending Fix
SSB_57455_83	Gas	STORAGE VARIABLE CACHING IN MEMORY	Pending Fix
SSB_57455_83	Gas	STORAGE VARIABLE CACHING IN MEMORY	Pending Fix
SSB_57455_84	Gas	STORAGE VARIABLE CACHING IN MEMORY	Pending Fix
SSB_57455_85	Gas	STORAGE VARIABLE CACHING IN MEMORY	Pending Fix

SSB_57455_86	Gas	STORAGE VARIABLE CACHING IN MEMORY	Pending Fix
SSB_57455_87	Gas	STORAGE VARIABLE CACHING IN MEMORY	Pending Fix
SSB_57455_88	Gas	STORAGE VARIABLE CACHING IN MEMORY	Pending Fix
SSB_57455_84	Gas	STORAGE VARIABLE CACHING IN MEMORY	Pending Fix
SSB_57455_89	Gas	STORAGE VARIABLE CACHING IN MEMORY	Pending Fix
SSB_57455_90	Gas	STORAGE VARIABLE CACHING IN MEMORY	Pending Fix
SSB_57455_91	Gas	STORAGE VARIABLE CACHING IN MEMORY	Pending Fix
SSB_57455_92	Gas	STORAGE VARIABLE CACHING IN MEMORY	Pending Fix
SSB_57455_93	Gas	STORAGE VARIABLE CACHING IN MEMORY	Pending Fix
SSB_57455_88	Gas	STORAGE VARIABLE CACHING IN MEMORY	Pending Fix
SSB_57455_30	Gas	UNNECESSARY CHECKED ARITHMETIC IN LOOP	Pending Fix
SSB_57455_22	Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix

SSB_57455_23	Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSB_57455_24	Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSB_57455_25	Gas	FUNCTION SHOULD BE EXTERNAL	Pending Fix
SSB_57455_7	Low	USE OWNABLE2STEP	Pending Fix
SSB_57455_40	Gas	VARIABLES DECLARED BUT NEVER USED	Pending Fix
SSB_57455_41	Gas	VARIABLES DECLARED BUT NEVER USED	Pending Fix

Vulnerability Details

Bug ID

SSB_57455_38

Severity

Informational

Confidence

Tentative

Line nos

724-724

Action Taken

Pending Fix

Bug Type

HARD-CODED ADDRESS DETECTED

File Location

WXZO.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 -
`0x8AF7ffcDD583601a1DC28A12b0327c6D11194F03`

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_57455_39

Severity

Informational

Confidence

Tentative

Line nos

762-762

Action Taken

Pending Fix

Bug Type

HARD-CODED ADDRESS DETECTED

File Location

WXZO.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 -
`0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D`

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_57455_54

Severity

Gas

Confidence

Certain

Line nos

946-948

Action Taken

Pending Fix

Bug Type

ARRAY LENGTH CACHING

File Location

WXZO.sol

Issue Description

During each iteration of the loop, reading the length of the array uses more gas than is necessary. In the most favorable scenario, in which the length is read from a memory variable, storing the array length in the stack can save about 3 gas per iteration. In the least favorable scenario, in which external calls are made during each iteration, the amount of gas wasted can be significant.
The following array was detected to be used inside loop without caching it's value in memory: _address.

Issue Remediation

Consider storing the array length of the variable before the loop and use the stored length instead of fetching it in each iteration.

Bug ID

SSB_57455_35

Severity

Informational

Confidence

Certain

Line nos

890-890

Action Taken

Pending Fix

Bug Type

BOOLEAN EQUALITY

File Location

WXZO.sol

Issue Description

In Solidity, and many other languages, boolean constants can be used directly in conditionals like `if` and `else` statements.
The contract was found to be equating constants in conditionals which is unnecessary.

Issue Remediation

It is recommended to directly use boolean constants. It is not required to equate them to `true` or `false.`

Bug ID

SSB_57455_36

Severity

Informational

Confidence

Certain

Line nos

891-891

Action Taken

Pending Fix

Bug Type

BOOLEAN EQUALITY

File Location

WXZO.sol

Issue Description

In Solidity, and many other languages, boolean constants can be used directly in conditionals like `if` and `else` statements.
The contract was found to be equating constants in conditionals which is unnecessary.

Issue Remediation

It is recommended to directly use boolean constants. It is not required to equate them to `true` or `false.`

Bug ID

SSB_57455_57

Severity

Gas

Confidence

Tentative

Line nos

255-255

Action Taken

Pending Fix

Bug Type

CHEAPER CONDITIONAL OPERATORS

File Location

WXZO.sol

Issue Description

During compilation, `x != 0` is cheaper than `x > 0` for unsigned integers in solidity inside conditional statements.

