SolidityScan | Powered by CRED SHIELDS

Security Assessment Report

Empx Limit Orders

7 Jan 2026

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

Table of Contents

IF-STATEMENT REFACTORING

MISSING @AUTHOR IN NATSPEC COMMENTS FOR CONTRACT DECLARATION

MISSING @DEV IN NATSPEC COMMENTS FOR CONTRACT DECLARATION

MISSING @DEV IN NATSPEC COMMENTS FOR FUNCTIONS

MISSING INDEXED KEYWORDS IN EVENTS

MISSING @INHERITDOC ON OVERRIDE FUNCTIONS

MISSING NATSPEC COMMENTS IN SCOPE BLOCKS

MISSING NATSPEC DESCRIPTIONS FOR PUBLIC VARIABLE DECLARATIONS

MISSING @NOTICE IN NATSPEC COMMENTS FOR CONSTRUCTORS

MISSING @NOTICE IN NATSPEC COMMENTS FOR FUNCTIONS

NAME MAPPING PARAMETERS

REVERT STATEMENTS WITHIN EXTERNAL AND PUBLIC FUNCTIONS CAN BE USED TO PERFORM DOS ATTACKS

USE OF DECIMALS

USE SCIENTIFIC NOTATION

ARRAY LENGTH CACHING

ASSIGNING TO STRUCTS CAN BE MORE EFFICIENT

AVOID RE-STORING VALUES

AVOID ZERO-TO-ONE STORAGE WRITES

CACHE ADDRESS(THIS) WHEN USED MORE THAN ONCE

GAS OPTIMIZATION FOR THIS KEYWORD

CHEAPER CONDITIONAL OPERATORS

CHEAPER INEQUALITIES IN IF()

CHEAPER INEQUALITIES IN REQUIRE()

DEFAULT INT VALUES ARE MANUALLY RESET

DEFINE CONSTRUCTOR AS PAYABLE

EMIT USED IN LOOP

FUNCTIONS CAN BE IN-LINED

01. Vulnerability Classification and Severity

Description

To enhance navigability, the document is organized in descending order of severity for easy reference. Issues are categorized as ✓ *Fixed*, ⚠ *Pending Fix*, or ✗ *Won't Fix*, indicating their current status. ✗ *Won't Fix* denotes that the team is aware of the issue but has chosen not to resolve it. Issues labeled as ⚠ *Pending Fix* state that the bug is yet to be resolved. Additionally, each issue's severity is assessed based on the risk of exploitation or the potential for other unexpected or unsafe behavior.

- **Critical**

  The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

- **High**

  High-severity vulnerabilities pose a significant risk to both the Smart Contract and the organization. They can lead to user fund losses, may have conditional requirements, and are challenging to exploit.

- **Medium**

  The issue affects the ability of the contract to operate in a way that doesn't significantly hinder its behavior.

- **Low**

  The issue has minimal impact on the contract's ability to operate.

- **Informational**

  The issue does not affect the contract's operational capability but is considered good practice to address.

- **Gas**

  This category deals with optimizing code and refactoring to conserve gas.

02. Executive Summary

Empx Limit Orders

Uploaded Solidity File(s)

Language	Audit Methodology	Website
Solidity	**Static Scanning**	-
Publishers/Owner Name	Organization	Contact Email
-	-	-

Security Score is GREAT

The SolidityScan score is calculated based on lines of code and weights assigned to each issue depending on the severity and confidence. To improve your score, view the detailed result and leverage the remediation solutions provided.

This report has been prepared for Empx Limit Orders 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 700+ modules. The scanning and auditing process covers the following areas:

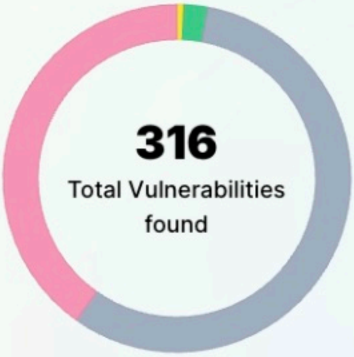
Various common and uncommon attack vectors will be investigated to ensure that the smart contracts are secure from malicious actors. The scanner modules find and flag issues related to Gas optimizations that help in reducing the overall Gas cost It scans and evaluates the codebase against industry best 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 Empx Limit Orders introduces new features or refactors the code.

03. Findings Summary

Empx Limit Orders
File Scan

Security Score	Scan duration	Lines of code
90.46/100	296 secs	674



Crit	High	Med	Low	Info	Gas
0	0	2	7	180	127

This audit report has not been verified by the SolidityScan team. To learn more about our published reports. click here

