CrosschainForwarder Allowances

deBridge

HALBORN

Report a Bug

CrosschainForwarder Allowances - deBridge

Prepared by: HALBORN Last Updated 09/17/2024 Date of Engagement by: September 3rd, 2024 - September 6th, 2024

Summary

100% ⓘ OF ALL REPORTED FINDINGS HAVE BEEN ADDRESSED

ALL FINDINGS	CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
1	0	0	0	1	0

1. Introduction

deBridge engaged Halborn to conduct a security assessment on their smart contracts beginning on **09-05-2024** and ending on **09-06-2024**. The security assessment was scoped to the smart contracts provided in the https://github.com/debridge-finance/cross-chain-swaps GitHub repository. Commit hashes and further details can be found in the Scope section of this report. The deBridge codebase in scope consists of **a cross-chain swap protocol.**

2. Assessment Summary

Halborn was engaged to verify that allowances granted by deBridge users to the CrosschainForwarder contract would not be affected in the event of a catastrophic compromise or exploitation of any whitelisted third-party router. Halborn assessed the time and resources required to perform a professional audit and assigned 1 full-time security engineer to review the security of the smart contracts in scope.

The purpose of the assessment is to:

- Identify potential security issues within the smart contracts.
- Ensure that smart contract functionality operates as intended.

TABLE OF CONTENTS

3. Test Approach And Methodology

`Halborn` performed a combination of manual review of the code and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this assessment. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of smart contracts and can quickly identify items that do not follow security best practices.

The following phases and associated tools were used throughout the term of the assessment:

- Research into architecture, purpose and use of the platform.
- Smart contract manual code review and walkthrough to identify any logic issue.
- Thorough assessment of safety and usage of critical Solidity variables and functions in scope that could led to arithmetic related vulnerabilities.
- Local testing with custom scripts (`Foundry`).
- Fork testing against main networks (`Foundry`).
- Static analysis of security for scoped contract, and imported functions (`Slither`).

3.1 Out-Of-Scope

- External libraries and financial-related attacks.
- New features/implementations after/within the **remediation commit IDs**.
- Changes that occur outside of the scope of PRs.

CONCLUSION

As a result of the audit, `Halborn` confirms that any security issues or vulnerabilities in whitelisted routers will **not directly** impact users who have granted token approvals to the `CrosschainForwarder` contract. The allowances remain secure, even in the event of a compromise or exploitation of these third-party routers.

Additionally, it has been manually confirmed that the bytecode of the `CrosschainForwarder` contract in scope identically matches the bytecode deployed in the following addresses:

- Ethereum => `0xe67534a9f24cc000f467eaa17f920bf63b87a2cd`
- BNB chain => `0xe67534a9f24cc000f467eaa17f920bf63b87a2cd`
- Polygon => `0xe67534a9f24cc000f467eaa17f920bf63b87a2cd`
- Arbitrum => `0xe67534a9f24cc000f467eaa17f920bf63b87a2cd`
- Avalanche => `0xe67534a9f24cc000f467eaa17f920bf63b87a2cd`
- Linea => `0xe67534a9f24cc000f467eaa17f920bf63b87a2cd`
- Optimism => `0xe67534a9f24cc000f467eaa17f920bf63b87a2cd`
- Base => `0xe67534a9f24cc000f467eaa17f920bf63b87a2cd`
- Neon => `0xF7399C83B12edD1a21aAB250D30CCB4474902688`
- Gnosis => `0xf7399c83b12edd1a21aab250d30ccb4474902688`
- Lightlink => `0xf7399c83b12edd1a21aab250d30ccb4474902688`
- Metis => `0xf7399c83b12edd1a21aab250d30ccb4474902688`

Proxy contract is deployed at **`0x663dc15d3c1ac63ff12e45ab68fea3f0a883c251`** in all chains.

4. RISK METHODOLOGY

Every vulnerability and issue observed by Halborn is ranked based on **two sets** of **Metrics** and a **Severity Coefficient**. This system is inspired by the industry standard Common Vulnerability Scoring System.

The two **Metric sets** are: **Exploitability** and **Impact**. **Exploitability** captures the ease and technical means by which vulnerabilities can be exploited and **Impact** describes the consequences of a successful exploit.

