SolidityScan | Powered by CRED SHIELDS

# Security Assessment Report

## Empx DEX Aggregator

8 Jan 2026

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

# Table of Contents

| |
|---|
| BLOCK VALUES AS A PROXY FOR TIME |
| MISSING @AUTHOR IN NATSPEC COMMENTS FOR CONTRACT DECLARATION |
| MISSING @DEV IN NATSPEC COMMENTS FOR CONTRACT DECLARATION |
| MISSING @DEV IN NATSPEC COMMENTS FOR FUNCTIONS |
| 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 |
| UNUSED RECEIVE FALLBACK |
| ARRAY LENGTH CACHING |
| AVOID RE-STORING VALUES |
| AVOID ZERO-TO-ONE STORAGE WRITES |
| CACHE ADDRESS(THIS) WHEN USED MORE THAN ONCE |
| CHEAPER CONDITIONAL OPERATORS |
| CHEAPER INEQUALITIES IN IF() |
| CHEAPER INEQUALITIES IN REQUIRE() |
| DEFAULT INT VALUES ARE MANUALLY RESET |
| DEFINE CONSTRUCTOR AS PAYABLE |
| FUNCTIONS CAN BE IN-LINED |
| GAS INEFFICIENCY DUE TO MULTIPLE OPERANDS IN SINGLE IF/ELSEIF CONDITION |
| GAS OPTIMIZATION IN INCREMENTS |
| LONG REQUIRE/REVERT STRINGS |
| NAMED RETURN OF LOCAL VARIABLE SAVES GAS AS COMPARED TO RETURN STATEMENT |
| PUBLIC CONSTANTS CAN BE PRIVATE |

| |
|---|
| STORAGE VARIABLE CACHING IN MEMORY |
| UNNECESSARY CHECKED ARITHMETIC IN LOOP |
| UNNECESSARY DEFAULT VALUE INITIALIZATION |
| UNUSED IMPORTS |

## 05 Scan History

## 06 Disclaimer

# 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 DEX Aggregator**

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 DEX Aggregator 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:

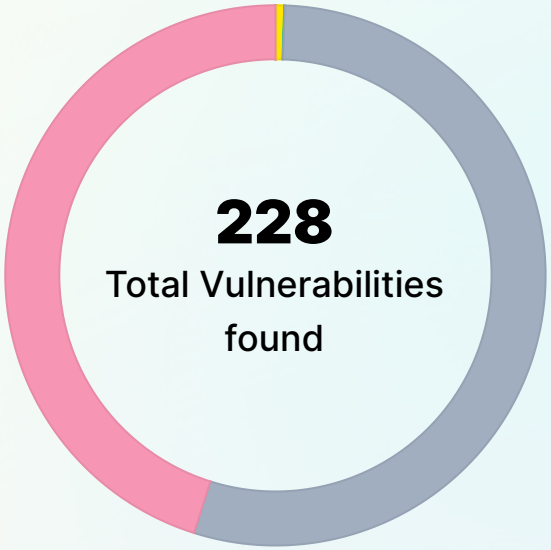
Various common and uncommon attack vectors will be investigated to ensure that the smart contracts are secure from malicious actors. The scanner modules find and flag issues related to Gas optimizations that help in reducing the overall Gas cost It scans and evaluates the codebase against industry best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

# 03. Findings Summary

**Empx DEX Aggregator**
File Scan

| Security Score | Scan duration | Lines of code |
|---|---|---|
| **91.41**/100 | **442 secs** | **426** |




| 0 | 0 | 1 | 0 | 124 | 103 |
|---|---|---|---|---|---|
| Crit | High | Med | Low | Info | Gas |

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

# ACTION TAKEN

| 0 | 34 | 0 | 233 |
|---|---|---|---|
| *Fixed* | *False Positive* | *Won't Fix* | *Pending Fix* |