ACTION TAKEN

0	24	3	319
Fixed	False Positive	Won't Fix	Pending Fix

S. No.	Severity	Bug Type	Instances	Detection Method	Status
H001	● High	BRIDGE MINT LIMITS NOT ENFORCED	1	SolidityScan AI	False Positive
M001	● Medium	UNCHECKED ARRAY LENGTH	5	Automated	False Positive
M002	● Medium	LIMITATIONS OF SOLIDITY'S TRY-CATCH IN EXTERNAL CALLS	1	Automated	False Positive
M003	● Medium	TOKEN DECIMALS MISMATCH IN SEIZE REPAY	1	SolidityScan AI	Pending Fix
L001	● Low	MISSING SAFE ERC20 USAGE	1	SolidityScan AI	Pending Fix
L002	● Low	ORDERCREATED EVENT EMITS USER-SPECIFIED AMOUNTIN INSTEAD OF THE ACTUAL TOKENS ESCROWED	1	SolidityScan AI	Pending Fix
L003	● Low	USE OF FLOATING PRAGMA	1	Automated	Pending Fix
L004	● Low	FUNCTION RETURNS TYPE AND NO RETURN	1	Automated	False Positive
L005	● Low	LACK OF ZERO VALUE CHECK IN TOKEN TRANSFERS	2	Automated	False Positive
L006	● Low	MISSING EVENTS	7	Automated	False Positive
L007	● Low	MISSING ZERO ADDRESS VALIDATION	7	Automated	False Positive
L008	● Low	OUTDATED COMPILER VERSION	1	Automated	Pending Fix
L009	● Low	USE FORCEAPPROVE IN PLACE OF APPROVE	2	Automated	Pending Fix
L010	● Low	USE OWNABLE2STEP	1	Automated	Pending Fix
I001	● Informational	ABI.ENCODEPACKED MAY CAUSE COLLISION	1	Automated	Pending Fix

ACTION TAKEN

0	24	3	319
Fixed	False Positive	Won't Fix	Pending Fix

S. No.	Severity	Bug Type	Instances	Detection Method	Status
H001	● High	BRIDGE MINT LIMITS NOT ENFORCED	1	SolidityScan AI	False Positive
M001	● Medium	UNCHECKED ARRAY LENGTH	5	Automated	False Positive
M002	● Medium	LIMITATIONS OF SOLIDITY'S TRY-CATCH IN EXTERNAL CALLS	1	Automated	False Positive
M003	● Medium	TOKEN DECIMALS MISMATCH IN SEIZE REPAY	1	SolidityScan AI	Pending Fix
L001	● Low	MISSING SAFE ERC20 USAGE	1	SolidityScan AI	Pending Fix
L002	● Low	ORDERCREATED EVENT EMITS USER-SPECIFIED AMOUNTIN INSTEAD OF THE ACTUAL TOKENS ESCROWED	1	SolidityScan AI	Pending Fix
L003	● Low	USE OF FLOATING PRAGMA	1	Automated	Pending Fix
L004	● Low	FUNCTION RETURNS TYPE AND NO RETURN	1	Automated	False Positive
L005	● Low	LACK OF ZERO VALUE CHECK IN TOKEN TRANSFERS	2	Automated	False Positive
L006	● Low	MISSING EVENTS	7	Automated	False Positive
L007	● Low	MISSING ZERO ADDRESS VALIDATION	7	Automated	False Positive
L008	● Low	OUTDATED COMPILER VERSION	1	Automated	Pending Fix
L009	● Low	USE FORCEAPPROVE IN PLACE OF APPROVE	2	Automated	Pending Fix
L010	● Low	USE OWNABLE2STEP	1	Automated	Pending Fix
I001	● Informational	ABI.ENCODEPACKED MAY CAUSE COLLISION	1	Automated	Pending Fix

