SolidityScan | Powered by CRED SHIELDS

Security Assessment Report

Empx Via Bridge

13 Jan 2026

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

Table of Contents

ADDING A RETURN STATEMENT WHEN THE FUNCTION DEFINES A NAMED RETURN VARIABLE IS REDUNDANT
HARD-CODED GAS LIMITS
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
REVERT STATEMENTS WITHIN EXTERNAL AND PUBLIC FUNCTIONS CAN BE USED TO PERFORM DOS ATTACKS
UNNAMED FUNCTION PARAMETERS
USE CALL INSTEAD OF TRANSFER OR SEND
ABI ENCODE IS LESS EFFICIENT THAN ABI ENCODEPACKED
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()
DEFINE CONSTRUCTOR AS PAYABLE
REVERTING FUNCTIONS CAN BE PAYABLE
FUNCTION SHOULD RETURN STRUCT
GAS INEFFICIENCY DUE TO MULTIPLE OPERANDS IN SINGLE IF/ELSEIF CONDITION

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 Via Bridge

Uploaded Solidity File(s)

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

Security Score is AVERAGE

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 Via Bridge 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:

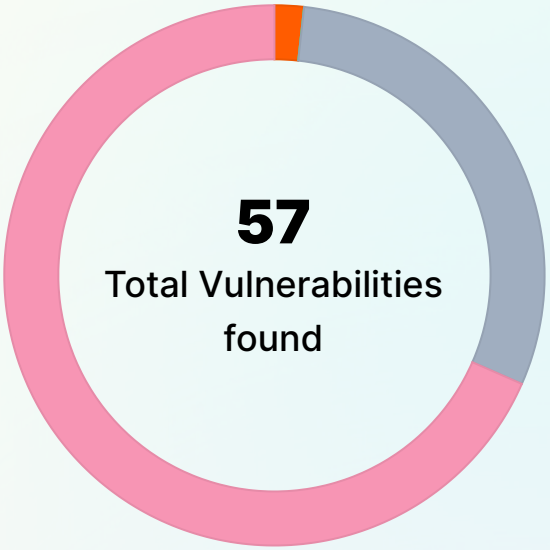
Various common and uncommon attack vectors will be investigated to ensure that the smart contracts are secure from malicious actors. The scanner modules find and flag issues related to Gas optimizations that help in reducing the overall Gas cost It scans and evaluates the codebase against industry best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

03. Findings Summary

Empx Via Bridge
File Scan

Security Score	Scan duration	Lines of code
72.77/100	305 secs	212



0	1	0	0	17	39
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	98	0	58
Fixed	False Positive	Won't Fix	Pending Fix

S. No.	Severity	Bug Type	Instances	Detection Method	Status
H001	High	FEE ON TRANSFER TOKEN INCOMPATIBILITY	1	SolidityScan AI	False Positive
H002	High	L2 MESSAGE REPLAY	1	SolidityScan AI	False Positive
H003	High	USE OF TX.GASPRICE	2	Automated	False Positive
H004	High	STALE UNLIMITED ALLOWANCE ON FEETOKEN TO PREVIOUS MESSAGEV3 AFTER RECONFIGURATION ENABLES FEE-TOKEN DRAIN	1	SolidityScan AI	False Positive
M001	Medium	PAUSABLE MECHANISM NOT ENFORCED	1	SolidityScan AI	False Positive
M002	Medium	USERS CANNOT BOUND PROTOCOLFEE/VIASOURCEFEE; OWNER CAN FRONT-RUN FEE INCREASES TO EXTRACT ARBITRARY FEES	1	SolidityScan AI	False Positive
M003	Medium	WRAPPED GAS TOKEN APPROVALS REMAIN FOR OLD MESSAGEV3 AND OLD WRAPPED TOKENS, ENABLING UNINTENDED TOKEN SPEND	1	SolidityScan AI	False Positive
M004	Medium	WRAPPED GAS TOKEN REQUIREMENT CAN BE BYPASSED WHEN VIADESTGAS == 0, RISKING STUCK MESSAGES	1	SolidityScan AI	False Positive
L001	Low	APPROVING MAXIMUM VALUE	3	Automated	False Positive
L002	Low	USE OF FLOATING PRAGMA	1	Automated	False Positive
L003	Low	LACK OF ZERO VALUE CHECK IN TOKEN TRANSFERS	3	Automated	False Positive
L004	Low	MISSING EVENTS	3	Automated	False Positive
L005	Low	MISSING ZERO ADDRESS VALIDATION	3	Automated	False Positive

