HALBORN

deBridge Cross Chain Swaps - SDK

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **March 25th, 2022 - April 20th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	04/19/2022	Alpcan Onaran
0.2	Document Edits	04/19/2022	Alpcan Onaran
0.3	Document Edits	04/20/2022	Alpcan Onaran
0.4	Document Edits	04/21/2022	Alpcan Onaran
0.5	Draft Review	04/22/2022	Gabi Urrutia
1.0	Remediation Plan	06/08/2022	Alpcan Onaran
1.1	Remediation Plan Review	06/08/2022	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
Timur Guvenkaya	Halborn	Timur.Guvenkaya@halborn.com
Alpcan Onaran	Halborn	Alpcan.onaran@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

deBridge engaged Halborn to conduct a security assessment on their smart contracts beginning on March 25th, 2022 and ending April 20th, 2022. deBridge 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 four weeks for the engagement and assigned one full-time security engineer 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 contracts.

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.

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 contracts and scripts in the following:
master branch 9e3262cd63cd9d916c9de00f021543888f3568ad commit (cross-chain-swaps)
master branch 340b77c551a8cde4800ebdc4665a580938816737 commit (cross-chain-sdk)

Smart contracts:

- CrossChainForwarder.sol
- ForwarderBase.sol
- LPConnector.sol
- ReceivingForwarder.sol
- CalldataUtils.sol
- Flags.sol
- SignatureUtil.sol
- SwapCalldataUtils.sol

SDK:

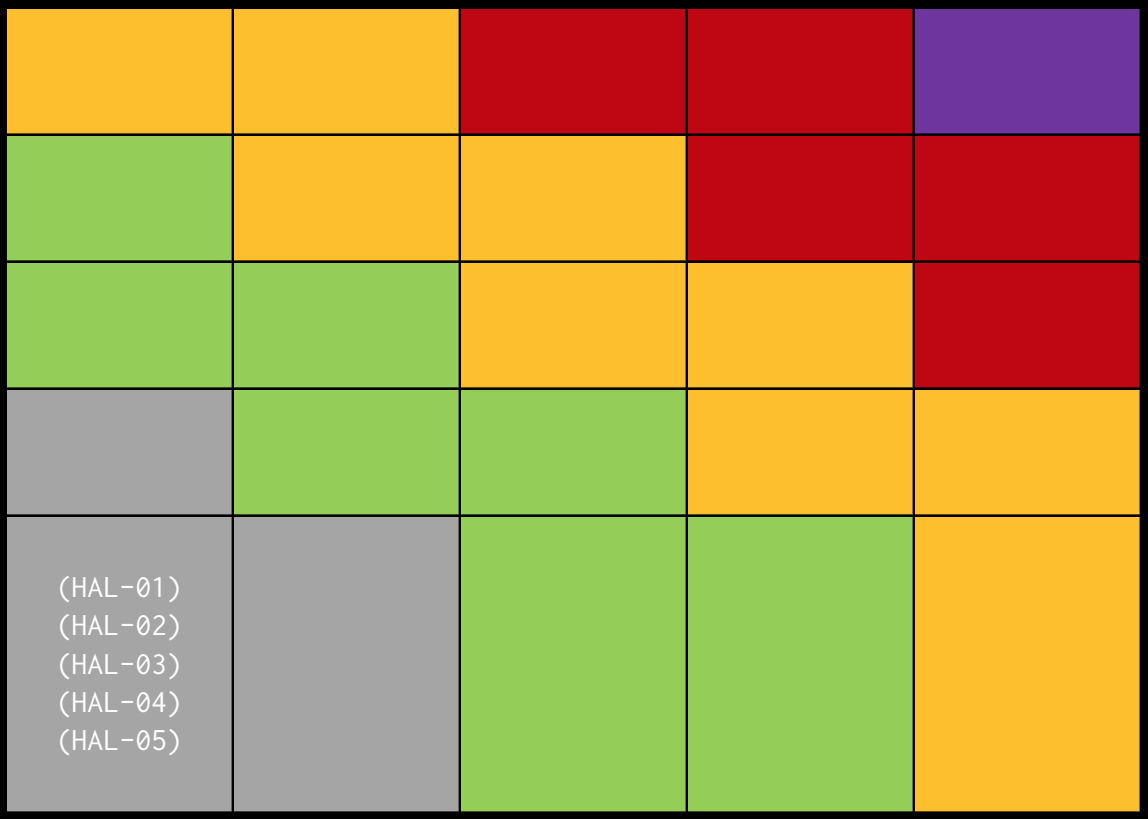
- Chain.ts
- CrossChainForwardingService.ts
- CrossChainPathFindingService.ts
- CrossChainResolver.ts
- DePair.ts
- DePairsCollection.ts
- Icons.ts
- RecommendExecutionFeeService.ts
- SDK.ts
- SDKError.ts
- SwapBuildingService.ts

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	0	5

LIKELIHOOD

IMPACT



SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL-01 SAME PAIR OF TOKENS CAN BE USED	Informational	NOT APPLICABLE
HAL-02 MISSING PAUSEABLE FUNCTIONALITY	Informational	NOT APPLICABLE
HAL-03 UNUSED RETURNS	Informational	FUTURE RELEASE
HAL-04 COGNITIVE COMPLEXITY OF FUNCTION IS TOO HIGH	Informational	ACKNOWLEDGED
HAL-05 POSSIBLE MISUSE OF PUBLIC FUNCTIONS	Informational	SOLVED - 12/05/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) SAME PAIR OF TOKENS CAN BE USED - INFORMATIONAL

Description:

It is observed that in forward() and swapAndSend() functions inside ReceivingForwarder.sol and in CrosschainForwarder.sol does not check if source token type equals to destination token type and accepts same pair of tokens as input.