S. No.	Severity	Bug Type	Instances	Detection Method	Status
I002	● Informational	ADDING A RETURN STATEMENT WHEN THE FUNCTION DEFINES A NAMED RETURN VARIABLE IS REDUNDANT	1	Automated	⚠ *Pending Fix*
I003	● Informational	AVOID ARITHMETIC DIRECTLY WITHIN ARRAY INDICES	1	Automated	⚠ *Pending Fix*
I004	● Informational	BLOCK VALUES AS A PROXY FOR TIME	3	Automated	⚠ *Pending Fix*
I005	● Informational	IF-STATEMENT REFACTORING	3	Automated	⚠ *Pending Fix*
I006	● Informational	MISSING @AUTHOR IN NATSPEC COMMENTS FOR CONTRACT DECLARATION	1	Automated	⚠ *Pending Fix*
I007	● Informational	MISSING @DEV IN NATSPEC COMMENTS FOR CONTRACT DECLARATION	1	Automated	⚠ *Pending Fix*
I008	● Informational	MISSING @DEV IN NATSPEC COMMENTS FOR FUNCTIONS	33	Automated	⚠ *Pending Fix*
I009	● Informational	MISSING INDEXED KEYWORDS IN EVENTS	1	Automated	⚠ *Pending Fix*
I010	● Informational	MISSING @INHERITDOC ON OVERRIDE FUNCTIONS	28	Automated	⚠ *Pending Fix*
I011	● Informational	MISSING NATSPEC COMMENTS IN SCOPE BLOCKS	40	Automated	⚠ *Pending Fix*
I012	● Informational	MISSING NATSPEC DESCRIPTIONS FOR PUBLIC VARIABLE DECLARATIONS	18	Automated	⚠ *Pending Fix*
I013	● Informational	MISSING @NOTICE IN NATSPEC COMMENTS FOR CONSTRUCTORS	1	Automated	⚠ *Pending Fix*
I014	● Informational	MISSING @NOTICE IN NATSPEC COMMENTS FOR FUNCTIONS	27	Automated	⚠ *Pending Fix*
I015	● Informational	NAME MAPPING PARAMETERS	8	Automated	⚠ *Pending Fix*
I016	● Informational	REVERT STATEMENTS WITHIN EXTERNAL AND PUBLIC FUNCTIONS CAN BE USED TO PERFORM DOS ATTACKS	9	Automated	⚠ *Pending Fix*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
I017	● Informational	USE OF DECIMALS	2	Automated	⚠ *Pending Fix*
I018	● Informational	USE SCIENTIFIC NOTATION	2	Automated	⚠ *Pending Fix*
G001	● Gas	ARRAY LENGTH CACHING	10	Automated	⚠ *Pending Fix*
G002	● Gas	ASSIGNING TO STRUCTS CAN BE MORE EFFICIENT	1	Automated	⚠ *Pending Fix*
G003	● Gas	AVOID RE-STORING VALUES	5	Automated	⚠ *Pending Fix*
G004	● Gas	AVOID ZERO-TO-ONE STORAGE WRITES	9	Automated	⚠ *Pending Fix*
G005	● Gas	CACHE ADDRESS(THIS) WHEN USED MORE THAN ONCE	8	Automated	⚠ *Pending Fix*
G006	● Gas	GAS OPTIMIZATION FOR THIS KEYWORD	1	Automated	⚠ *Pending Fix*
G007	● Gas	CHEAPER CONDITIONAL OPERATORS	6	Automated	⚠ *Pending Fix*
G008	● Gas	CHEAPER INEQUALITIES IN IF()	9	Automated	⚠ *Pending Fix*
G009	● Gas	CHEAPER INEQUALITIES IN REQUIRE()	4	Automated	⚠ *Pending Fix*
G010	● Gas	DEFAULT INT VALUES ARE MANUALLY RESET	3	Automated	⚠ *Pending Fix*
G011	● Gas	DEFINE CONSTRUCTOR AS PAYABLE	1	Automated	⚠ *Pending Fix*
G012	● Gas	EMIT USED IN LOOP	1	Automated	⚠ *Pending Fix*
G013	● Gas	FUNCTIONS CAN BE IN-LINED	8	Automated	⚠ *Pending Fix*
G014	● Gas	REVERTING FUNCTIONS CAN BE PAYABLE	11	Automated	⚠ *Pending Fix*
G015	● Gas	FUNCTION SHOULD RETURN STRUCT	1	Automated	⚠ *Pending Fix*
G016	● Gas	GAS INEFFICIENCY DUE TO MULTIPLE OPERANDS IN SINGLE IF/ELSEIF CONDITION	8	Automated	⚠ *Pending Fix*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
G017	● Gas	GAS OPTIMIZATION FOR STATE VARIABLES	1	Automated	⚠ ***Pending Fix***
G018	● Gas	GAS OPTIMIZATION IN INCREMENTS	10	Automated	⚠ ***Pending Fix***
G019	● Gas	NAMED RETURN OF LOCAL VARIABLE SAVES GAS AS COMPARED TO RETURN STATEMENT	2	Automated	⚠ ***Pending Fix***
G020	● Gas	OPTIMIZING ADDRESS ID MAPPING	4	Automated	⚠ ***Pending Fix***
G021	● Gas	SIMILAR DATATYPES CAN BE PACKED TOGETHER	1	Automated	⚠ ***Pending Fix***
G022	● Gas	SMALLER DATA TYPES COST MORE	2	Automated	⚠ ***Pending Fix***
G023	● Gas	SPLITTING REQUIRE STATEMENTS	1	Automated	⚠ ***Pending Fix***
G024	● Gas	SPLITTING REVERT STATEMENTS	2	Automated	⚠ ***Pending Fix***
G025	● Gas	STORAGE VARIABLE CACHING IN MEMORY	8	Automated	⚠ ***Pending Fix***
G026	● Gas	STORING STORAGE VARIABLES IN MEMORY	3	Automated	⚠ ***Pending Fix***
G027	● Gas	UNNECESSARY CHECKED ARITHMETIC IN LOOP	14	Automated	⚠ ***Pending Fix***

04. Vulnerability Details

Issue Type

BRIDGE MINT LIMITS NOT ENFORCED

Upgrade your Plan to view the full report

1 High Issues Found

Please upgrade your plan to view all the issues in your report.

Upgrade

Issue Type

UNCHECKED ARRAY LENGTH

S. No.	Severity	Detection Method	Instances
M001	● Medium	Automated	5

[illegible]

Upgrade your Plan to view the full report

5 Medium Issues Found

Please upgrade your plan to view all the issues in your report.

Upgrade

Issue Type

MISSING SAFE ERC20 USAGE

S. No.	Severity	Detection Method	Instances
L001	● Low	SolidityScan AI	1

[illegible]

Upgrade your Plan to view the full report

1 Low Issues Found

Please upgrade your plan to view all the issues in your report.