Issue Remediation

Consider using `x != 0` in place of `x > 0` in `uint` wherever possible.

Bug ID

SSB_57455_58

Severity

Gas

Confidence

Tentative

Line nos

924-924

Action Taken

Pending Fix

Bug Type

CHEAPER CONDITIONAL OPERATORS

File Location

WXZO.sol

Issue Description

During compilation, `x != 0` is cheaper than `x > 0` for unsigned integers in solidity inside conditional statements.

Issue Remediation

Consider using `x != 0` in place of `x > 0` in `uint` wherever possible.

Bug ID

SSB_57455_59

Severity

Gas

Confidence

Tentative

Line nos

931-931

Action Taken

Pending Fix

Bug Type

CHEAPER CONDITIONAL OPERATORS

File Location

WXZO.sol

Issue Description

During compilation, `x != 0` is cheaper than `x > 0` for unsigned integers in solidity inside conditional statements.

Issue Remediation

Consider using `x != 0` in place of `x > 0` in `uint` wherever possible.

Bug ID

SSB_57455_31

Severity

Gas

Confidence

Firm

Line nos

924-924

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

WXZO.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_57455_32

Severity

Gas

Confidence

Firm

Line nos

931-931

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

WXZO.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_57455_42

Severity

Gas

Confidence

Firm

Line nos

158-158

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN REQUIRE()

File Location

WXZO.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_57455_43

Severity

Gas

Confidence

Firm

Line nos

192-192

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN REQUIRE()

File Location

WXZO.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_57455_44

Severity

Gas

Confidence

Firm

Line nos

843-843

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN REQUIRE()

File Location

WXZO.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_57455_45

Severity

Gas

Confidence

Firm

Line nos

875-875

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN REQUIRE()

File Location

WXZO.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_57455_46

Severity

Gas

Confidence

Firm

Line nos

905-905

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN REQUIRE()

File Location

WXZO.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_57455_47

Severity

Gas

Confidence

Firm

Line nos

918-918

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN REQUIRE()

File Location

WXZO.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_57455_48

Severity

Gas

Confidence

Firm

Line nos

957-957

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN REQUIRE()

File Location

WXZO.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_57455_49

Severity

Gas

Confidence

Firm

Line nos

992-992

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN REQUIRE()

File Location

WXZO.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_57455_50

Severity

Gas

Confidence

Firm

Line nos

1002-1002

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN REQUIRE()

File Location

WXZO.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_57455_51

Severity

Gas

Confidence

Firm

Line nos

1022-1022

Action Taken

Pending Fix

Bug Type

CHEAPER INEQUALITIES IN REQUIRE()

File Location

WXZO.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_57455_4

Severity

Gas

Confidence

Tentative

Line nos

325-329

Action Taken

Pending Fix

Bug Type

DEFINE CONSTRUCTOR AS PAYABLE

File Location

WXZO.sol

Issue Description

Developers can save around 10 opcodes and some gas if the constructors are defined as payable.
However, it should be noted that it comes with risks because payable constructors can accept ETH during deployment.

Issue Remediation

It is suggested to mark the constructors as payable to save some gas. Make sure it does not lead to any adverse effects in case an upgrade pattern is involved.

Bug ID

SSB_57455_5

Severity

Gas

Confidence

Tentative

Line nos

755-775

Action Taken

Pending Fix

Bug Type

DEFINE CONSTRUCTOR AS PAYABLE

File Location

WXZO.sol

Issue Description

Developers can save around 10 opcodes and some gas if the constructors are defined as payable.
However, it should be noted that it comes with risks because payable constructors can accept ETH during deployment.

Issue Remediation

It is suggested to mark the constructors as payable to save some gas. Make sure it does not lead to any adverse effects in case an upgrade pattern is involved.

Bug ID

SSB_57455_26

Severity

Low

Confidence

Certain

Line nos

2-2

Action Taken

Pending Fix

Bug Type

USE OF FLOATING PRAGMA

File Location

WXZO.sol

Issue Description

Solidity source files indicate the versions of the compiler they can be compiled with using a pragma directive at the top of the solidity file. This can either be a floating pragma or a specific compiler version.
The contract was found to be using a floating pragma which is not considered safe as it can be compiled with all the versions described.
The following affected files were found to be using floating pragma:
`['WXZO.sol'] - ^0.8.18`

Issue Remediation

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

Instead of `^0.8.18` use `pragma solidity 0.8.18`, which is a stable and recommended version right now.