S. No.	Severity	Bug Type	Instances	Detection Method	Status
L007	Low	USE OWNABLE2STEP	1	Automated	*False Positive*
L008	Low	UNPROTECTED ETHER WITHDRAWAL	1	SolidityScan AI	*False Positive*
I001	Informational	ADDING A RETURN STATEMENT WHEN THE FUNCTION DEFINES A NAMED RETURN VARIABLE IS REDUNDANT	1	Automated	*Pending Fix*
I002	Informational	HARD-CODED GAS LIMITS	1	Automated	*Pending Fix*
I003	Informational	IF-STATEMENT REFACTORING	1	Automated	*Pending Fix*
I004	Informational	MISSING @AUTHOR IN NATSPEC COMMENTS FOR CONTRACT DECLARATION	1	Automated	*Pending Fix*
I005	Informational	MISSING @DEV IN NATSPEC COMMENTS FOR CONTRACT DECLARATION	1	Automated	*Pending Fix*
I006	Informational	MISSING @DEV IN NATSPEC COMMENTS FOR FUNCTIONS	15	Automated	*False Positive*
I007	Informational	MISSING INDEXED KEYWORDS IN EVENTS	2	Automated	*Pending Fix*
I008	Informational	MISSING @INHERITDOC ON OVERRIDE FUNCTIONS	15	Automated	*False Positive*
I009	Informational	MISSING NATSPEC COMMENTS IN SCOPE BLOCKS	7	Automated	*Pending Fix*
I010	Informational	MISSING NATSPEC DESCRIPTIONS FOR PUBLIC VARIABLE DECLARATIONS	13	Automated	*False Positive*
I011	Informational	MISSING @NOTICE IN NATSPEC COMMENTS FOR CONSTRUCTORS	1	Automated	*Pending Fix*
I012	Informational	MISSING @NOTICE IN NATSPEC COMMENTS FOR FUNCTIONS	15	Automated	*False Positive*
I013	Informational	REVERT STATEMENTS WITHIN EXTERNAL AND PUBLIC FUNCTIONS CAN BE USED TO PERFORM DOS ATTACKS	15	Automated	*False Positive*
I014	Informational	UNNAMED FUNCTION PARAMETERS	1	Automated	*Pending Fix*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
I015	Informational	USE CALL INSTEAD OF TRANSFER OR SEND	1	Automated	*Pending Fix*
G001	Gas	ABI ENCODE IS LESS EFFICIENT THAN ABI ENCODEPACKED	1	Automated	*Pending Fix*
G002	Gas	AVOID RE-STORING VALUES	5	Automated	*Pending Fix*
G003	Gas	AVOID ZERO-TO-ONE STORAGE WRITES	4	Automated	*Pending Fix*
G004	Gas	CACHE ADDRESS(THIS) WHEN USED MORE THAN ONCE	4	Automated	*Pending Fix*
G005	Gas	CHEAPER CONDITIONAL OPERATORS	3	Automated	*Pending Fix*
G006	Gas	CHEAPER INEQUALITIES IN IF()	5	Automated	*Pending Fix*
G007	Gas	DEFINE CONSTRUCTOR AS PAYABLE	1	Automated	*Pending Fix*
G008	Gas	REVERTING FUNCTIONS CAN BE PAYABLE	8	Automated	*Pending Fix*
G009	Gas	FUNCTION SHOULD RETURN STRUCT	2	Automated	*Pending Fix*
G010	Gas	GAS INEFFICIENCY DUE TO MULTIPLE OPERANDS IN SINGLE IF/ELSEIF CONDITION	2	Automated	*Pending Fix*
G011	Gas	GAS OPTIMIZATION FOR STATE VARIABLES	1	Automated	*Pending Fix*
G012	Gas	STORAGE VARIABLE CACHING IN MEMORY	4	Automated	*Pending Fix*