The **Severity Coefficients** is designed to further refine the accuracy of the ranking with two factors: **Reversibility** and **Scope**. These capture the impact of the vulnerability on the environment as well as the number of users and smart contracts affected.

The final score is a value between 0-10 rounded up to 1 decimal place and 10 corresponding to the highest security risk. This provides an objective and accurate rating of the severity of security vulnerabilities in smart contracts.

The system is designed to assist in identifying and prioritizing vulnerabilities based on their level of risk to address the most critical issues in a timely manner.

4.1 EXPLOITABILITY

ATTACK ORIGIN (AO):

Captures whether the attack requires compromising a specific account.

ATTACK COST (AC):

Captures the cost of exploiting the vulnerability incurred by the attacker relative to sending a single transaction on the relevant blockchain. Includes but is not limited to financial and computational cost.

ATTACK COMPLEXITY (AX):

Describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability. Includes but is not limited to macro situation, available third-party liquidity and regulatory challenges.

METRICS:

EXPLOITABILIY METRIC (M_E)	METRIC VALUE	NUMERICAL VALUE
Attack Origin (AO)	Arbitrary (AO:A) Specific (AO:S)	1 0.2
Attack Cost (AC)	Low (AC:L) Medium (AC:M) High (AC:H)	1 0.67 0.33
Attack Complexity (AX)	Low (AX:L) Medium (AX:M) High (AX:H)	1 0.67 0.33

Exploitability E is calculated using the following formula:

$$E = \prod m_e$$

4.2 IMPACT

CONFIDENTIALITY (C):

Measures the impact to the confidentiality of the information resources managed by the contract due to a successfully exploited vulnerability. Confidentiality refers to limiting access to authorized users only.

INTEGRITY (I):

Measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of data stored and/or processed on-chain. Integrity impact directly affecting Deposit or Yield records is excluded.

AVAILABILITY (A):

Measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability. This metric refers to smart contract features and functionality, not state. Availability impact directly affecting Deposit or Yield is excluded.

DEPOSIT (D):

Measures the impact to the deposits made to the contract by either users or owners.

YIELD (Y):

Measures the impact to the yield generated by the contract for either users or owners.

METRICS:

IMPACT METRIC (M_I)	METRIC VALUE	NUMERICAL VALUE
Confidentiality (C)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1

IMPACT METRIC (M_I)	METRIC VALUE	NUMERICAL VALUE
Integrity (I)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1
Availability (A)	None (A:N) Low (A:L) Medium (A:M) High (A:H) Critical (A:C)	0 0.25 0.5 0.75 1
Deposit (D)	None (D:N) Low (D:L) Medium (D:M) High (D:H) Critical (D:C)	0 0.25 0.5 0.75 1
Yield (Y)	None (Y:N) Low (Y:L) Medium (Y:M) High (Y:H) Critical (Y:C)	0 0.25 0.5 0.75 1

Impact I is calculated using the following formula:

$$I = max(m_I) + \frac{\sum m_I - max(m_I)}{4}$$

4.3 SEVERITY COEFFICIENT

REVERSIBILITY (R):

Describes the share of the exploited vulnerability effects that can be reversed. For upgradeable contracts, assume the contract private key is available.

SCOPE (S):

Captures whether a vulnerability in one vulnerable contract impacts resources in other contracts.

METRICS:

SEVERITY COEFFICIENT (C)	COEFFICIENT VALUE	NUMERICAL VALUE
Reversibility (r)	None (R:N) Partial (R:P) Full (R:F)	1 0.5 0.25
Scope (s)	Changed (S:C) Unchanged (S:U)	1.25 1

Severity Coefficient C is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score S is obtained by:

$$S = min(10, EIC * 10)$$

The score is rounded up to 1 decimal places.

SEVERITY	SCORE VALUE RANGE
Critical	9 - 10
High	7 - 8.9
Medium	4.5 - 6.9
Low	2 - 4.4
Informational	0 - 1.9

5. SCOPE

FILES AND REPOSITORY

(a) Repository: cross-chain-swaps

(b) Assessed Commit ID: e978fef

(c) Items in scope:

- CrosschainForwarder.sol
- Verification of possibility for allowances given by users to CrosschainForwarder to be affected if a third party router is compromised.

