



deBridge – CrosschainForwarder Update

Smart Contract Security Audit

Prepared by: Halborn

Date of Engagement: July 17th, 2022 – July 25th, 2022

Visit: Halborn.com

DOCUMENT REVISION HISTORY	3
CONTACTS	3
1 EXECUTIVE OVERVIEW	4
1.1 INTRODUCTION	5
1.2 AUDIT SUMMARY	5
1.3 TEST APPROACH & METHODOLOGY	5
RISK METHODOLOGY	6
1.4 SCOPE	8
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW	9
3 FINDINGS & TECH DETAILS	10
3.1 (HAL-01) LOW LEVEL CALLS WITH SOLIDITY VERSION 0.8.7 CAN RESULT IN OPTIMIZER BUG - LOW	12
Description	12
Code Location	12
Risk Level	12
Recommendation	12
3.2 (HAL-02) USE DISABLEINITIALIZERS IN THE UPGRADABLE CONTRACTS - INFORMATIONAL	13
Description	13
Risk Level	13
Code Location	13
Recommendation	13
3.3 (HAL-03) OPEN TODOs - INFORMATIONAL	15
Description	15
Code Location	15

	TO-DO	15
	Risk Level	15
	Recommendation	15
3.4	(HAL-04) LACK OF PAUSE/UNPAUSE FUNCTIONALITY - INFORMATIONAL	16
	Description	16
	Risk Level	16
	Recommendation	16
3.5	(HAL-05) OPTIMIZE UNSIGNED INTEGER COMPARISON - INFORMATIONAL	17
	Description	17
	Risk Level	17
	Code Location	17
	Recommendation	17
3.6	(HAL-06) GATEPARAMS ARE NOT VALIDATED - INFORMATIONAL	18
	Description	18
	Risk Level	18
	Code Location	18
	Recommendation	18
3.7	(HAL-07) OUT OF DATE OPENZEPPELIN PACKAGES - INFORMATIONAL	19
	Description	19
	Risk Level	19
	Code Location	19
	Recommendation	20

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	07/17/2022	Gokberk Gulgun
0.2	Document Edits	07/25/2022	Gokberk Gulgun
0.3	Draft Review	07/25/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com



EXECUTIVE OVERVIEW



1.1 INTRODUCTION

deBridge engaged Halborn to conduct a security assessment on their smart contract update beginning on July 17th, 2022 and ending July 25th, 2022. deBridge is a cross-chain interoperability and liquidity transfer protocol that allows truly decentralized transfer of assets between various blockchains. deBridge is a cross-chain interoperability and liquidity transfer protocol that allows decentralized transfer of assets between blockchains.

1.2 AUDIT SUMMARY

The team at Halborn was provided one week for the engagement and assigned two full-time security engineers to audit the security of the assets in scope. The engineer is a blockchain and smart contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to achieve the following:

- Identify potential security issues with the smart contract changes.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy with the scope of the smart contract audit. While manual testing is recommended to uncover flaws in logic, process, and implementation, automated testing techniques help enhance the smart contract code coverage and quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose.

- Smart contract manual code review and walkthrough.
- Graphing out functionality and contract logic/connectivity/functions([solgraph](#)).
- Manual testing of core functions through [Hardhat](#) and [Ganache](#).
- Manual testing with custom scripts.
- Static Analysis of security for scoped contract, and imported functions.([Slither](#)).
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. ([MythX](#)).
- Testnet deployment ([Remix IDE](#)).

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

- 5 - Almost certain an incident will occur.
- 4 - High probability of an incident occurring.
- 3 - Potential of a security incident in the long term.
- 2 - Low probability of an incident occurring.
- 1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

- 5 - May cause devastating and unrecoverable impact or loss.
- 4 - May cause a significant level of impact or loss.
- 3 - May cause a partial impact or loss to many.
- 2 - May cause temporary impact or loss.
- 1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
----------	------	--------	-----	---------------

10 - CRITICAL

9 - 8 - HIGH

7 - 6 - MEDIUM

5 - 4 - LOW

3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

The review was scoped to only changes applied to the `contracts/Cross-chainForwarder.sol`:

Commit

Smart contracts:

- `CrossChainForwarder.sol`

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	1	6

LIKELIHOOD

IMPACT

(HAL-01)				
(HAL-02)				
(HAL-03) (HAL-04) (HAL-05) (HAL-06) (HAL-07)				

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL-01 - LOW LEVEL CALLS WITH SOLIDITY VERSION 0.8.7 CAN RESULT IN OPTIMIZER BUG	Low	-
HAL-02 - USE DISABLEINITIALIZERS IN THE UPGRADABLE CONTRACTS	Informational	-
HAL-03 - OPEN TODOs	Informational	-
HAL-04 - LACK OF PAUSE/UNPAUSE FUNCTIONALITY	Informational	-
HAL-05 - OPTIMIZE UNSIGNED INTEGER COMPARISON	Informational	-
HAL-06 - GATEPARAMS ARE NOT VALIDATED	Informational	-
HAL-07 - OUT OF DATE OPENZEPELIN PACKAGES	Informational	-



FINDINGS & TECH DETAILS



3.1 (HAL-01) LOW LEVEL CALLS WITH SOLIDITY VERSION 0.8.7 CAN RESULT IN OPTIMIZER BUG - LOW

Description:

Using low-level calls with solidity version 0.8.7 which can result in optimizer bug. The bug can detail can be seen from the following [LINK](#)

Code Location:

Listing 1

```
1 pragma solidity 0.8.7;
```

Risk Level:

Likelihood - 1

Impact - 3

Recommendation:

It is recommended to update the contracts to the recently patched version 0.8.15.