| S. No. | Severity | Bug Type | Instances | Detection Method | Status |
|---|---|---|---|---|---|
| H001 | High | IMPROPER VALIDATION IN REQUIRE/ASSERT STATEMENTS | 2 | Automated | *False Positive* |
| H002 | High | REENTRANCY | 2 | Automated | *False Positive* |
| H003 | High | TRANSFER INSIDE A LOOP | 2 | Automated | *False Positive* |
| H004 | High | ZERO AMOUNT SWAPS NOT REJECTED | 2 | SolidityScan AI | *False Positive* |
| M001 | Medium | ERC20 NON STANDARD BEHAVIOR | 1 | SolidityScan AI | *False Positive* |
| M002 | Medium | UNCHECKED ARRAY LENGTH | 2 | Automated | *False Positive* |
| M003 | Medium | LIMITATIONS OF SOLIDITY'S TRY-CATCH IN EXTERNAL CALLS | 2 | Automated | *False Positive* |
| M004 | Medium | PRECISION LOSS DURING DIVISION BY LARGE NUMBERS | 3 | Automated | *False Positive* |
| M005 | Medium | DEPRECATED SAFEAPPROVE | 1 | Automated | *False Positive* |
| M006 | Medium | SWAP LIMIT MISSING | 1 | SolidityScan AI | *False Positive* |
| L001 | Low | EXECUTESPLITSWAP DOES NOT ENFORCE THAT EACH PATH STARTS WITH THE SAME TOKENIN AS THE ONE BEING DISTRIBUTED | 1 | SolidityScan AI | *False Positive* |
| L002 | Low | SLIPPAGE CHECK IN EXECUTECONVERGESWAP DEPENDS ON ERC20.BALANCEOF() READINGS OF ARBITRARY TOKENOUT, WHICH CAN BE SPOOFED BY MALICIOUS TOKENS | 1 | SolidityScan AI | *False Positive* |
| L003 | Low | APPROVING MAXIMUM VALUE | 1 | Automated | *False Positive* |

| S. No. | Severity | Bug Type | Instances | Detection Method | Status |
|---|---|---|---|---|---|
| L005 | Low | MISSING EVENTS | 5 | Automated | *False Positive* |
| L006 | Low | MISSING ZERO ADDRESS VALIDATION | 3 | Automated | *False Positive* |
| L007 | Low | OUTDATED COMPILER VERSION | 1 | Automated | *False Positive* |
| I001 | Informational | AVOID ARITHMETIC DIRECTLY WITHIN ARRAY INDICES | 16 | Automated | *Pending Fix* |
| I002 | Informational | BLOCK VALUES AS A PROXY FOR TIME | 2 | Automated | *Pending Fix* |
| I003 | Informational | MISSING @AUTHOR IN NATSPEC COMMENTS FOR CONTRACT DECLARATION | 1 | Automated | *Pending Fix* |
| I004 | Informational | MISSING @DEV IN NATSPEC COMMENTS FOR CONTRACT DECLARATION | 1 | Automated | *Pending Fix* |
| I005 | Informational | MISSING @DEV IN NATSPEC COMMENTS FOR FUNCTIONS | 24 | Automated | *Pending Fix* |
| I006 | Informational | MISSING @INHERITDOC ON OVERRIDE FUNCTIONS | 11 | Automated | *Pending Fix* |
| I007 | Informational | MISSING NATSPEC COMMENTS IN SCOPE BLOCKS | 35 | Automated | *Pending Fix* |
| I008 | Informational | MISSING NATSPEC DESCRIPTIONS FOR PUBLIC VARIABLE DECLARATIONS | 8 | Automated | *Pending Fix* |
| I009 | Informational | MISSING @NOTICE IN NATSPEC COMMENTS FOR CONSTRUCTORS | 1 | Automated | *Pending Fix* |
| I010 | Informational | MISSING @NOTICE IN NATSPEC COMMENTS FOR FUNCTIONS | 24 | Automated | *Pending Fix* |
| I011 | Informational | NAME MAPPING PARAMETERS | 1 | Automated | *Pending Fix* |
| I012 | Informational | UNUSED RECEIVE FALLBACK | 1 | Automated | *Pending Fix* |
| G001 | Gas | ARRAY LENGTH CACHING | 11 | Automated | *Pending Fix* |