Out-of-Scope: Paraswap Augustus V6 deployed at: 0x6a000f20005980200259b80c5102003040001068 [Ethereum, Arbitrum, Avalanche, Base, BNB, Fantom, Optimism, Polygon], AllowanceHolder deployed at: 0x0000000000001fF3684f28c67538d4D072C22734 [Ethereum, Polygon, Base, Optimism, Arbitrum], 0x0000000000005E88410CcDFaDe4a5EfaE4b49562 [BNB Chain, Avalanche], 0x000000000000175a8b9bC6d539B3708EEd92EA6c [Linea]

Out-of-Scope: New features/implementations after the remediation commit IDs.

6. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW
0	0	0	1

INFORMATIONAL

0

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
ASSETS AT POTENTIAL RISK DUE TO EXCESSIVE ALLOWANCE	LOW	RISK ACCEPTED

7. FINDINGS & TECH DETAILS

7.1 ASSETS AT POTENTIAL RISK DUE TO EXCESSIVE ALLOWANCE

// LOW

Description

The `CrosschainForwarder` contract uses the `_lazyApprove()` function to approve an unlimited allowance for the specified spender. This function is called when interacting with supported routers and the `deBridge` gate.

```
537 function _lazyApprove(address _tokenAddress, address _spender, uint2
538     IERC20Upgradeable token = IERC20Upgradeable(_tokenAddress);
539     uint256 currentAllowance = token.allowance(address(this), _spend
540 
541     if (currentAllowance < _amount) {
542         // if an approval was issued before
543         token.safeApprove(_spender, 0);
544         // create permanent approve
545         token.safeApprove(_spender, type(uint256).max);
546     }
547 }
```

While this implementation wouldn't directly affect user allowances, it could potentially put the tokens held by the `CrosschainForwarder` contract at risk if a supported router is compromised. This could impact the contract's ability to process transactions if its token holdings are compromised.

BVSS

AO:A/AC:L/AX:L/R:N/S:U/C:N/A:N/I:N/D:L/Y:N (2.5)

Recommendation

To mitigate this risk, consider implementing a per-transaction approval mechanism instead of unlimited allowances for routers. This would limit the potential impact of a compromised router on the `CrosschainForwarder` contract's token holdings.

Remediation

RISK ACCEPTED : `Halborn` confirms that any security issues or vulnerabilities in whitelisted routers will **not directly** impact users who have granted token approvals to the `CrosschainForwarder` contract. The allowances remain secure, even in the event of a compromise or exploitation of these third-party routers.

References

debridge-finance/cross-chain-swaps/contracts/CrosschainForwarder.sol#L537-L547

STATIC ANALYSIS REPORT

Description

Halborn used automated testing techniques to enhance the coverage of certain areas of the smart contracts in scope. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified the smart contracts in the repository and was able to compile them correctly into their abis and binary format, Slither was run against the contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.
The security team assessed all findings identified by the Slither software, however, findings with related to external dependencies are not included in the below results for the sake of report readability.

