SolidityScan

Powered by CRED SHIELDS

Security Assessment Report

contracts/2_Owner.sol

30 June 2025

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

Table of Contents

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

contracts/2_Owner.sol

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 contracts/2_Owner.sol 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 450+ modules. The scanning and auditing process covers the following areas:

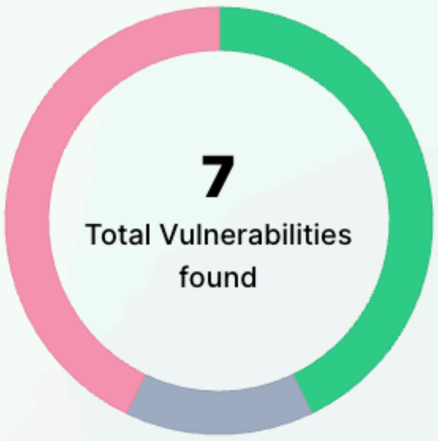
Various common and uncommon attack vectors will be investigated to ensure that the smart contracts are secure from malicious actors. The scanner modules find and flag issues related to Gas optimizations that help in reducing the overall Gas cost It scans and evaluates the codebase against industry best 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 contracts/2_Owner.sol introduces new features or refactors the code.

03. Findings Summary

contracts/2_Owner.sol
File Scan

Security Score **93.48**/100	Scan duration **4 secs**	Lines of code **46**



0	0	0	3	1	3
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	0	0	8
✓ *Fixed*	*False Positive*	*Won't Fix*	⚠ *Pending Fix*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
L001	● Low	COMPILER VERSION TOO RECENT	1	Automated	⚠ *Pending Fix*
L002	● Low	USE OF FLOATING PRAGMA	1	Automated	⚠ *Pending Fix*
L003	● Low	OUTDATED COMPILER VERSION	1	Automated	⚠ *Pending Fix*
I001	● Informational	MISSING UNDERSCORE IN NAMING VARIABLES	1	Automated	⚠ *Pending Fix*
G001	● Gas	AVOID RE-STORING VALUES	1	Automated	⚠ *Pending Fix*
G002	● Gas	DEFINE CONSTRUCTOR AS PAYABLE	1	Automated	⚠ *Pending Fix*
G003	● Gas	LONG REQUIRE/REVERT STRINGS	1	Automated	⚠ *Pending Fix*

04. Vulnerability Details

Issue Type

COMPILER VERSION TOO RECENT

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

MISSING UNDERSCORE IN NAMING VARIABLES

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

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

AVOID RE-STORING VALUES

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

Description

The function is found to be allowing re-storing the value in the contract's state variable even when the old valu e is equal to the new value. This practice results in unnecessary gas consumption due to the `Gsreset` operati on (2900 gas), which could be avoided. If the old value and the new value are the same, not updating the stora ge would avoid this cost and could instead incur a `Gcoldsload` (2100 gas) or a `Gwarmaccess` (100 gas), pot entially saving gas.

Bug ID	File Location	Line No.	Action Taken
SSP_84877_1	/2_Owner.sol	L42 - L46	⚠ *Pending Fix*

Issue Type

DEFINE CONSTRUCTOR AS PAYABLE

S. No.	Severity	Detection Method	Instances
G002	● 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 depl
oyment.

Bug ID	File Location	Line No.	Action Taken
SSP_84877_2	/2_Owner.sol	L32 - L36	⚠ *Pending Fix*

Issue Type

LONG REQUIRE/REVERT STRINGS

S. No.	Severity	Detection Method	Instances
G003	● Gas	Automated	1

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`, alo ng with additional overhead for computing memory offset, and other parameters.

Bug ID	File Location	Line No.	Action Taken
SSP_84877_4	/2_Owner.sol	L43 - L43	⚠ *Pending Fix*

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.

05. Scan History

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

No	Date	Security Score	Scan Overview
1.	2025-06-04	**93.48**	● 0 ● 0 ● 0 ● 3 ● 1 ● 3