Upgrade

Issue Type

ABI.ENCODEPACKED MAY CAUSE COLLISION

S. No.	Severity	Detection Method	Instances
I001	● Informational	Automated	1

[redacted]

Upgrade your Plan to view the full report

1 Informational Issues Found

Please upgrade your plan to view all the issues in your report.

Upgrade

Issue Type

ARRAY LENGTH CACHING

S. No.	Severity	Detection Method	Instances
G001	● Gas	Automated	10

Description

During each iteration of the loop, reading the length of the array uses more gas than is necessary. In the most favorable sc enario, in which the length is read from a memory variable, storing the array length in the stack can save about 3 gas per it eration. In the least favorable scenario, in which external calls are made during each iteration, the amount of gas wasted c an be significant.

Bug ID	File Location	Line No.	Action Taken
SSP_120106_38	LimitOrdersEscrow.sol	L178 - L180	⚠ *Pending Fix*
SSP_120106_39	LimitOrdersEscrow.sol	L276 - L290	⚠ *Pending Fix*
SSP_120106_40	LimitOrdersEscrow.sol	L381 - L401	⚠ *Pending Fix*
SSP_120106_41	LimitOrdersEscrow.sol	L421 - L435	⚠ *Pending Fix*
SSP_120106_42	LimitOrdersEscrow.sol	L440 - L452	⚠ *Pending Fix*
SSP_120106_43	LimitOrdersEscrow.sol	L458 - L462	⚠ *Pending Fix*
SSP_120106_44	LimitOrdersEscrow.sol	L466 - L470	⚠ *Pending Fix*
SSP_120106_45	LimitOrdersEscrow.sol	L480 - L482	⚠ *Pending Fix*
SSP_120106_46	LimitOrdersEscrow.sol	L774 - L777	⚠ *Pending Fix*
SSP_120106_47	LimitOrdersEscrow.sol	L782 - L784	⚠ *Pending Fix*

Issue Type

ASSIGNING TO STRUCTS CAN BE MORE EFFICIENT

S. No.	Severity	Detection Method	Instances
G002	● Gas	Automated	1

Description

The contract is found to contain a struct with multiple variables defined in it. When a struct is assigned in a single operatio n, Solidity may perform costly storage operations, which can be inefficient. This often results in increased gas costs due to multiple SLOAD and SSTORE operations happening at once

Bug ID	File Location	Line No.	Action Taken
SSP_120106_58	LimitOrdersEscrow.sol	L211 - L226	⚠ *Pending Fix*

Issue Type

AVOID RE-STORING VALUES

S. No.	Severity	Detection Method	Instances
G003	● Gas	Automated	5

Description

The function is found to be allowing re-storing the value in the contract's state variable even when the old value is equal to the new value. This practice results in unnecessary gas consumption due to the `Gsreset` operation (2900 gas), which co uld be avoided. If the old value and the new value are the same, not updating the storage would avoid this cost and could i nstead incur a `Gcoldsload` (2100 gas) or a `Gwarmaccess` (100 gas), potentially saving gas.

Bug ID	File Location	Line No.	Action Taken
SSP_120106_326	LimitOrdersEscrow.sol	L749 - L752	⚠ *Pending Fix*
SSP_120106_327	LimitOrdersEscrow.sol	L757 - L761	⚠ *Pending Fix*
SSP_120106_328	LimitOrdersEscrow.sol	L768 - L771	⚠ *Pending Fix*
SSP_120106_329	LimitOrdersEscrow.sol	L788 - L791	⚠ *Pending Fix*
SSP_120106_330	LimitOrdersEscrow.sol	L793 - L796	⚠ *Pending Fix*

Issue Type

AVOID ZERO-TO-ONE STORAGE WRITES

S. No.	Severity	Detection Method	Instances
G004	● Gas	Automated	9

Description

Writing a storage variable from zero to a non-zero value costs 22,100 gas (20,000 for the write and 2,100 for cold access), making it one of the most expensive operations. This is why patterns like OpenZeppelin's `ReentrancyGuard` use `1` and `2` instead of `0` and `1`—to avoid the high cost of zero-to-non-zero writes. Non-zero to non-zero updates cost only 5,00 0 gas.

Bug ID	File Location	Line No.	Action Taken
SSP_120106_125	LimitOrdersEscrow.sol	L173 - L173	⚠ *Pending Fix*
SSP_120106_126	LimitOrdersEscrow.sol	L174 - L174	⚠ *Pending Fix*
SSP_120106_127	LimitOrdersEscrow.sol	L175 - L175	⚠ *Pending Fix*
SSP_120106_127	LimitOrdersEscrow.sol	L769 - L769	⚠ *Pending Fix*
SSP_120106_128	LimitOrdersEscrow.sol	L176 - L176	⚠ *Pending Fix*
SSP_120106_129	LimitOrdersEscrow.sol	L702 - L702	⚠ *Pending Fix*
SSP_120106_130	LimitOrdersEscrow.sol	L759 - L759	⚠ *Pending Fix*
SSP_120106_131	LimitOrdersEscrow.sol	L760 - L760	⚠ *Pending Fix*

Bug ID	File Location	Line No.	Action Taken
SSP_120106_132	LimitOrdersEscrow.sol	L770 - L770	⚠ *Pending Fix*

Issue Type

CACHE ADDRESS(THIS) WHEN USED MORE THAN ONCE

S. No.	Severity	Detection Method	Instances
G005	● Gas	Automated	8

Description

The repeated usage of `address(this)` within the contract could result in increased gas costs due to multiple executions of the same computation, potentially impacting efficiency and overall transaction expenses.