Output

The findings obtained as a result of the Slither scan were reviewed, and were not included in the report because: **a)** they were determined as false positives, and/or **b)** findings were not applicable due to being out-of-scope issues.

```
CrosschainForwarder._distributeAffiliateFee(address,uint256,uint256,address) (contracts/CrosschainForwarder.sol#290-315) sends eth to arbitrary user
        Dangerous calls:
        - (success,None) = _affiliateFeeRecipient.call{value: _affiliateFeeAmount}() (contracts/CrosschainForwarder.sol#307)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
CrosschainForwarder.strictlySwapAndCallDln(address,uint256,bytes,CrosschainForwarder.SwapDetails,address,bytes,bytes32) (contracts/CrosschainForwarder.sol#135-165) ignores return value by (status,None,None) = IDlnDestination(_target).takeOrders(_orderId) (contracts/CrosschainForwarder.sol#148)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Reentrancy in CrosschainForwarder._performSwap(address,uint256,uint256,address,bytes,address) (contracts/CrosschainForwarder.sol#327-347):
        External calls:
        - srcAmountOut = _swapToERC20Via(_srcSwapRouter,_srcSwapCalldata,_srcAmountIn,IERC20Upgradeable(_srcTokenOut)) (contracts/CrosschainForwarder.sol#338)
                - (success,returnData) = _to.call{value: _msgValue}(_data) (contracts/CrosschainForwarder.sol#429)
        - _lazyApprove(_srcTokenIn,_srcSwapRouter,_srcAmountIn) (contracts/CrosschainForwarder.sol#340)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#110)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
                - token.safeApprove(_spender,0) (contracts/CrosschainForwarder.sol#421)
                - token.safeApprove(_spender,type()(uint256).max) (contracts/CrosschainForwarder.sol#423)
        - srcAmountOut = _swapToETHVia(_srcSwapRouter,_srcSwapCalldata) (contracts/CrosschainForwarder.sol#341)
                - (success,returnData) = _to.call{value: _msgValue}(_data) (contracts/CrosschainForwarder.sol#429)
        - srcAmountOut = _swapToERC20Via(_srcSwapRouter,_srcSwapCalldata,0,IERC20Upgradeable(_srcTokenOut)) (contracts/CrosschainForwarder.sol#342)
                - (success,returnData) = _to.call{value: _msgValue}(_data) (contracts/CrosschainForwarder.sol#429)
        External calls sending eth:
        - srcAmountOut = _swapToERC20Via(_srcSwapRouter,_srcSwapCalldata,_srcAmountIn,IERC20Upgradeable(_srcTokenOut)) (contracts/CrosschainForwarder.sol#338)
                - (success,returnData) = _to.call{value: _msgValue}(_data) (contracts/CrosschainForwarder.sol#429)
        - _lazyApprove(_srcTokenIn,_srcSwapRouter,_srcAmountIn) (contracts/CrosschainForwarder.sol#340)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
        - srcAmountOut = _swapToETHVia(_srcSwapRouter,_srcSwapCalldata) (contracts/CrosschainForwarder.sol#341)
                - (success,returnData) = _to.call{value: _msgValue}(_data) (contracts/CrosschainForwarder.sol#429)
        - srcAmountOut = _swapToERC20Via(_srcSwapRouter,_srcSwapCalldata,0,IERC20Upgradeable(_srcTokenOut)) (contracts/CrosschainForwarder.sol#342)
                - (success,returnData) = _to.call{value: _msgValue}(_data) (contracts/CrosschainForwarder.sol#429)
        Event emitted after the call(s):
        - SwapExecuted(_srcSwapRouter,_srcTokenIn,_srcAmountIn,_srcTokenOut,srcAmountOut) (contracts/CrosschainForwarder.sol#345)
Reentrancy in CrosschainForwarder._strictlySwapAndCall(address,uint256,bytes,address,bytes,address,uint256,address,address,bytes) (contracts/CrosschainForwarder.sol#207-253):
        External calls:
        - _obtainSrcTokenIn(_srcTokenIn,_srcAmountIn,_srcTokenInPermitEnvelope) (contracts/CrosschainForwarder.sol#222)
                - Permit.executePermit(address(_token),_permitEnvelope) (contracts/CrosschainForwarder.sol#351)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#110)
                - _token.safeTransferFrom(msg.sender,address(this),_amount) (contracts/CrosschainForwarder.sol#354)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
                - IERC20Permit(_tokenAddress).permit(msg.sender,address(this),permitAmount,deadline,v,r,s) (contracts/libraries/Permit.sol#16-32)
        - (srcAmountOut,msgValueAfterSwap) = _performSwap(_srcTokenIn,_srcAmountIn,msg.value,_srcSwapRouter,_srcSwapCalldata,_srcTokenOut) (contracts/CrosschainForwarder.sol#227)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#110)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