Bug ID

SSB_57455_52

Severity

Gas

Confidence

Certain

Line nos

1009-1009

Action Taken

Pending Fix

Bug Type

GAS OPTIMIZATION FOR STATE VARIABLES

File Location

WXZO.sol

Issue Description

Plus equals (`+=`) costs more gas than addition operator. The same thing happens with minus equals (`-=`). Therefore, `x +=y` costs more gas than `x = x + y`.

Issue Remediation

Consider

- addition operator over plus equals
- subtraction operator over minus equals
- division operator over divide equals
- multiplication operator over multiply equals

Bug ID

SSB_57455_53

Severity

Gas

Confidence

Certain

Line nos

1024-1024

Action Taken

Pending Fix

Bug Type

GAS OPTIMIZATION FOR STATE VARIABLES

File Location

WXZO.sol

Issue Description

Plus equals (`+=`) costs more gas than addition operator. The same thing happens with minus equals (`-=`). Therefore, `x -=y` costs more gas than `x = x - y`.

Issue Remediation

Consider

- addition operator over plus equals
- subtraction operator over minus equals
- division operator over divide equals
- multiplication operator over multiply equals

Bug ID

SSB_57455_13

Severity

Gas

Confidence

Tentative

Line nos

946-946

Action Taken

Pending Fix

Bug Type

GAS OPTIMIZATION IN INCREMENTS

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_34

Severity

Critical

Confidence

Firm

Line nos

833-853

Action Taken

Pending Fix

Bug Type

INCORRECT ACCESS CONTROL

File Location

WXZO.sol

Issue Description

Access control plays an important role in segregation of privileges in smart contracts and other applications. If this is misconfigured or not properly validated on sensitive functions, it may lead to loss of funds, tokens and in some cases compromise of the smart contract.

The contract EXZO is importing an access control library @openzeppelin/contracts/access/Ownable.sol but the function transferFrom is missing the modifier onlyOwner.

Issue Remediation

It is recommended to go through the contract and observe the functions that are lacking an access control modifier. If they contain sensitive administrative actions, it is advised to add a suitable modifier to the same

Bug ID

SSB_57455_6

Severity

Gas

Confidence

Tentative

Line nos

125-128

Action Taken

Pending Fix

Bug Type

INTERNAL FUNCTIONS NEVER USED

File Location

WXZO.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_57455_60

Severity

Gas

Confidence

Certain

Line nos

217-217

Action Taken

Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

WXZO.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_57455_61

Severity

Gas

Confidence

Certain

Line nos

363-366

Action Taken

Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

WXZO.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_57455_62

Severity

Gas

Confidence

Certain

Line nos

874-877

Action Taken

Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

WXZO.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_57455_63

Severity

Gas

Confidence

Certain

Line nos

888-888

Action Taken

Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

WXZO.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_57455_64

Severity

Gas

Confidence

Certain

Line nos

889-889

Action Taken

Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

WXZO.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_57455_65

Severity

Gas

Confidence

Certain

Line nos

890-890

Action Taken

Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

WXZO.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_57455_66

Severity

Gas

Confidence

Certain

Line nos

891-891

Action Taken

Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

WXZO.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_57455_67

Severity

Gas

Confidence

Certain

Line nos

917-920

Action Taken

Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

WXZO.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_57455_68

Severity

Gas

Confidence

Certain

Line nos

905-905

Action Taken

Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

WXZO.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_57455_69

Severity

Gas

Confidence

Certain

Line nos

945-945

Action Taken

Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

WXZO.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_57455_70

Severity

Gas

Confidence

Certain

Line nos

956-959

Action Taken

Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

WXZO.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_57455_71

Severity

Gas

Confidence

Certain

Line nos

971-971

Action Taken

Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

WXZO.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_57455_72

Severity

Gas

Confidence

Certain

Line nos

986-986

Action Taken

Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

WXZO.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_57455_73

Severity

Gas

Confidence

Certain

Line nos

991-994

Action Taken

Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

WXZO.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_57455_74

Severity

Gas

Confidence

Certain

Line nos

1001-1004

Action Taken

Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

WXZO.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_57455_75

Severity

Gas

Confidence

Certain

Line nos

1022-1022

Action Taken

Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

WXZO.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_57455_76

Severity

Gas

Confidence

Certain

Line nos

1034-1034

Action Taken

Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

WXZO.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_57455_77

Severity

Gas

Confidence

Certain

Line nos

1035-1035

Action Taken

Pending Fix

Bug Type

LONG REQUIRE/REVERT STRINGS

File Location

WXZO.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_57455_2

Severity

Low

Confidence

Tentative

Line nos

1007-1007

Action Taken

Pending Fix

Bug Type

MISCONFIGURED BEFORETOKENTRANSFER

File Location

WXZO.sol

Issue Description

The `_beforeTokenTransfer()` function used by the contract was found to be misconfigured and might give incorrect results or fail to compile.