| S. No. | Severity | Bug Type | Instances | Detection Method | Status |
|---|---|---|---|---|---|
| G002 | Gas | AVOID RE-STORING VALUES | 4 | Automated | *Pending Fix* |
| G003 | Gas | AVOID ZERO-TO-ONE STORAGE WRITES | 1 | Automated | *Pending Fix* |
| G004 | Gas | CACHE ADDRESS(THIS) WHEN USED MORE THAN ONCE | 14 | Automated | *Pending Fix* |
| G005 | Gas | CHEAPER CONDITIONAL OPERATORS | 13 | Automated | *Pending Fix* |
| G006 | Gas | CHEAPER INEQUALITIES IN IF() | 6 | Automated | *Pending Fix* |
| G007 | Gas | CHEAPER INEQUALITIES IN REQUIRE() | 8 | Automated | *Pending Fix* |
| G008 | Gas | DEFAULT INT VALUES ARE MANUALLY RESET | 1 | Automated | *Pending Fix* |
| G009 | Gas | DEFINE CONSTRUCTOR AS PAYABLE | 1 | Automated | *Pending Fix* |
| G010 | Gas | FUNCTIONS CAN BE IN-LINED | 2 | Automated | *Pending Fix* |
| G011 | Gas | GAS INEFFICIENCY DUE TO MULTIPLE OPERANDS IN SINGLE IF/ELSEIF CONDITION | 3 | Automated | *Pending Fix* |
| G012 | Gas | GAS OPTIMIZATION IN INCREMENTS | 11 | Automated | *Pending Fix* |
| G013 | Gas | LONG REQUIRE/REVERT STRINGS | 5 | Automated | *Pending Fix* |
| G014 | Gas | NAMED RETURN OF LOCAL VARIABLE SAVES GAS AS COMPARED TO RETURN STATEMENT | 4 | Automated | *Pending Fix* |
| G015 | Gas | PUBLIC CONSTANTS CAN BE PRIVATE | 2 | Automated | *Pending Fix* |
| G016 | Gas | STORAGE VARIABLE CACHING IN MEMORY | 6 | Automated | *Pending Fix* |
| G017 | Gas | UNNECESSARY CHECKED ARITHMETIC IN LOOP | 11 | Automated | *Pending Fix* |
| G018 | Gas | UNNECESSARY DEFAULT VALUE INITIALIZATION | 1 | Automated | *Pending Fix* |
| G019 | Gas | UNUSED IMPORTS | 1 | Automated | *Pending Fix* |

# 04. Vulnerability Details

Issue Type

**IMPROPER VALIDATION IN REQUIRE/ASSERT STATEMENTS**

| S. No. | Severity | Detection Method | Instances |
|---|---|---|---|
| H001 | High | Automated | 2 |

| Bug ID | File Location | Line No. | Action Taken |
|---|---|---|---|
| SSP_120277_15 | -- | -- | *False Positive* |
| SSP_120277_16 | -- | -- | *False Positive* |

**Upgrade your Plan to view the full report**

**2 High Issues Found**

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

Upgrade

Issue Type

**ERC20 NON STANDARD BEHAVIOR**

| S. No. | Severity | Detection Method | Instances |
|---|---|---|---|
| M001 | Medium | SolidityScan AI | 1 |

| Bug ID | File Location | Line No. | Action Taken |
|---|---|---|---|
| SSP_120277_265 | -- | -- | *False Positive* |

**Upgrade your Plan to view the full report**

**1 Medium Issues Found**

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

Upgrade

Issue Type

**EXECUTESPLITSWAP DOES NOT ENFORCE THAT EACH PATH STARTS WITH THE SAME TOKENIN AS THE ONE BEING DISTRIBUTED**

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

| Bug ID | File Location | Line No. | Action Taken |
|---|---|---|---|
| SSP_120277_266 | -- | -- | *False Positive* |

**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

## AVOID ARITHMETIC DIRECTLY WITHIN ARRAY INDICES

| S. No. | Severity | Detection Method | Instances |
|---|---|---|---|
| I001 | Informational | Automated | 16 |

| Bug ID | File Location | Line No. | Action Taken |
|---|---|---|---|
| SSP_120277_58 | -- | -- | Pending Fix |
| SSP_120277_59 | -- | -- | Pending Fix |
| SSP_120277_60 | -- | -- | Pending Fix |
| SSP_120277_61 | -- | -- | Pending Fix |
| SSP_120277_62 | -- | -- | Pending Fix |
| SS | | | |
| SS | | | |
| SS | | | |
| SS | | | |
| SS | | | |
| SSP_120277_68 | -- | -- | Pending Fix |
| SSP_120277_69 | -- | -- | Pending Fix |
| SSP_120277_70 | -- | -- | Pending Fix |

| Bug ID | File Location | Line No. | Action Taken |
|---|---|---|---|
| SSP_120277_71 | -- | -- | ⚠ *Pending Fix* |
| SSP_120277_72 | -- | -- | ⚠ *Pending Fix* |
| SSP_120277_73 | -- | -- | ⚠ *Pending Fix* |