04. Vulnerability Details

Issue Type

FEE ON TRANSFER TOKEN INCOMPATIBILITY

S. No.	Severity	Detection Method	Instances
H001	High	SolidityScan AI	1

Bug ID	File Location	Line No.	Action Taken
SSP_120371_152	--	--	*False Positive*

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

PAUSABLE MECHANISM NOT ENFORCED

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

Bug ID	File Location	Line No.	Action Taken
SSP_120371_154	--	--	*False Positive*

Issue Type

APPROVING MAXIMUM VALUE

S. No.	Severity	Detection Method	Instances
L001	Low	Automated	3

Bug ID	File Location	Line No.	Action Taken
SSP_120371_83	--	--	*False Positive*
SSP_120371_84	--	--	*False Positive*
SSP_120371_85	--	--	*False Positive*

Issue Type

ADDING A RETURN STATEMENT WHEN THE FUNCTION DEFINES A NAMED RETURN VARIABLE IS REDUNDANT

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

Bug ID	File Location	Line No.	Action Taken
SSP_120371_12	--	--	*Pending Fix*

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

ABI ENCODE IS LESS EFFICIENT THAN ABI ENCODEPACKED

S. No.	Severity	Detection Method	Instances
G001	Gas	Automated	1

Description

The contract is using `abi.encode()` in the function. In `abi.encode()`, all elementary types are padded to 32 bytes and dynamic arrays include their length, whereas `abi.encodePacked()` will only use the minimal required memory to enco de the data.

Bug ID	File Location	Line No.	Action Taken
SSP_120371_87	via-collateral-v3.sol	L136 - L136	⚠ *Pending Fix*

Issue Type

AVOID RE-STORING VALUES

S. No.	Severity	Detection Method	Instances
G002	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 eq ual to the new value. This practice results in unnecessary gas consumption due to the `Gsreset` operation (2900 ga s), which could be avoided. If the old value and the new value are the same, not updating the storage would avoid thi s 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_120371_30	via-collateral-v3.sol	L161 - L188	*Pending Fix*
SSP_120371_31	via-collateral-v3.sol	L190 - L194	*Pending Fix*
SSP_120371_32	via-collateral-v3.sol	L196 - L200	*Pending Fix*
SSP_120371_33	via-collateral-v3.sol	L202 - L205	*Pending Fix*
SSP_120371_34	via-collateral-v3.sol	L207 - L212	*Pending Fix*

Issue Type

AVOID ZERO-TO-ONE STORAGE WRITES

S. No.	Severity	Detection Method	Instances
G003	Gas	Automated	4

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_120371_95	via-collateral-v3.sol	L82 - L82	*Pending Fix*
SSP_120371_96	via-collateral-v3.sol	L83 - L83	*Pending Fix*
SSP_120371_97	via-collateral-v3.sol	L199 - L199	*Pending Fix*
SSP_120371_98	via-collateral-v3.sol	L204 - L204	*Pending Fix*

Issue Type

CACHE ADDRESS(THIS) WHEN USED MORE THAN ONCE

S. No.	Severity	Detection Method	Instances
G004	Gas	Automated	4

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_120371_140	via-collateral-v3.sol	L123 - L123	*Pending Fix*
SSP_120371_141	via-collateral-v3.sol	L129 - L129	*Pending Fix*
SSP_120371_142	via-collateral-v3.sol	L133 - L133	*Pending Fix*
SSP_120371_143	via-collateral-v3.sol	L256 - L256	*Pending Fix*