                - token.safeApprove(_spender,0) (contracts/CrosschainForwarder.sol#421)
                - token.safeApprove(_spender,type()(uint256).max) (contracts/CrosschainForwarder.sol#423)
                - (success,returnData) = _to.call{value: _msgValue}(_data) (contracts/CrosschainForwarder.sol#429)
        External calls sending eth:
        - _obtainSrcTokenIn(_srcTokenIn,_srcAmountIn,_srcTokenInPermitEnvelope) (contracts/CrosschainForwarder.sol#222)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
        - (srcAmountOut,msgValueAfterSwap) = _performSwap(_srcTokenIn,_srcAmountIn,msg.value,_srcSwapRouter,_srcSwapCalldata,_srcTokenOut) (contracts/CrosschainForwarder.sol#227)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
                - (success,returnData) = _to.call{value: _msgValue}(_data) (contracts/CrosschainForwarder.sol#429)
        Event emitted after the call(s):
        - SwapExecuted(_srcSwapRouter,_srcTokenIn,_srcAmountIn,_srcTokenOut,srcAmountOut) (contracts/CrosschainForwarder.sol#345)
                - (srcAmountOut,msgValueAfterSwap) = _performSwap(_srcTokenIn,_srcAmountIn,msg.value,_srcSwapRouter,_srcSwapCalldata,_srcTokenOut) (contracts/CrosschainForwarder.sol#227)
```

```
Reentrancy in CrosschainForwarder._strictlySwapAndCall(address,uint256,bytes,address,bytes,address,uint256,address,address,bytes) (contracts/CrosschainForwarder.sol#207-253):
        External calls:
        - _obtainSrcTokenIn(_srcTokenIn,_srcAmountIn,_srcTokenInPermitEnvelope) (contracts/CrosschainForwarder.sol#222)
                - Permit.executePermit(address(_token),_permitEnvelope) (contracts/CrosschainForwarder.sol#351)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#110)
                - _token.safeTransferFrom(msg.sender,address(this),_amount) (contracts/CrosschainForwarder.sol#354)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
                - IERC20Permit(_tokenAddress).permit(msg.sender,address(this),permitAmount,deadline,v,r,s) (contracts/libraries/Permit.sol#16-32)
        - (srcAmountOut,msgValueAfterSwap) = _performSwap(_srcTokenIn,_srcAmountIn,msg.value,_srcSwapRouter,_srcSwapCalldata,_srcTokenOut) (contracts/CrosschainForwarder.sol#227)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#110)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
                - token.safeApprove(_spender,0) (contracts/CrosschainForwarder.sol#421)
                - token.safeApprove(_spender,type()(uint256).max) (contracts/CrosschainForwarder.sol#423)
                - (success,returnData) = _to.call{value: _msgValue}(_data) (contracts/CrosschainForwarder.sol#429)
        - IERC20Upgradeable(_srcTokenOut).safeTransfer(_srcTokenRefundRecipient,refundAmount) (contracts/CrosschainForwarder.sol#244)
        External calls sending eth:
        - _obtainSrcTokenIn(_srcTokenIn,_srcAmountIn,_srcTokenInPermitEnvelope) (contracts/CrosschainForwarder.sol#222)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
        - (srcAmountOut,msgValueAfterSwap) = _performSwap(_srcTokenIn,_srcAmountIn,msg.value,_srcSwapRouter,_srcSwapCalldata,_srcTokenOut) (contracts/CrosschainForwarder.sol#227)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
                - (success,returnData) = _to.call{value: _msgValue}(_data) (contracts/CrosschainForwarder.sol#429)
        - address(_srcTokenRefundRecipient).transfer(refundAmount) (contracts/CrosschainForwarder.sol#241)
        Event emitted after the call(s):
        - Refund(_srcTokenOut,refundAmount,_srcTokenRefundRecipient) (contracts/CrosschainForwarder.sol#246)
Reentrancy in CrosschainForwarder.swapAndSendV2(address,uint256,bytes,address,bytes,address,ICrossChainForwarder.GateParams) (contracts/CrosschainForwarder.sol#110-124):
        External calls:
        - _obtainSrcTokenIn(_srcTokenIn,_srcAmountIn,_srcTokenInPermitEnvelope) (contracts/CrosschainForwarder.sol#119)
                - Permit.executePermit(address(_token),_permitEnvelope) (contracts/CrosschainForwarder.sol#351)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#110)
                - _token.safeTransferFrom(msg.sender,address(this),_amount) (contracts/CrosschainForwarder.sol#354)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
                - IERC20Permit(_tokenAddress).permit(msg.sender,address(this),permitAmount,deadline,v,r,s) (contracts/libraries/Permit.sol#16-32)