### Upgrade your Plan to view the full report

**16 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 | 11 |

### Description

During each iteration of the loop, reading the length of the array uses more gas than is necessary. In the most favora ble 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.

| Bug ID | File Location | Line No. | Action Taken |
|---|---|---|---|
| SSP_120277_38 | EmpsealRouter.sol | L95 - L97 | *Pending Fix* |
| SSP_120277_39 | EmpsealRouter.sol | L102 - L104 | *Pending Fix* |
| SSP_120277_40 | EmpsealRouter.sol | L196 - L224 | *Pending Fix* |
| SSP_120277_41 | EmpsealRouter.sol | L299 - L345 | *Pending Fix* |
| SSP_120277_42 | EmpsealRouter.sol | L317 - L326 | *Pending Fix* |
| SSP_120277_43 | EmpsealRouter.sol | L330 - L342 | *Pending Fix* |
| SSP_120277_44 | EmpsealRouter.sol | L371 - L373 | *Pending Fix* |
| SSP_120277_45 | EmpsealRouter.sol | L377 - L383 | *Pending Fix* |
| SSP_120277_46 | EmpsealRouter.sol | L387 - L399 | *Pending Fix* |
| SSP_120277_47 | EmpsealRouter.sol | L458 - L464 | *Pending Fix* |

| Bug ID | File Location | Line No. | Action Taken |
| --- | --- | --- | --- |
| SSP_120277_48 | EmpsealRouter.sol | L475 - L482 | ⚠ *Pending Fix* |

Issue Type

## AVOID RE-STORING VALUES

| S. No. | Severity | Detection Method | Instances |
|---|---|---|---|
| G002 | Gas | Automated | 4 |

### 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 could be avoided. If the old value and the new value are the same, not updating the storage would avoid this cost and could instead incur a `Gcoldsload` (2100 gas) or a `Gwarmaccess` (100 gas), potentially saving gas.

| Bug ID | File Location | Line No. | Action Taken |
|---|---|---|---|
| SSP_120277_89 | EmpsealRouter.sol | L88 - L91 | *Pending Fix* |
| SSP_120277_90 | EmpsealRouter.sol | L93 - L106 | *Pending Fix* |
| SSP_120277_91 | EmpsealRouter.sol | L108 - L112 | *Pending Fix* |
| SSP_120277_92 | EmpsealRouter.sol | L114 - L118 | *Pending Fix* |

Issue Type

## AVOID ZERO-TO-ONE STORAGE WRITES

| S. No. | Severity | Detection Method | Instances |
|---|---|---|---|
| G003 | Gas | Automated | 1 |

### 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 5,000 gas.

| Bug ID | File Location | Line No. | Action Taken |
|---|---|---|---|
| SSP_120277_179 | EmpsealRouter.sol | L110 - L110 | ⚠ *Pending Fix* |

Issue Type

## CACHE ADDRESS(THIS) WHEN USED MORE THAN ONCE

| S. No. | Severity | Detection Method | Instances |
|---|---|---|---|
| G004 | Gas | Automated | 14 |

### 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_120277_97 | EmpsealRouter.sol | L81 - L81 | *Pending Fix* |
| SSP_120277_98 | EmpsealRouter.sol | L137 - L137 | *Pending Fix* |
| SSP_120277_99 | EmpsealRouter.sol | L142 - L142 | *Pending Fix* |
| SSP_120277_100 | EmpsealRouter.sol | L172 - L172 | *Pending Fix* |
| SSP_120277_100 | EmpsealRouter.sol | L278 - L278 | *Pending Fix* |
| SSP_120277_101 | EmpsealRouter.sol | L186 - L186 | *Pending Fix* |
| SSP_120277_102 | EmpsealRouter.sol | L187 - L187 | *Pending Fix* |
| SSP_120277_103 | EmpsealRouter.sol | L191 - L191 | *Pending Fix* |
| SSP_120277_104 | EmpsealRouter.sol | L222 - L222 | *Pending Fix* |
| SSP_120277_105 | EmpsealRouter.sol | L226 - L226 | *Pending Fix* |
| SSP_120277_106 | EmpsealRouter.sol | L292 - L292 | *Pending Fix* |

| Bug ID | File Location | Line No. | Action Taken |
| --- | --- | --- | --- |
| SSP_120277_107 | EmpsealRouter.sol | L293 - L293 | ⚠ *Pending Fix* |
| SSP_120277_108 | EmpsealRouter.sol | L421 - L421 | ⚠ *Pending Fix* |
| SSP_120277_109 | EmpsealRouter.sol | L430 - L430 | ⚠ *Pending Fix* |