Issue Type

CHEAPER CONDITIONAL OPERATORS

S. No.	Severity	Detection Method	Instances
G005	Gas	Automated	3

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_120371_145	via-collateral-v3.sol	L117 - L117	*Pending Fix*
SSP_120371_146	via-collateral-v3.sol	L122 - L122	*Pending Fix*
SSP_120371_147	via-collateral-v3.sol	L127 - L127	*Pending Fix*

Issue Type

CHEAPER INEQUALITIES IN IF()

S. No.	Severity	Detection Method	Instances
G006	Gas	Automated	5

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_120371_5	via-collateral-v3.sol	L112 - L112	*Pending Fix*
SSP_120371_6	via-collateral-v3.sol	L117 - L117	*Pending Fix*
SSP_120371_7	via-collateral-v3.sol	L122 - L122	*Pending Fix*
SSP_120371_8	via-collateral-v3.sol	L127 - L127	*Pending Fix*
SSP_120371_9	via-collateral-v3.sol	L148 - L148	*Pending Fix*

Issue Type

DEFINE CONSTRUCTOR AS PAYABLE

S. No.	Severity	Detection Method	Instances
G007	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_120371_11	via-collateral-v3.sol	L66 - L90	*Pending Fix*

Issue Type

REVERTING FUNCTIONS CAN BE PAYABLE

S. No.	Severity	Detection Method	Instances
G008	Gas	Automated	8

Description

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

Bug ID	File Location	Line No.	Action Taken
SSP_120371_99	via-collateral-v3.sol	L161 - L188	*Pending Fix*
SSP_120371_100	via-collateral-v3.sol	L190 - L194	*Pending Fix*
SSP_120371_101	via-collateral-v3.sol	L196 - L200	*Pending Fix*
SSP_120371_102	via-collateral-v3.sol	L202 - L205	*Pending Fix*
SSP_120371_103	via-collateral-v3.sol	L207 - L212	*Pending Fix*
SSP_120371_104	via-collateral-v3.sol	L214 - L220	*Pending Fix*
SSP_120371_105	via-collateral-v3.sol	L222 - L224	*Pending Fix*
SSP_120371_106	via-collateral-v3.sol	L226 - L237	*Pending Fix*

Issue Type

FUNCTION SHOULD RETURN STRUCT

S. No.	Severity	Detection Method	Instances
G009	Gas	Automated	2

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_120371_3	via-collateral-v3.sol	L240 - L249	*Pending Fix*
SSP_120371_4	via-collateral-v3.sol	L263 - L265	*Pending Fix*

Issue Type

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

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

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_120371_1	via-collateral-v3.sol	L75 - L75	*Pending Fix*
SSP_120371_2	via-collateral-v3.sol	L169 - L171	*Pending Fix*

Issue Type

GAS OPTIMIZATION FOR STATE VARIABLES

S. No.	Severity	Detection Method	Instances
G011	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_120371_10	via-collateral-v3.sol	L139 - L139	⚠ *Pending Fix*

Issue Type

STORAGE VARIABLE CACHING IN MEMORY

S. No.	Severity	Detection Method	Instances
G012	Gas	Automated	4

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_120371_63	via-collateral-v3.sol	L93 - L141	*Pending Fix*
SSP_120371_63	via-collateral-v3.sol	L93 - L141	*Pending Fix*
SSP_120371_64	via-collateral-v3.sol	L214 - L220	*Pending Fix*
SSP_120371_65	via-collateral-v3.sol	L240 - L249	*Pending Fix*

05. Scan History

Critical High Medium Low Informational Gas

No	Date	Security Score	Scan Overview
1.	2026-01-09	72.77	0 1 0 0 17 39

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.