Bug ID	File Location	Line No.	Action Taken
SSP_120106_177	LimitOrdersEscrow.sol	L203 - L203	⚠ *Pending Fix*
SSP_120106_178	LimitOrdersEscrow.sol	L204 - L204	⚠ *Pending Fix*
SSP_120106_179	LimitOrdersEscrow.sol	L205 - L205	⚠ *Pending Fix*
SSP_120106_180	LimitOrdersEscrow.sol	L305 - L305	⚠ *Pending Fix*
SSP_120106_181	LimitOrdersEscrow.sol	L500 - L500	⚠ *Pending Fix*
SSP_120106_182	LimitOrdersEscrow.sol	L513 - L513	⚠ *Pending Fix*
SSP_120106_183	LimitOrdersEscrow.sol	L530 - L530	⚠ *Pending Fix*
SSP_120106_184	LimitOrdersEscrow.sol	L532 - L532	⚠ *Pending Fix*

Issue Type

GAS OPTIMIZATION FOR THIS KEYWORD

S. No.	Severity	Detection Method	Instances
G006	● Gas	Automated	1

Description

Calling an external function internally, through the use of `this` keyword wastes gas overhead of calling an external functi on (100 gas).

Bug ID	File Location	Line No.	Action Taken
SSP_120106_230	LimitOrdersEscrow.sol	L278 - L278	⚠ *Pending Fix*

Issue Type

CHEAPER CONDITIONAL OPERATORS

S. No.	Severity	Detection Method	Instances
G007	● Gas	Automated	6

Description

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

Bug ID	File Location	Line No.	Action Taken
SSP_120106_12	LimitOrdersEscrow.sol	L368 - L368	⚠ *Pending Fix*
SSP_120106_12	LimitOrdersEscrow.sol	L683 - L683	⚠ *Pending Fix*
SSP_120106_13	LimitOrdersEscrow.sol	L373 - L373	⚠ *Pending Fix*
SSP_120106_14	LimitOrdersEscrow.sol	L560 - L560	⚠ *Pending Fix*
SSP_120106_15	LimitOrdersEscrow.sol	L565 - L565	⚠ *Pending Fix*
SSP_120106_16	LimitOrdersEscrow.sol	L630 - L630	⚠ *Pending Fix*

Issue Type

CHEAPER INEQUALITIES IN IF()

S. No.	Severity	Detection Method	Instances
G008	● Gas	Automated	9

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 `(>, <).`

Bug ID	File Location	Line No.	Action Taken
SSP_120106_73	LimitOrdersEscrow.sol	L326 - L326	⚠ *Pending Fix*
SSP_120106_74	LimitOrdersEscrow.sol	L337 - L337	⚠ *Pending Fix*
SSP_120106_75	LimitOrdersEscrow.sol	L340 - L340	⚠ *Pending Fix*
SSP_120106_76	LimitOrdersEscrow.sol	L368 - L368	⚠ *Pending Fix*
SSP_120106_76	LimitOrdersEscrow.sol	L683 - L683	⚠ *Pending Fix*
SSP_120106_77	LimitOrdersEscrow.sol	L373 - L373	⚠ *Pending Fix*
SSP_120106_78	LimitOrdersEscrow.sol	L536 - L536	⚠ *Pending Fix*
SSP_120106_79	LimitOrdersEscrow.sol	L560 - L560	⚠ *Pending Fix*
SSP_120106_80	LimitOrdersEscrow.sol	L565 - L565	⚠ *Pending Fix*

Issue Type

CHEAPER INEQUALITIES IN REQUIRE()

S. No.	Severity	Detection Method	Instances
G009	● Gas	Automated	4

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 `(>, <)`.

Bug ID	File Location	Line No.	Action Taken
SSP_120106_224	LimitOrdersEscrow.sol	L172 - L172	⚠ *Pending Fix*
SSP_120106_225	LimitOrdersEscrow.sol	L349 - L349	⚠ *Pending Fix*
SSP_120106_226	LimitOrdersEscrow.sol	L553 - L553	⚠ *Pending Fix*
SSP_120106_227	LimitOrdersEscrow.sol	L758 - L758	⚠ *Pending Fix*

Issue Type

DEFAULT INT VALUES ARE MANUALLY RESET

S. No.	Severity	Detection Method	Instances
G010	● Gas	Automated	3

Description

The contract is found to inefficiently reset integer variables to their default value of zero using manual assignment. In Solid ity, manually setting a variable to its default value does not free up storage space, leading to unnecessary gas consumptio n. Instead, using the `.delete` keyword can achieve the same result while also freeing up storage space on the Ethereum blockchain, resulting in gas cost savings.