Issue Type

## CHEAPER CONDITIONAL OPERATORS

| S. No. | Severity | Detection Method | Instances |
|---|---|---|---|
| G005 | Gas | Automated | 13 |

### 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_120277_74 | EmpsealRouter.sol | L162 - L162 | *Pending Fix* |
| SSP_120277_75 | EmpsealRouter.sol | L163 - L163 | *Pending Fix* |
| SSP_120277_76 | EmpsealRouter.sol | L228 - L228 | *Pending Fix* |
| SSP_120277_77 | EmpsealRouter.sol | L268 - L268 | *Pending Fix* |
| SSP_120277_78 | EmpsealRouter.sol | L358 - L358 | *Pending Fix* |
| SSP_120277_79 | EmpsealRouter.sol | L82 - L82 | *Pending Fix* |
| SSP_120277_80 | EmpsealRouter.sol | L169 - L169 | *Pending Fix* |
| SSP_120277_81 | EmpsealRouter.sol | L177 - L177 | *Pending Fix* |
| SSP_120277_82 | EmpsealRouter.sol | L179 - L179 | *Pending Fix* |
| SSP_120277_83 | EmpsealRouter.sol | L275 - L275 | *Pending Fix* |
| SSP_120277_84 | EmpsealRouter.sol | L284 - L284 | *Pending Fix* |

| Bug ID | File Location | Line No. | Action Taken |
| --- | --- | --- | --- |
| SSP_120277_85 | EmpsealRouter.sol | L286 - L286 | ⚠ *Pending Fix* |
| SSP_120277_86 | EmpsealRouter.sol | L363 - L363 | ⚠ *Pending Fix* |

Issue Type

## CHEAPER INEQUALITIES IN IF()

| S. No. | Severity | Detection Method | Instances |
|---|---|---|---|
| G006 | Gas | Automated | 6 |

### 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_120277_30 | EmpsealRouter.sol | L82 - L82 | *Pending Fix* |
| SSP_120277_31 | EmpsealRouter.sol | L169 - L169 | *Pending Fix* |
| SSP_120277_32 | EmpsealRouter.sol | L177 - L177 | *Pending Fix* |
| SSP_120277_33 | EmpsealRouter.sol | L275 - L275 | *Pending Fix* |
| SSP_120277_34 | EmpsealRouter.sol | L284 - L284 | *Pending Fix* |
| SSP_120277_35 | EmpsealRouter.sol | L363 - L363 | *Pending Fix* |

Issue Type

## CHEAPER INEQUALITIES IN REQUIRE()

| S. No. | Severity | Detection Method | Instances |
|---|---|---|---|
| G007 | Gas | Automated | 8 |

### 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_120277_7 | EmpsealRouter.sol | L109 - L109 | *Pending Fix* |
| SSP_120277_8 | EmpsealRouter.sol | L123 - L123 | *Pending Fix* |
| SSP_120277_9 | EmpsealRouter.sol | L124 - L124 | *Pending Fix* |
| SSP_120277_10 | EmpsealRouter.sol | L160 - L160 | *Pending Fix* |
| SSP_120277_11 | EmpsealRouter.sol | L253 - L253 | *Pending Fix* |
| SSP_120277_12 | EmpsealRouter.sol | L267 - L267 | *Pending Fix* |
| SSP_120277_13 | EmpsealRouter.sol | L347 - L347 | *Pending Fix* |
| SSP_120277_14 | EmpsealRouter.sol | L385 - L385 | *Pending Fix* |

Issue Type

## DEFAULT INT VALUES ARE MANUALLY RESET

| S. No. | Severity | Detection Method | Instances |
|---|---|---|---|
| G008 | Gas | Automated | 1 |