3.2 (HAL-02) USE DISABLEINITIALIZERS IN THE UPGRADABLE CONTRACTS - INFORMATIONAL

Description:

In the proxy pattern, an uninitialized implementation contract can be initialized by someone else taking over the contract. Even if it doesn't affect the proxy contracts, it's a good practice to initialize them yourself to prevent any mishap against unseen vulnerabilities.

Risk Level:

Likelihood - 1

Impact - 2

Code Location:

Location

Listing 2: CrosschainForwarder.sol

```
39     function initialize(IDeBridgeGate _deBridgeGate) external
    ↳ initializer {
40         ForwarderBase.initializeBase();
41         deBridgeGate = _deBridgeGate;
42     }
```

Recommendation:

For the deployed contracts, execute the initialize() functions on the implementation contracts. There's a risk that they might be front run, but it's less likely since they are still uninitialized, and the front-runner

is not directly benefiting from executing the transaction itself. For future, consider calling OZ's `_disableInitializers()` in the implementation contract's constructor. Use the same name for both arguments.

3.3 (HAL-03) OPEN TODOs - INFORMATIONAL

Description:

Open To-dos can point to architecture or programming issues that still need to be resolved. Often these kinds of comments indicate areas of complexity or confusion for developers. This provides value and insight to an attacker who aims to cause damage to the protocol.

Code Location:

TO-DO:

Listing 3: Open TODOs

```
1 840976fbb38d09ec6c1cf66f5f82d744a50cffe7/contracts/  
↳ CrosschainForwarder.sol#L313  
2 840976fbb38d09ec6c1cf66f5f82d744a50cffe7/contracts/  
↳ CrosschainForwarder.sol#L173
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

Consider resolving the To-dos before deploying code to a production context. Use an independent issue tracker or other project management software to track development tasks.

3.4 (HAL-04) LACK OF PAUSE/UNPAUSE FUNCTIONALITY – INFORMATIONAL

Description:

In case a hack is occurring, or an exploit is discovered, the team should be able to pause functionality until the necessary changes are made to the system. Because an attack would probably span some blocks, a method for pausing the contract would be able to interrupt any such attack if discovered.

To use a thorchain example again, the team behind thorchain noticed an attack was going to occur well before the system transferred funds to the hacker. However, they were unable to shut the system down fast enough. According to the incidence report [here](#)

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

Pause functionality on the contract would have helped secure the funds quickly.

3.5 (HAL-05) OPTIMIZE UNSIGNED INTEGER COMPARISON - INFORMATIONAL

Description:

The check `!= 0` costs less gas compared to `> 0` for unsigned integers in require statements with the optimizer enabled. While it may seem that `> 0` is cheaper than `!=0`, this is only true without the optimizer enabled and outside a require statement. If the optimizer is enabled at 10k and It is in a require statement, that would be more gas efficient.

Risk Level:

Likelihood - 1

Impact - 1

Code Location:

Location

Listing 4: CrosschainForwarder.sol

```
140         if (_affiliateFeeAmount > 0 && _affiliateFeeRecipient !=  
    ↳ address(0)) {  
141     ...
```

Recommendation:

Consider changing `> 0` comparison with `!= 0`.

3.6 (HAL-06) GATEPARAMS ARE NOT VALIDATED – INFORMATIONAL

Description:

During the new commit review, It has been observed that **GateParams** has been given as a function parameter on the bridge functionalities. However, in the **CrossChainForwarder** contract, **chainId** and other parameters are not validated.

Risk Level:

Likelihood - 1

Impact - 1

Code Location:

Location

Listing 5: ICrossChainForwarder.sol

```
1      struct GateParams {
2          uint256 chainId;
3          address receiver;
4          bool useAssetFee;
5          uint32 referralCode;
6          bytes autoParams;
7      }
8
```

Recommendation:

Ensure that all parameters are validated. Missing validation on the **chainId** can leads to funds locking on the deBridge gate.

3.7 (HAL-07) OUT OF DATE OPENZEPPELIN PACKAGES – INFORMATIONAL

Description:

The **OpenZeppelin** library dependency version is 4.4.2 for the **CrossChainForwarder**. Recently, two additional **CVEs** have been found on the OpenZeppelin contracts. Even if the advisory is not directly affect the **CrossChainForwarder** contract, It is recommended to update Openzeppelin contracts.

Risk Level:

Likelihood - 1

Impact - 1

Code Location:

Location

Listing 6: package.json

```
1  "dependencies": {
2    "@openzeppelin/contracts": "^4.4.2",
3    "@openzeppelin/contracts-upgradeable": "^4.4.2",
4    "@openzeppelin/hardhat-upgrades": "^1.14.0",
5    "dotenv": "^15.0.0",
6    "hardhat": "^2.8.3",
7    "hardhat-deploy": "^0.11.0",
8    "node-fetch": "^2.6.7",
9    "prettier": "^2.5.1",
10   "prettier-plugin-solidity": "^1.0.0-beta.19",
11   "solhint": "^3.3.6"
12 },
```

Recommendation:

Use the latest version of OpenZeppelin contracts available.



THANK YOU FOR CHOOSING

 **HALBORN**