Bug ID	File Location	Line No.	Action Taken
SSP_120106_60	LimitOrdersEscrow.sol	L288 - L288	⚠ *Pending Fix*
SSP_120106_61	LimitOrdersEscrow.sol	L741 - L741	⚠ *Pending Fix*
SSP_120106_62	LimitOrdersEscrow.sol	L783 - L783	⚠ *Pending Fix*

Issue Type

DEFINE CONSTRUCTOR AS PAYABLE

S. No.	Severity	Detection Method	Instances
G011	● Gas	Automated	1

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.

Bug ID	File Location	Line No.	Action Taken
SSP_120106_231	LimitOrdersEscrow.sol	L146 - L181	⚠ *Pending Fix*

Issue Type

EMIT USED IN LOOP

S. No.	Severity	Detection Method	Instances
G012	● Gas	Automated	1

Description

In Solidity, when a code emits an event inside of a loop, internally it performs a LOG operation N times, where N represents the number of iterations in the loop. This can lead to inflated gas costs and potentially impact the efficiency of the code.

Bug ID	File Location	Line No.	Action Taken
SSP_120106_59	LimitOrdersEscrow.sol	L400 - L400	⚠ *Pending Fix*

Issue Type

FUNCTIONS CAN BE IN-LINED

S. No.	Severity	Detection Method	Instances
G013	● Gas	Automated	8

Description

The internal function was called only once throughout the contract. Internal functions cost more gas due to additional `JUMP` instructions and stack operations.

Bug ID	File Location	Line No.	Action Taken
SSP_120106_110	LimitOrdersEscrow.sol	L507 - L539	⚠ *Pending Fix*
SSP_120106_111	LimitOrdersEscrow.sol	L541 - L588	⚠ *Pending Fix*
SSP_120106_112	LimitOrdersEscrow.sol	L594 - L601	⚠ *Pending Fix*
SSP_120106_113	LimitOrdersEscrow.sol	L604 - L618	⚠ *Pending Fix*
SSP_120106_114	LimitOrdersEscrow.sol	L620 - L644	⚠ *Pending Fix*
SSP_120106_115	LimitOrdersEscrow.sol	L646 - L653	⚠ *Pending Fix*
SSP_120106_116	LimitOrdersEscrow.sol	L655 - L664	⚠ *Pending Fix*
SSP_120106_117	LimitOrdersEscrow.sol	L676 - L689	⚠ *Pending Fix*

Issue Type

REVERTING FUNCTIONS CAN BE PAYABLE

S. No.	Severity	Detection Method	Instances
G014	● Gas	Automated	11

Description

If a function modifier such as `onlyOwner` is used, the function will revert if a normal user tries to pay the function. Markin g the function as payable will lower the gas cost for legitimate callers because the compiler will not include checks for whe ther a payment was provided.

Bug ID	File Location	Line No.	Action Taken
SSP_120106_331	LimitOrdersEscrow.sol	L714 - L717	⚠ *Pending Fix*
SSP_120106_332	LimitOrdersEscrow.sol	L722 - L725	⚠ *Pending Fix*
SSP_120106_333	LimitOrdersEscrow.sol	L727 - L729	⚠ *Pending Fix*
SSP_120106_334	LimitOrdersEscrow.sol	L740 - L743	⚠ *Pending Fix*
SSP_120106_335	LimitOrdersEscrow.sol	L749 - L752	⚠ *Pending Fix*
SSP_120106_336	LimitOrdersEscrow.sol	L757 - L761	⚠ *Pending Fix*
SSP_120106_337	LimitOrdersEscrow.sol	L768 - L771	⚠ *Pending Fix*
SSP_120106_338	LimitOrdersEscrow.sol	L773 - L779	⚠ *Pending Fix*
SSP_120106_339	LimitOrdersEscrow.sol	L781 - L786	⚠ *Pending Fix*
SSP_120106_340	LimitOrdersEscrow.sol	L788 - L791	⚠ *Pending Fix*
SSP_120106_341	LimitOrdersEscrow.sol	L793 - L796	⚠ *Pending Fix*

Issue Type

FUNCTION SHOULD RETURN STRUCT

S. No.	Severity	Detection Method	Instances
G015	● Gas	Automated	1

Description

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

Bug ID	File Location	Line No.	Action Taken
SSP_120106_229	LimitOrdersEscrow.sol	L486 - L495	⚠ *Pending Fix*

Issue Type

GAS INEFFICIENCY DUE TO MULTIPLE OPERANDS IN SINGLE IF/ELSEIF CONDITION

S. No.	Severity	Detection Method	Instances
G016	● Gas	Automated	8

Description

The contract is found to use multiple operands within a single `if` or `else if` statement, which can lead to unnecessary gas consumption due to the way the EVM evaluates compound boolean expressions. Each operand in a compound conditi on is evaluated even if the first condition fails, unless short-circuiting occurs, and the combined logic can result in more co mplex bytecode and higher gas usage compared to using nested `if` statements. This inefficiency is particularly relevant i n functions that are called frequently or within loops.