### Description

The contract is found to inefficiently reset integer variables to their default value of zero using manual assignment. In Solidity, manually setting a variable to its default value does not free up storage space, leading to unnecessary gas c onsumption. 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_120277_217 | EmpsealRouter.sol | L96 - L96 | *Pending Fix* |

Issue Type

## DEFINE CONSTRUCTOR AS PAYABLE

| S. No. | Severity | Detection Method | Instances |
|---|---|---|---|
| G009 | 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 deployme
nt.

| Bug ID | File Location | Line No. | Action Taken |
|---|---|---|---|
| SSP_120277_94 | EmpsealRouter.sol | L63 - L75 | *Pending Fix* |

Issue Type

**FUNCTIONS CAN BE IN-LINED**

| S. No. | Severity | Detection Method | Instances |
|---|---|---|---|
| G010 | Gas | Automated | 2 |

### Description

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

| Bug ID | File Location | Line No. | Action Taken |
|---|---|---|---|
| SSP_120277_49 | EmpsealRouter.sol | L132 - L134 | *Pending Fix* |
| SSP_120277_50 | EmpsealRouter.sol | L141 - L151 | *Pending Fix* |

Issue Type

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

| S. No. | Severity | Detection Method | Instances |
|---|---|---|---|
| G011 | Gas | Automated | 3 |

### 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 condition is evaluated even if the first condition fails, unless short-circuiting occurs, and the combined logic can result in more complex bytecode and higher gas usage compared to using nested `if` statements. This inefficiency is particularly relevant in functions that are called frequently or within loops.

| Bug ID | File Location | Line No. | Action Taken |
|---|---|---|---|
| SSP_120277_161 | EmpsealRouter.sol | L169 - L173 | *Pending Fix* |
| SSP_120277_162 | EmpsealRouter.sol | L275 - L279 | *Pending Fix* |
| SSP_120277_163 | EmpsealRouter.sol | L363 - L368 | *Pending Fix* |

Issue Type

## GAS OPTIMIZATION IN INCREMENTS

| S. No. | Severity | Detection Method | Instances |
|---|---|---|---|
| G012 | Gas | Automated | 11 |

### Description

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

| Bug ID | File Location | Line No. | Action Taken |
|---|---|---|---|
| SSP_120277_206 | EmpsealRouter.sol | L95 - L95 | *Pending Fix* |
| SSP_120277_207 | EmpsealRouter.sol | L102 - L102 | *Pending Fix* |
| SSP_120277_208 | EmpsealRouter.sol | L196 - L196 | *Pending Fix* |
| SSP_120277_209 | EmpsealRouter.sol | L299 - L299 | *Pending Fix* |
| SSP_120277_210 | EmpsealRouter.sol | L317 - L317 | *Pending Fix* |
| SSP_120277_211 | EmpsealRouter.sol | L330 - L330 | *Pending Fix* |
| SSP_120277_212 | EmpsealRouter.sol | L371 - L371 | *Pending Fix* |
| SSP_120277_213 | EmpsealRouter.sol | L377 - L377 | *Pending Fix* |
| SSP_120277_214 | EmpsealRouter.sol | L387 - L387 | *Pending Fix* |
| SSP_120277_215 | EmpsealRouter.sol | L458 - L458 | *Pending Fix* |
| SSP_120277_216 | EmpsealRouter.sol | L475 - L475 | *Pending Fix* |

Issue Type

## LONG REQUIRE/REVERT STRINGS

| S. No. | Severity | Detection Method | Instances |
|---|---|---|---|
| G013 | Gas | Automated | 5 |

### 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.

| Bug ID | File Location | Line No. | Action Taken |
|---|---|---|---|
| SSP_120277_2 | EmpsealRouter.sol | L109 - L109 | *Pending Fix* |
| SSP_120277_3 | EmpsealRouter.sol | L115 - L115 | *Pending Fix* |
| SSP_120277_4 | EmpsealRouter.sol | L418 - L418 | *Pending Fix* |
| SSP_120277_5 | EmpsealRouter.sol | L419 - L419 | *Pending Fix* |
| SSP_120277_6 | EmpsealRouter.sol | L429 - L429 | *Pending Fix* |

Issue Type

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

| S. No. | Severity | Detection Method | Instances |
|---|---|---|---|
| G014 | Gas | Automated | 4 |