Issue Remediation

Ensure that the detected function has a virtual modifier and uses the super keyword.

Bug ID

SSB_57455_3

Severity

Low

Confidence

Tentative

Line nos

1019-1019

Action Taken

Pending Fix

Bug Type

MISCONFIGURED BEFORETOKENTRANSFER

File Location

WXZO.sol

Issue Description

The `_beforeTokenTransfer()` function used by the contract was found to be misconfigured and might give incorrect results or fail to compile.

Issue Remediation

Ensure that the detected function has a virtual modifier and uses the super keyword.

Bug ID

SSB_57455_27

Severity

Low

Confidence

Firm

Line nos

941-949

Action Taken

Pending Fix

Bug Type

MISSING EVENTS

File Location

WXZO.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 EXZO was found to be missing these events on the function addToBlacklist 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_57455_28

Severity

Low

Confidence

Firm

Line nos

967-973

Action Taken

Pending Fix

Bug Type

MISSING EVENTS

File Location

WXZO.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 EXZO was found to be missing these events on the function ExcludeOrIncludeFromFee 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_57455_29

Severity

Low

Confidence

Firm

Line nos

984-988

Action Taken

Pending Fix

Bug Type

MISSING EVENTS

File Location

WXZO.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 EXZO was found to be missing these events on the function SetAutomaticMarketMaker 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_57455_55

Severity

Informational

Confidence

Certain

Line nos

465-465

Action Taken

Pending Fix

Bug Type

MISSING INDEXED KEYWORDS IN EVENTS

File Location

WXZO.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_57455_56

Severity

Informational

Confidence

Certain

Line nos

749-753

Action Taken

Pending Fix

Bug Type

MISSING INDEXED KEYWORDS IN EVENTS

File Location

WXZO.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_57455_14

Severity

Informational

Confidence

Tentative

Line nos

156-161

Action Taken

Pending Fix

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_15

Severity

Informational

Confidence

Tentative

Line nos

173-175

Action Taken

Pending Fix

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_16

Severity

Informational

Confidence

Tentative

Line nos

187-196

Action Taken

Pending Fix

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_17

Severity

Informational

Confidence

Tentative

Line nos

208-220

Action Taken

Pending Fix

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_18

Severity

Informational

Confidence

Tentative

Line nos

234-236

Action Taken

Pending Fix

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_19

Severity

Informational

Confidence

Tentative

Line nos

250-260

Action Taken

Pending Fix

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_20

Severity

Informational

Confidence

Tentative

Line nos

274-276

Action Taken

Pending Fix

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_21

Severity

Informational

Confidence

Tentative

Line nos

290-297

Action Taken

Pending Fix

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_8

Severity

Informational

Confidence

Tentative

Line nos

718-718

Action Taken

Pending Fix

Bug Type

NAME MAPPING PARAMETERS

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_9

Severity

Informational

Confidence

Tentative

Line nos

720-720

Action Taken

Pending Fix

Bug Type

NAME MAPPING PARAMETERS

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_10

Severity

Informational

Confidence

Tentative

Line nos

721-721

Action Taken

Pending Fix

Bug Type

NAME MAPPING PARAMETERS

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_11

Severity

Informational

Confidence

Tentative

Line nos

722-722

Action Taken

Pending Fix

Bug Type

NAME MAPPING PARAMETERS

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_12

Severity

Informational

Confidence

Tentative

Line nos

723-723

Action Taken

Pending Fix

Bug Type

NAME MAPPING PARAMETERS

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_78

Severity

Gas

Confidence

Tentative

Line nos

718-718

Action Taken

Pending Fix

Bug Type

OPTIMIZING ADDRESS ID MAPPING

File Location

WXZO.sol

Issue Description

Combining multiple address/ID mappings into a single mapping using a struct enhances storage efficiency, simplifies code, and reduces gas costs, resulting in a more streamlined and cost-effective smart contract design.
It saves storage slot for the mapping and depending on the circumstances and sizes of types, it can avoid a Gsset (20000 gas) per mapping combined. Reads and subsequent writes can also be cheaper when a function requires both values and they fit in the same storage slot.

