SolidityScan | Powered by CRED SHIELDS

Security Assessment Report

Reef Bridge Unlocker Full Report

28 Jan 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

Reef Bridge Unlocker Full Report

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 Reef Bridge Unlocker Full Report 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 (100+) 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 Reef Bridge Unlocker Full Report introduces new features or refactors the code.

03. Findings Summary

Reef Bridge Unlocker Full Report
File Scan

Security Score	Scan duration	Lines of code
78.08/100	**4 secs**	**73**

0	0	0	2	5	7
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	14
Fixed	False Positive	Won't Fix	Pending Fix

S. No.	Severity	Bug Type	Instances	Detection Method	Status
L001	Low	EVENT BASED REENTRANCY	1	Automated	Pending Fix
L002	Low	OUTDATED COMPILER VERSION	1	Automated	Pending Fix
I001	Informational	MISSING INDEXED KEYWORDS IN EVENTS	2	Automated	Pending Fix
I002	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	3	Automated	Pending Fix
G001	Gas	ABI ENCODE IS LESS EFFICIENT THAN ABI ENCODEPACKED	1	Automated	Pending Fix
G002	Gas	AVOID RE-STORING VALUES	1	Automated	Pending Fix
G003	Gas	CACHE ADDRESS(THIS) WHEN USED MORE THAN ONCE	2	Automated	Pending Fix
G004	Gas	CHEAPER INEQUALITIES IN IF()	1	Automated	Pending Fix
G005	Gas	STORAGE VARIABLE CACHING IN MEMORY	3	Automated	Pending Fix

04. Vulnerability Details

Issue Type

EVENT BASED REENTRANCY

S. No.	Severity	Detection Method	Instances
L001	Low	Automated	1

Description

In a Re-entrancy attack, a malicious contract calls back into the calling contract before the first invocation of the fun ction is finished. This may cause the different invocations of the function to interact in undesirable ways, especially i n cases where the function is updating state variables after the external calls.
In the case of event-based Re-entrancy attacks, events are emitted after an external call leading to missing event cal ls.

Bug ID	File Location	Line No.	Action Taken
SSP_56810_14	/BurnBridge.sol	L77 - L84	*Pending Fix*

/BurnBridge.sol L77 - L84

```
76
77      function burn(uint256 amount) external onlyMessageOwner {
78          uint256 balance = IERC20(i_token).balanceOf(address(this));
79
80          if (amount == 0 || balance < amount) amount = balance;
81
82          emit Burned(amount);
83          SafeERC20.safeTransfer(IERC20(i_token), address(0), amount);
84      }
85
86      function getToken() external view returns (address) {
```

Issue Type

OUTDATED COMPILER VERSION

S. No.	Severity	Detection Method	Instances
L002	Low	Automated	1

Description

Using an outdated compiler version can be problematic especially if there are publicly disclosed bugs and issues that affect the current compiler version.

Bug ID	File Location	Line No.	Action Taken
SSP_56810_10	/BurnBridge.sol	L2 - L2	*Pending Fix*

/BurnBridge.sol L2 - L2

```
1  // SPDX-License-Identifier: UNLICENSED
2  pragma solidity 0.8.17;
3
4  // import {MessageClient, IERC20cl} from "@vialabs/contracts/MessageClient.sol";
```

Issue Type

MISSING INDEXED KEYWORDS IN EVENTS

S. No.	Severity	Detection Method	Instances
I001	Informational	Automated	2

Description

Events are essential for tracking off-chain data and when the event paraemters are `indexed` they can be used as fil ter options which will help getting only the specific data instead of all the logs.

Bug ID	File Location	Line No.	Action Taken
SSP_56810_12	/BurnBridge.sol	L21 - L21	*Pending Fix*

/BurnBridge.sol L21 - L21

```
20        event Burned(uint256 amount);
21        event Sent(address recipient, uint256 amount);
22        event Received(address recipient, uint256 amount);
23        event Fee(uint256 amount);
```

Bug ID	File Location	Line No.	Action Taken
SSP_56810_13	/BurnBridge.sol	L22 - L22	⚠ *Pending Fix*

/BurnBridge.sol L22 - L22

```
21        event Sent(address recipient, uint256 amount);
22        event Received(address recipient, uint256 amount);
23        event Fee(uint256 amount);
24
```

Issue Type

MISSING UNDERSCORE IN NAMING VARIABLES

S. No.	Severity	Detection Method	Instances
I002	Informational	Automated	3

Description

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

Bug ID	File Location	Line No.	Action Taken
SSP_56810_7	/BurnBridge.sol	L15 - L15	*Pending Fix*

/BurnBridge.sol L15 - L15

```
14
15        address private immutable i_token;
16        uint256 private immutable i_destChainId;
17
```

Bug ID	File Location	Line No.	Action Taken
SSP_56810_8	/BurnBridge.sol	L16 - L16	*Pending Fix*

/BurnBridge.sol L16 - L16

```
15        address private immutable i_token;
16        uint256 private immutable i_destChainId;
17
18        uint256 private s_fee = 0.1 ether;
```

Bug ID	File Location	Line No.	Action Taken
SSP_56810_9	/BurnBridge.sol	L18 - L18	*Pending Fix*

/BurnBridge.sol L18 - L18

```
17
18        uint256 private s_fee = 0.1 ether;
19
20        event Burned(uint256 amount);
```

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 byt es and dynamic arrays include their length, whereas `abi.encodePacked()` will only use the minimal required mem ory to encode the data.