### Description

The function having a return type is found to be declaring a local variable for returning, which causes extra gas cons umption. 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_120277_197 | EmpsealRouter.sol | L153 - L257 | *Pending Fix* |
| SSP_120277_198 | EmpsealRouter.sol | L443 - L452 | *Pending Fix* |
| SSP_120277_199 | EmpsealRouter.sol | L454 - L466 | *Pending Fix* |
| SSP_120277_200 | EmpsealRouter.sol | L468 - L484 | *Pending Fix* |

Issue Type

**PUBLIC CONSTANTS CAN BE PRIVATE**

| S. No. | Severity | Detection Method | Instances |
|---|---|---|---|
| G015 | Gas | Automated | 2 |

## 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.

| Bug ID | File Location | Line No. | Action Taken |
|---|---|---|---|
| SSP_120277_180 | EmpsealRouter.sol | L26 - L26 | *Pending Fix* |
| SSP_120277_181 | EmpsealRouter.sol | L27 - L27 | *Pending Fix* |

Issue Type

## STORAGE VARIABLE CACHING IN MEMORY

| S. No. | Severity | Detection Method | Instances |
|---|---|---|---|
| G016 | Gas | Automated | 6 |

### 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_120277_172 | EmpsealRouter.sol | L93 - L106 | *Pending Fix* |
| SSP_120277_173 | EmpsealRouter.sol | L122 - L126 | *Pending Fix* |
| SSP_120277_174 | EmpsealRouter.sol | L153 - L257 | *Pending Fix* |
| SSP_120277_174 | EmpsealRouter.sol | L153 - L257 | *Pending Fix* |
| SSP_120277_175 | EmpsealRouter.sol | L259 - L349 | *Pending Fix* |
| SSP_120277_176 | EmpsealRouter.sol | L454 - L466 | *Pending Fix* |

Issue Type

## UNNECESSARY CHECKED ARITHMETIC IN LOOP

| S. No. | Severity | Detection Method | Instances |
|---|---|---|---|
| G017 | Gas | Automated | 11 |

### Description

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

| Bug ID | File Location | Line No. | Action Taken |
|---|---|---|---|
| SSP_120277_19 | EmpsealRouter.sol | L95 - L95 | *Pending Fix* |
| SSP_120277_20 | EmpsealRouter.sol | L102 - L102 | *Pending Fix* |
| SSP_120277_21 | EmpsealRouter.sol | L196 - L196 | *Pending Fix* |
| SSP_120277_22 | EmpsealRouter.sol | L299 - L299 | *Pending Fix* |
| SSP_120277_23 | EmpsealRouter.sol | L317 - L317 | *Pending Fix* |
| SSP_120277_24 | EmpsealRouter.sol | L330 - L330 | *Pending Fix* |
| SSP_120277_25 | EmpsealRouter.sol | L371 - L371 | *Pending Fix* |
| SSP_120277_26 | EmpsealRouter.sol | L377 - L377 | *Pending Fix* |
| SSP_120277_27 | EmpsealRouter.sol | L387 - L387 | *Pending Fix* |
| SSP_120277_28 | EmpsealRouter.sol | L458 - L458 | *Pending Fix* |
| SSP_120277_29 | EmpsealRouter.sol | L475 - L475 | *Pending Fix* |

Issue Type

## UNNECESSARY DEFAULT VALUE INITIALIZATION

| S. No. | Severity | Detection Method | Instances |
|---|---|---|---|
| G018 | Gas | Automated | 1 |

### Description

The contract was found to be initializing the value of the variable to it’s default value.
This is redundant and not required.

| Bug ID | File Location | Line No. | Action Taken |
|---|---|---|---|
| SSP_120277_93 | EmpsealRouter.sol | L30 - L30 | *Pending Fix* |

Issue Type

**UNUSED IMPORTS**

| S. No. | Severity | Detection Method | Instances |
|---|---|---|---|
| **G019** | Gas | Automated | 1 |

## Description

Solidity is a Gas-constrained language. Having unused code or import statements incurs extra gas usage when deploying the contract.

| Bug ID | File Location | Line No. | Action Taken |
|---|---|---|---|
| SSP_120277_160 | EmpsealRouter.sol | L10 - L10 | ⚠ *Pending Fix* |

# 05. Scan History

Critical High Medium Low Informational Gas

| No | Date | Security Score | Scan Overview |
|---|---|---|---|
| 1. | 2026-01-08 | **91.41** | 0 0 1 0 124 103 |

# 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.