Issue Remediation

It is suggested to modify the code so that multiple mappings using the address→id parameter are combined into a struct.

Bug ID

SSB_57455_79

Severity

Gas

Confidence

Tentative

Line nos

720-720

Action Taken

Pending Fix

Bug Type

OPTIMIZING ADDRESS ID MAPPING

File Location

WXZO.sol

Issue Description

Combining multiple address/ID mappings into a single mapping using a struct enhances storage efficiency, simplifies code, and reduces gas costs, resulting in a more streamlined and cost-effective smart contract design.
It saves storage slot for the mapping and depending on the circumstances and sizes of types, it can avoid a Gsset (20000 gas) per mapping combined. Reads and subsequent writes can also be cheaper when a function requires both values and they fit in the same storage slot.

Issue Remediation

It is suggested to modify the code so that multiple mappings using the address→id parameter are combined into a struct.

Bug ID

SSB_57455_80

Severity

Gas

Confidence

Tentative

Line nos

721-721

Action Taken

Pending Fix

Bug Type

OPTIMIZING ADDRESS ID MAPPING

File Location

WXZO.sol

Issue Description

Combining multiple address/ID mappings into a single mapping using a struct enhances storage efficiency, simplifies code, and reduces gas costs, resulting in a more streamlined and cost-effective smart contract design.
It saves storage slot for the mapping and depending on the circumstances and sizes of types, it can avoid a Gsset (20000 gas) per mapping combined. Reads and subsequent writes can also be cheaper when a function requires both values and they fit in the same storage slot.

Issue Remediation

It is suggested to modify the code so that multiple mappings using the address→id parameter are combined into a struct.

Bug ID

SSB_57455_81

Severity

Gas

Confidence

Tentative

Line nos

722-722

Action Taken

Pending Fix

Bug Type

OPTIMIZING ADDRESS ID MAPPING

File Location

WXZO.sol

Issue Description

Combining multiple address/ID mappings into a single mapping using a struct enhances storage efficiency, simplifies code, and reduces gas costs, resulting in a more streamlined and cost-effective smart contract design.
It saves storage slot for the mapping and depending on the circumstances and sizes of types, it can avoid a Gsset (20000 gas) per mapping combined. Reads and subsequent writes can also be cheaper when a function requires both values and they fit in the same storage slot.

Issue Remediation

It is suggested to modify the code so that multiple mappings using the address→id parameter are combined into a struct.

Bug ID

SSB_57455_82

Severity

Gas

Confidence

Tentative

Line nos

723-723

Action Taken

Pending Fix

Bug Type

OPTIMIZING ADDRESS ID MAPPING

File Location

WXZO.sol

Issue Description

Combining multiple address/ID mappings into a single mapping using a struct enhances storage efficiency, simplifies code, and reduces gas costs, resulting in a more streamlined and cost-effective smart contract design.
It saves storage slot for the mapping and depending on the circumstances and sizes of types, it can avoid a Gsset (20000 gas) per mapping combined. Reads and subsequent writes can also be cheaper when a function requires both values and they fit in the same storage slot.

Issue Remediation

It is suggested to modify the code so that multiple mappings using the address→id parameter are combined into a struct.

Bug ID

SSB_57455_37

Severity

Gas

Confidence

Certain

Line nos

724-724

Action Taken

Pending Fix

Bug Type

PUBLIC CONSTANTS CAN BE PRIVATE

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_33

Severity

Informational

Confidence

Certain

Line nos

838-838

Action Taken

Pending Fix

Bug Type

REQUIRE WITH EMPTY MESSAGE

File Location

WXZO.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_57455_1

Severity

Gas

Confidence

Certain

Line nos

717-717

Action Taken

Pending Fix

Bug Type

USE OF SAFEMATH LIBRARY

File Location

WXZO.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_57455_83

Severity

Gas

Confidence

Tentative

Line nos

313-313

Action Taken

Pending Fix

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_83

Severity

Gas

Confidence

Tentative

Line nos

313-313

Action Taken

Pending Fix

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_84

Severity

Gas

Confidence

Tentative

Line nos

721-721

Action Taken

Pending Fix

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_85

Severity

Gas

Confidence

Tentative

Line nos

730-730

Action Taken

Pending Fix

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_86

Severity

Gas

Confidence

Tentative

Line nos

742-742

Action Taken

Pending Fix

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

WXZO.sol

Issue Description

The contract `EXZO` is using the state variable `uniswapV2Pair` multiple times in the function .
`SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD` / `MSTORE` (3 gas each).