Bug ID	File Location	Line No.	Action Taken
SSP_56810_4	/BurnBridge.sol	L41 - L41	*Pending Fix*

/BurnBridge.sol L41 - L41

```
40
41          bytes memory messageData = abi.encode(recipient, amount);
42
43          emit Sent(recipient, amount);
```

Issue Type

AVOID RE-STORING VALUES

S. No.	Severity	Detection Method	Instances
G002	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 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_56810_3	/BurnBridge.sol	L71 - L75	*Pending Fix*

/BurnBridge.sol L71 - L75

```
70
71      function setFee(uint256 amount) external onlyMessageOwner {
72          s_fee = amount;
73
74          emit Fee(s_fee);
75      }
76
77      function burn(uint256 amount) external onlyMessageOwner {
```

Issue Type

CACHE ADDRESS(THIS) WHEN USED MORE THAN ONCE

S. No.	Severity	Detection Method	Instances
G003	Gas	Automated	2

Description

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

Bug ID	File Location	Line No.	Action Taken
SSP_56810_1	/BurnBridge.sol	L47 - L47	*Pending Fix*

/BurnBridge.sol L47 - L47

```
46                msg.sender,
47                address(this),
48                amount
49            );
```

Bug ID	File Location	Line No.	Action Taken
SSP_56810_2	/BurnBridge.sol	L78 - L78	⚠ *Pending Fix*

/BurnBridge.sol L78 - L78

```
77        function burn(uint256 amount) external onlyMessageOwner {
78            uint256 balance = IERC20(i_token).balanceOf(address(this));
79
80            if (amount == 0 || balance < amount) amount = balance;
```

Issue Type

CHEAPER INEQUALITIES IN IF()

S. No.	Severity	Detection Method	Instances
G004	Gas	Automated	1

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_56810_11	/BurnBridge.sol	L80 - L80	*Pending Fix*

/BurnBridge.sol L80 - L80

```
79
80          if (amount == 0 || balance < amount) amount = balance;
81
82          emit Burned(amount);
```

Issue Type

STORAGE VARIABLE CACHING IN MEMORY

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

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_56810_5	/BurnBridge.sol	L32 - L51	*Pending Fix*

/BurnBridge.sol L32 - L51

```
31
32      function bridge(
33          address recipient,
34          uint256 amount
35      ) external payable onlyActiveChain(i_destChainId) {
36          uint256 chainId = block.chainid;
37          // @note MUST NOT WORK on reef
38          if (chainId == 13939) revert BurnBridge__InvalidSource();
39          if (msg.value != s_fee) revert BurnBridge__InvalidFee(s_fee);
40
41          bytes memory messageData = abi.encode(recipient, amount);
42
43          emit Sent(recipient, amount);
44          SafeERC20.safeTransferFrom(
45              IERC20(i_token),
46              msg.sender,
47              address(this),
48              amount
49          );
50          _sendMessage(i_destChainId, messageData);
51      }
52
53      function messageProcess(
```

Bug ID	File Location	Line No.	Action Taken
SSP_56810_5	/BurnBridge.sol	L32 - L51	⚠ *Pending Fix*

/BurnBridge.sol L32 - L51

```
31
32      function bridge(
33          address recipient,
34          uint256 amount
35      ) external payable onlyActiveChain(i_destChainId) {
36          uint256 chainId = block.chainid;
37          // @note MUST NOT WORK on reef
38          if (chainId == 13939) revert BurnBridge__InvalidSource();
39          if (msg.value != s_fee) revert BurnBridge__InvalidFee(s_fee);
40
41          bytes memory messageData = abi.encode(recipient, amount);
42
43          emit Sent(recipient, amount);
44          SafeERC20.safeTransferFrom(
45              IERC20(i_token),
46              msg.sender,
47              address(this),
48              amount
49          );
50          _sendMessage(i_destChainId, messageData);
51      }
52
53      function messageProcess(
```

Bug ID	File Location	Line No.	Action Taken
SSP_56810_6	/BurnBridge.sol	L77 - L84	*Pending Fix*

/BurnBridge.sol L77 - L84

```
76
77        function burn(uint256 amount) external onlyMessageOwner {
78            uint256 balance = IERC20(i_token).balanceOf(address(this));
79
80            if (amount == 0 || balance < amount) amount = balance;
81
82            emit Burned(amount);
83            SafeERC20.safeTransfer(IERC20(i_token), address(0), amount);
84        }
85
86        function getToken() external view returns (address) {
```

05. Scan History

Critical High Medium Low Informational Gas

No	Date	Security Score	Scan Overview
1.	2025-01-28	78.08	0 0 0 2 5 7

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.