Bug ID	File Location	Line No.	Action Taken
SSP_120106_66	LimitOrdersEscrow.sol	L157 - L164	⚠ *Pending Fix*
SSP_120106_67	LimitOrdersEscrow.sol	L194 - L194	⚠ *Pending Fix*
SSP_120106_68	LimitOrdersEscrow.sol	L320 - L323	⚠ *Pending Fix*
SSP_120106_68	LimitOrdersEscrow.sol	L360 - L363	⚠ *Pending Fix*
SSP_120106_69	LimitOrdersEscrow.sol	L340 - L342	⚠ *Pending Fix*
SSP_120106_70	LimitOrdersEscrow.sol	L596 - L598	⚠ *Pending Fix*
SSP_120106_71	LimitOrdersEscrow.sol	L629 - L634	⚠ *Pending Fix*
SSP_120106_72	LimitOrdersEscrow.sol	L699 - L699	⚠ *Pending Fix*

Issue Type

GAS OPTIMIZATION FOR STATE VARIABLES

S. No.	Severity	Detection Method	Instances
G017	● Gas	Automated	1

Description

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

Bug ID	File Location	Line No.	Action Taken
SSP_120106_318	LimitOrdersEscrow.sol	L208 - L208	⚠ *Pending Fix*

Issue Type

GAS OPTIMIZATION IN INCREMENTS

S. No.	Severity	Detection Method	Instances
G018	● Gas	Automated	10

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, th us, making it a cheaper alternative.

Bug ID	File Location	Line No.	Action Taken
SSP_120106_99	LimitOrdersEscrow.sol	L178 - L178	⚠ *Pending Fix*
SSP_120106_100	LimitOrdersEscrow.sol	L276 - L276	⚠ *Pending Fix*
SSP_120106_101	LimitOrdersEscrow.sol	L381 - L381	⚠ *Pending Fix*
SSP_120106_102	LimitOrdersEscrow.sol	L421 - L421	⚠ *Pending Fix*
SSP_120106_103	LimitOrdersEscrow.sol	L440 - L440	⚠ *Pending Fix*
SSP_120106_104	LimitOrdersEscrow.sol	L458 - L458	⚠ *Pending Fix*
SSP_120106_104	LimitOrdersEscrow.sol	L466 - L466	⚠ *Pending Fix*
SSP_120106_105	LimitOrdersEscrow.sol	L480 - L480	⚠ *Pending Fix*
SSP_120106_106	LimitOrdersEscrow.sol	L774 - L774	⚠ *Pending Fix*
SSP_120106_107	LimitOrdersEscrow.sol	L782 - L782	⚠ *Pending Fix*

Issue Type

NAMED RETURN OF LOCAL VARIABLE SAVES GAS AS COMPARED TO RETURN STATEMENT

S. No.	Severity	Detection Method	Instances
G019	● Gas	Automated	2

Description

The function having a return type is found to be declaring a local variable for returning, which causes extra gas consumpti on. This inefficiency arises because creating and manipulating local variables requires additional computational steps and memory allocation.

Bug ID	File Location	Line No.	Action Taken
SSP_120106_310	LimitOrdersEscrow.sol	L455 - L472	⚠ *Pending Fix*
SSP_120106_311	LimitOrdersEscrow.sol	L478 - L484	⚠ *Pending Fix*

Issue Type

OPTIMIZING ADDRESS ID MAPPING

S. No.	Severity	Detection Method	Instances
G020	● Gas	Automated	4

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 t hey fit in the same storage slot.

Bug ID	File Location	Line No.	Action Taken
SSP_120106_186	LimitOrdersEscrow.sol	L77 - L77	⚠ *Pending Fix*
SSP_120106_187	LimitOrdersEscrow.sol	L87 - L87	⚠ *Pending Fix*
SSP_120106_188	LimitOrdersEscrow.sol	L90 - L90	⚠ *Pending Fix*
SSP_120106_189	LimitOrdersEscrow.sol	L91 - L91	⚠ *Pending Fix*

Issue Type

SIMILAR DATATYPES CAN BE PACKED TOGETHER

S. No.	Severity	Detection Method	Instances
G021	● Gas	Automated	1

Description

The contract is found to be using similar data types within a struct, leading to extra gas usage in Solidity. When a struct inc orporates fields with identical data types (such as multiple uint256 variables), failing to pack them efficiently can result in alignment gaps and increased gas consumption due to inefficient storage usage.

Bug ID	File Location	Line No.	Action Taken
SSP_120106_57	LimitOrdersEscrow.sol	L44 - L59	⚠ *Pending Fix*

Issue Type

SMALLER DATA TYPES COST MORE

S. No.	Severity	Detection Method	Instances
G022	● Gas	Automated	2

Description

Usage of smaller integer types such as `uint8`, `uint16`, `int8`, or `int16` in arithmetic operations incur additional gas costs compared to the default `uint` and `int` types, which are typically `uint256` and `int256` respectively.