        - (srcAmountOut,msgValueAfterSwap) = _performSwap(_srcTokenIn,_srcAmountIn,msg.value,_srcSwapRouter,_srcSwapCalldata,_srcTokenOut) (contracts/CrosschainForwarder.sol#121)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#110)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
                - token.safeApprove(_spender,0) (contracts/CrosschainForwarder.sol#421)
                - token.safeApprove(_spender,type()(uint256).max) (contracts/CrosschainForwarder.sol#423)
                - (success,returnData) = _to.call{value: _msgValue}(_data) (contracts/CrosschainForwarder.sol#429)
        External calls sending eth:
        - _obtainSrcTokenIn(_srcTokenIn,_srcAmountIn,_srcTokenInPermitEnvelope) (contracts/CrosschainForwarder.sol#119)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
        - (srcAmountOut,msgValueAfterSwap) = _performSwap(_srcTokenIn,_srcAmountIn,msg.value,_srcSwapRouter,_srcSwapCalldata,_srcTokenOut) (contracts/CrosschainForwarder.sol#121)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
                - (success,returnData) = _to.call{value: _msgValue}(_data) (contracts/CrosschainForwarder.sol#429)
        Event emitted after the call(s):
        - SwapExecuted(_srcSwapRouter,_srcTokenIn,_srcAmountIn,_srcTokenOut,srcAmountOut) (contracts/CrosschainForwarder.sol#345)
                - (srcAmountOut,msgValueAfterSwap) = _performSwap(_srcTokenIn,_srcAmountIn,msg.value,_srcSwapRouter,_srcSwapCalldata,_srcTokenOut) (contracts/CrosschainForwarder.sol#121)
```

```
Reentrancy in CrosschainForwarder.swapAndSendV3(address,uint256,bytes,uint256,address,address,bytes,address,ICrossChainForwarder.GateParams) (contracts/CrosschainForwarder.sol#84-108):
        External calls:
        - _obtainSrcTokenIn(_srcTokenIn,_srcAmountIn,_srcTokenInPermitEnvelope) (contracts/CrosschainForwarder.sol#95)
                - Permit.executePermit(address(_token),_permitEnvelope) (contracts/CrosschainForwarder.sol#351)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#110)
                - _token.safeTransferFrom(msg.sender,address(this),_amount) (contracts/CrosschainForwarder.sol#354)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
                - IERC20Permit(_tokenAddress).permit(msg.sender,address(this),permitAmount,deadline,v,r,s) (contracts/libraries/Permit.sol#16-32)
        - (srcAmountInAfterFee,msgValueAfterFee) = _distributeAffiliateFee(_srcTokenIn,_srcAmountIn,_affiliateFeeAmount,_affiliateFeeRecipient) (contracts/CrosschainForwarder.sol#96)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#110)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
                - (success,None) = _affiliateFeeRecipient.call{value: _affiliateFeeAmount}() (contracts/CrosschainForwarder.sol#307)
                - IERC20Upgradeable(_srcTokenIn).safeTransfer(_affiliateFeeRecipient,_affiliateFeeAmount) (contracts/CrosschainForwarder.sol#312)
        - (srcAmountOut,msgValueAfterSwap) = _performSwap(_srcTokenIn,srcAmountInAfterFee,msgValueAfterFee,_srcSwapRouter,_srcSwapCalldata,_srcTokenOut) (contracts/CrosschainForwarder.sol#98-105)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#110)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
                - token.safeApprove(_spender,0) (contracts/CrosschainForwarder.sol#421)
                - token.safeApprove(_spender,type()(uint256).max) (contracts/CrosschainForwarder.sol#423)
                - (success,returnData) = _to.call{value: _msgValue}(_data) (contracts/CrosschainForwarder.sol#429)
        External calls sending eth:
        - _obtainSrcTokenIn(_srcTokenIn,_srcAmountIn,_srcTokenInPermitEnvelope) (contracts/CrosschainForwarder.sol#95)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
        - (srcAmountInAfterFee,msgValueAfterFee) = _distributeAffiliateFee(_srcTokenIn,_srcAmountIn,_affiliateFeeAmount,_affiliateFeeRecipient) (contracts/CrosschainForwarder.sol#96)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
                - (success,None) = _affiliateFeeRecipient.call{value: _affiliateFeeAmount}() (contracts/CrosschainForwarder.sol#307)
        - (srcAmountOut,msgValueAfterSwap) = _performSwap(_srcTokenIn,srcAmountInAfterFee,msgValueAfterFee,_srcSwapRouter,_srcSwapCalldata,_srcTokenOut) (contracts/CrosschainForwarder.sol#98-105)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
                - (success,returnData) = _to.call{value: _msgValue}(_data) (contracts/CrosschainForwarder.sol#429)
        Event emitted after the call(s):