Issue Remediation

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

Bug ID

SSB_57455_87

Severity

Gas

Confidence

Tentative

Line nos

720-720

Action Taken

Pending Fix

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_88

Severity

Gas

Confidence

Tentative

Line nos

718-718

Action Taken

Pending Fix

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_84

Severity

Gas

Confidence

Tentative

Line nos

721-721

Action Taken

Pending Fix

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_89

Severity

Gas

Confidence

Tentative

Line nos

722-722

Action Taken

Pending Fix

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_90

Severity

Gas

Confidence

Tentative

Line nos

723-723

Action Taken

Pending Fix

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_91

Severity

Gas

Confidence

Tentative

Line nos

724-724

Action Taken

Pending Fix

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_92

Severity

Gas

Confidence

Tentative

Line nos

743-743

Action Taken

Pending Fix

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_93

Severity

Gas

Confidence

Tentative

Line nos

727-727

Action Taken

Pending Fix

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_88

Severity

Gas

Confidence

Tentative

Line nos

718-718

Action Taken

Pending Fix

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

WXZO.sol

Issue Description

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

Issue Remediation

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

Bug ID

SSB_57455_30

Severity

Gas

Confidence

Tentative

Line nos

946-946

Action Taken

Pending Fix

Bug Type

UNNECESSARY CHECKED ARITHMETIC IN LOOP

File Location

WXZO.sol

Issue Description

Increments inside a loop could never overflow due to the fact that the transaction will run out of gas before the variable reaches its limits. Therefore, it makes no sense to have checked arithmetic in such a place.

Issue Remediation

It is recommended to have the increment value inside the unchecked block to save some gas.

Bug ID

SSB_57455_22

Severity

Gas

Confidence

Certain

Line nos

353-356

Action Taken

Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

WXZO.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_57455_23

Severity

Gas

Confidence

Certain

Line nos

362-369

Action Taken

Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

WXZO.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_57455_24

Severity

Gas

Confidence

Certain

Line nos

868-881

Action Taken

Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

WXZO.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_57455_25

Severity

Gas

Confidence

Certain

Line nos

855-866

Action Taken

Pending Fix

Bug Type

FUNCTION SHOULD BE EXTERNAL

File Location

WXZO.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_57455_7

Severity

Low

Confidence

Tentative

Line nos

716-1046

Action Taken

Pending Fix

Bug Type

USE OWNABLE2STEP

File Location

WXZO.sol

Issue Description

`Ownable2Step` is safer than `Ownable` for smart contracts because the owner cannot accidentally transfer the ownership to a mistyped address. Rather than directly transferring to the new owner, the transfer only completes when the new owner accepts ownership.

Issue Remediation

It is recommended to use either `Ownable2Step` or `Ownable2StepUpgradeable` depending on the smart contract.

Bug ID

SSB_57455_40

Severity

Gas

Confidence

Tentative

Line nos

314-314

Action Taken

Pending Fix

Bug Type

VARIABLES DECLARED BUT NEVER USED

File Location

WXZO.sol

Issue Description

The contract Ownable has declared a variable _previousOwner but it is not used anywhere in the code. This represents dead code or missing logic.
Unused variables increase the contract's size and complexity, potentially leading to higher gas costs and a larger attack surface.

Issue Remediation

To remediate this vulnerability, developers should perform a code review and remove any variables that are declared but never used.

Bug ID

SSB_57455_41

Severity

Gas

Confidence

Tentative

Line nos

315-315

Action Taken

Pending Fix

Bug Type

VARIABLES DECLARED BUT NEVER USED

File Location

WXZO.sol

Issue Description

The contract Ownable has declared a variable _lockTime but it is not used anywhere in the code. This represents dead code or missing logic.
Unused variables increase the contract's size and complexity, potentially leading to higher gas costs and a larger attack surface.

Issue Remediation

To remediate this vulnerability, developers should perform a code review and remove any variables that are declared but never used.

Scan History

Critical High Medium Low Informational Gas

No	Date	Security Score	Scan Overview
1.	2023-09-08	89.77	1 0 0 7 20 68

Disclaimer

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

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

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

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

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

The assessment provided by SolidityScan is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. SolidityScan owes no duty to any third party by virtue of publishing these Reports.

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