Bug ID	File Location	Line No.	Action Taken
SSP_120106_1	LimitOrdersEscrow.sol	L614 - L614	⚠ *Pending Fix*
SSP_120106_2	LimitOrdersEscrow.sol	L615 - L615	⚠ *Pending Fix*

Issue Type

SPLITTING REQUIRE STATEMENTS

S. No.	Severity	Detection Method	Instances
G023	● Gas	Automated	1

Description

Require statements when combined using operators in a single statement usually lead to a larger deployment gas cost but with each runtime calls, the whole thing ends up being cheaper by some gas units.

Bug ID	File Location	Line No.	Action Taken
SSP_120106_325	LimitOrdersEscrow.sol	L268 - L273	⚠ *Pending Fix*

Issue Type

SPLITTING REVERT STATEMENTS

S. No.	Severity	Detection Method	Instances
G024	● Gas	Automated	2

Description

The contract is using multiple conditions in a single `if` statement followed by a revert. This costs some extra gas.

Bug ID	File Location	Line No.	Action Taken
SSP_120106_108	LimitOrdersEscrow.sol	L157 - L164	⚠ *Pending Fix*
SSP_120106_109	LimitOrdersEscrow.sol	L340 - L342	⚠ *Pending Fix*

Issue Type

STORAGE VARIABLE CACHING IN MEMORY

S. No.	Severity	Detection Method	Instances
G025	● Gas	Automated	8

Description

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

Bug ID	File Location	Line No.	Action Taken
SSP_120106_161	LimitOrdersEscrow.sol	L184 - L235	⚠ *Pending Fix*
SSP_120106_162	LimitOrdersEscrow.sol	L409 - L453	⚠ *Pending Fix*
SSP_120106_163	LimitOrdersEscrow.sol	L455 - L472	⚠ *Pending Fix*
SSP_120106_164	LimitOrdersEscrow.sol	L507 - L539	⚠ *Pending Fix*
SSP_120106_165	LimitOrdersEscrow.sol	L541 - L588	⚠ *Pending Fix*
SSP_120106_165	LimitOrdersEscrow.sol	L541 - L588	⚠ *Pending Fix*
SSP_120106_166	LimitOrdersEscrow.sol	L594 - L601	⚠ *Pending Fix*
SSP_120106_167	LimitOrdersEscrow.sol	L655 - L664	⚠ *Pending Fix*

Issue Type

STORING STORAGE VARIABLES IN MEMORY

S. No.	Severity	Detection Method	Instances
G026	● Gas	Automated	3

Description

Whenever a struct, array, or a mapping is stored and copied to a memory variable, each member is read from the storage a nd then copied. This becomes expensive. This could easily be optimized by using the storage keyword which just stores a pointer to the storage instead, making the whole process a lot cheaper.

Bug ID	File Location	Line No.	Action Taken
SSP_120106_63	LimitOrdersEscrow.sol	L417 - L417	⚠ *Pending Fix*
SSP_120106_64	LimitOrdersEscrow.sol	L456 - L456	⚠ *Pending Fix*
SSP_120106_65	LimitOrdersEscrow.sol	L520 - L525	⚠ *Pending Fix*

Issue Type

UNNECESSARY CHECKED ARITHMETIC IN LOOP

S. No.	Severity	Detection Method	Instances
G027	● Gas	Automated	14

Description

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

Bug ID	File Location	Line No.	Action Taken
SSP_120106_25	LimitOrdersEscrow.sol	L178 - L178	⚠ *Pending Fix*
SSP_120106_26	LimitOrdersEscrow.sol	L276 - L276	⚠ *Pending Fix*
SSP_120106_27	LimitOrdersEscrow.sol	L381 - L381	⚠ *Pending Fix*
SSP_120106_28	LimitOrdersEscrow.sol	L421 - L421	⚠ *Pending Fix*
SSP_120106_29	LimitOrdersEscrow.sol	L433 - L433	⚠ *Pending Fix*
SSP_120106_30	LimitOrdersEscrow.sol	L440 - L440	⚠ *Pending Fix*
SSP_120106_31	LimitOrdersEscrow.sol	L450 - L450	⚠ *Pending Fix*
SSP_120106_32	LimitOrdersEscrow.sol	L458 - L458	⚠ *Pending Fix*
SSP_120106_32	LimitOrdersEscrow.sol	L466 - L466	⚠ *Pending Fix*
SSP_120106_33	LimitOrdersEscrow.sol	L460 - L460	⚠ *Pending Fix*
SSP_120106_34	LimitOrdersEscrow.sol	L468 - L468	⚠ *Pending Fix*
SSP_120106_35	LimitOrdersEscrow.sol	L480 - L480	⚠ *Pending Fix*
SSP_120106_36	LimitOrdersEscrow.sol	L774 - L774	⚠ *Pending Fix*

Bug ID	File Location	Line No.	Action Taken
SSP_120106_37	LimitOrdersEscrow.sol	L782 - L782	⚠ *Pending Fix*

05. Scan History

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

No	Date	Security Score	Scan Overview
1.	2026-01-06	**90.46**	● 0 ● 0 ● 2 ● 7 ● 180 ● 127

06. 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.