        - SwapExecuted(_srcSwapRouter,_srcTokenIn,_srcAmountIn,_srcTokenOut,srcAmountOut) (contracts/CrosschainForwarder.sol#345)
                - (srcAmountOut,msgValueAfterSwap) = _performSwap(_srcTokenIn,srcAmountInAfterFee,msgValueAfterFee,_srcSwapRouter,_srcSwapCalldata,_srcTokenOut) (contracts/CrosschainForwarder.sol#98-105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Version constraint 0.8.7 contains known severe issues (https://solidity.readthedocs.io/en/latest/bugs.html)
        - VerbatimInvalidDeduplication
        - FullInlinerNonExpressionSplitArgumentEvaluationOrder
        - MissingSideEffectsOnSelectorAccess
        - AbiReencodingHeadOverflowWithStaticArrayCleanup
        - DirtyBytesArrayToStorage
        - DataLocationChangeInInternalOverride
        - NestedCalldataArrayAbiReencodingSizeValidation
        - SignedImmutables.
It is used by:
        - 0.8.7 (node_modules/@debridge-finance/debridge-protocol-evm-interfaces/contracts/interfaces/IDeBridgeGate.sol#2)
        - 0.8.7 (node_modules/@debridge-finance/debridge-protocol-evm-interfaces/contracts/interfaces/IDeBridgeGateExtended.sol#2)
        - 0.8.7 (contracts/CrosschainForwarder.sol#2)
        - 0.8.7 (contracts/ForwarderBase.sol#2)
        - 0.8.7 (contracts/ReceivingForwarder.sol#2)
        - 0.8.7 (contracts/libraries/SignatureUtil.sol#2)
Version constraint ^0.8.0 contains known severe issues (https://solidity.readthedocs.io/en/latest/bugs.html)
        - FullInlinerNonExpressionSplitArgumentEvaluationOrder
        - MissingSideEffectsOnSelectorAccess
        - AbiReencodingHeadOverflowWithStaticArrayCleanup
        - DirtyBytesArrayToStorage
        - DataLocationChangeInInternalOverride
        - NestedCalldataArrayAbiReencodingSizeValidation
        - SignedImmutables
        - ABIDecodeTwoDimensionalArrayMemory
        - KeccakCaching.
```

```
        - KeccakCaching.
It is used by:
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/IAccessControlUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-IERC20PermitUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/interfaces/draft-IERC1822.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Proxy.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/proxy/Proxy.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/proxy/beacon/IBeacon.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/proxy/transparent/TransparentUpgradeableProxy.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-ERC20Permit.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Counters.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/StorageSlot.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/cryptography/EIP712.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/Math.sol#4)
        - ^0.8.0 (contracts/interfaces/IPmmBase.sol#1)
        - ^0.8.0 (contracts/interfaces/IPmmSource.sol#1)
Version constraint ^0.8.2 contains known severe issues (https://solidity.readthedocs.io/en/latest/bugs.html)
        - FullInlinerNonExpressionSplitArgumentEvaluationOrder
        - MissingSideEffectsOnSelectorAccess
        - AbiReencodingHeadOverflowWithStaticArrayCleanup
        - DirtyBytesArrayToStorage
        - DataLocationChangeInInternalOverride
        - NestedCalldataArrayAbiReencodingSizeValidation
        - SignedImmutables
        - ABIDecodeTwoDimensionalArrayMemory
        - KeccakCaching.
It is used by:
        - ^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.2 (node_modules/@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#4)
Version constraint ^0.8.1 contains known severe issues (https://solidity.readthedocs.io/en/latest/bugs.html)
        - FullInlinerNonExpressionSplitArgumentEvaluationOrder
        - MissingSideEffectsOnSelectorAccess
        - AbiReencodingHeadOverflowWithStaticArrayCleanup
        - DirtyBytesArrayToStorage
        - DataLocationChangeInInternalOverride
        - NestedCalldataArrayAbiReencodingSizeValidation
        - SignedImmutables
        - ABIDecodeTwoDimensionalArrayMemory
        - KeccakCaching.
It is used by:
```

```
        - ^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4)
        - ^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4)
```

Halborn strongly recommends conducting a follow-up assessment of the project either within six months or immediately following any material changes to the codebase, whichever comes first. This approach is crucial for maintaining the project's integrity and addressing potential vulnerabilities introduced by code modifications.