This type of bugs may lead to critical attack vectors or griefing attacks.

Code Location:

ReceivingForwarder.sol

- forward

Listing 1: ReceivingForwarder.sol (Lines 30,33)

```
29     function forward(
30         address _dstTokenIn,
31         address _router,
32         bytes memory _routerCalldata,
33         address _dstTokenOut,
34         address _fallbackAddress
35     ) external payable override{
36         if (_dstTokenIn == NATIVE_TOKEN) {
37             return _forwardFromETH(
38                 _router,
39                 _routerCalldata,
40                 _dstTokenOut,
41                 _fallbackAddress
42             );
43         }
44         else {
45             return _forwardFromERC20(
46                 IERC20Upgradeable(_dstTokenIn),
47                 _router,
```

```
48                  _routerCalldata,
49                  _dstTokenOut,
50                  _fallbackAddress
51              );
```

CrosschainForwarder.sol

- swapAndSend

Listing 2: LPConnector.sol (Lines 47,52)

```
46      function swapAndSend(
47          address _srcTokenIn,
48          uint _srcAmountIn,
49          bytes memory _srcTokenInPermit,
50          address _srcSwapRouter,
51          bytes calldata _srcSwapCalldata,
52          address _srcTokenOut,
53          bytes calldata _dstDetails
54      ) external payable override {
55          if (!supportedRouters[_srcSwapRouter]) revert
↳ NotSupportedRouter();
56
57          uint ethBalanceBefore = address(this).balance - msg.value;
58          uint srcAmountOut;
59
60          if (_srcTokenIn == NATIVE_TOKEN) {
61              _validateSrcETHIn(_srcAmountIn);
62              srcAmountOut = _swapToERC20Via(
63                  _srcSwapRouter,
64                  _srcSwapCalldata,
65                  _srcAmountIn,
66                  IERC20Upgradeable(_srcTokenOut)
67              );
68          }
69          else {
70              ...
```

Proof of Concept:

In the Proof of Concept below, we created a test script which gives same input token type for source and destination.

Listing 3: Proof of concept

```
    it('Same pair of tokens', async () => {
      await state.usdToken.instance.mint(state.user.address,usd2v
↳ (10));
      console.log(await state.usdToken.instance.balanceOf(state.
↳ user.address));

      await state.receivingForwarder.instance.connect(state.user)
        .forward(
          state.usdToken.instance.address, // _wrappedToken
          state.dex.instance.address, // _router
          SAMPLE_CALLDATA.swap.sample, // _routerCalldata
          state.usdToken.instance.address, // _targetToken
          state.user.address, // _fallbackAddress
        );

        console.log(await state.usdToken.instance.balanceOf(state.
↳ user.address));
    });
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to ensure that source and destination tokens are not of the same type.

Remediation Plan:

NOT APPLICABLE: The issue is marked as **not applicable** by the DeBridge team with an explanation: Both CrosschainForwarder and ReceivingForwarder work as relayers: they take tokenIn from msg.sender, swap it to tokenOut via the swapRouter, then pass the resulting tokenOut along with deBridgeGate or the destination address. It seems that there is nothing wrong with the situation when both tokens are the same. Moreover, this opens broad possibilities for arbitrage.

3.2 (HAL-02) MISSING PAUSEABLE FUNCTIONALITY - INFORMATIONAL

Description:

Even code that has been thoroughly audited and tested may contain bugs or defective code parts. These flaws are frequently undetected until they are employed in an attack by an opponent. Because immutability is one of the basic characteristics of the blockchain, it is difficult to correct if a critical fault is discovered. While some patterns (such as the Proxy Delegate pattern) allow for upgradeable code to some extent, these solutions normally take a long time to implement and come into action. Before the update is transmitted to the network, the attackers could continue with their malicious actions and cause harm.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to add an option to disable critical contract functionality in case of an emergency.

Remediation Plan:

NOT APPLICABLE: The issue is marked as **not applicable** by the DeBridge team with an explanation: ReceivingForwarder and CrosschainForwarder communicate with the deBridgeGate contract. If we ever need to stop the contracts, we can either stop deBridgeGate or quickly upgrade the proxy to a fixed or paused implementation.

3.3 (HAL-03) UNUSED RETURN VALUES - INFORMATIONAL

Description:

The return value of an external call is not stored in a local or state variable. In CrosschainForwarder.sol, LPConnector.sol and ReceivingForwarder.sol contracts, there are few instances where external methods are being called using approve and return values(bool) are being ignored.

Code Location:

CrosschainForwarder.sol

- swapAndSend

Listing 4: CrosschainForwarder.sol

```
74     srcTokenIn.approve(_srcSwapRouter, srcAmountOut);
```

Listing 5: CrosschainForwarder.sol

```
82     srcTokenIn.approve(_srcSwapRouter, srcAmountOut);
```

- _sendToBridge

Listing 6: CrosschainForwarder.sol

```
215     IERC20Upgradeable(token).approve(address(deBridgeGate), 0);
```

LPConnector.sol

- swapFrom

Listing 7: LPConnector.sol

```
77      _tokenIn.approve(target.pool, _amountIn);
```

Listing 8: LPConnector.sol

```
86      _tokenIn.approve(target.router, _amountIn);
```

Listing 9: LPConnector.sol

```
98      _tokenIn.approve(target.pool, 0);
```

ReceivingForwarder.sol

- _forwardFromERC20

Listing 10: ReceivingForwarder.sol

```
97      dstTokenIn.approve(_router, dstTokenInAmount);
```

Listing 11: ReceivingForwarder.sol

```
109     dstTokenIn.approve(_router, 0);
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to add return value checks to avoid unexpected crash of the contract. Return value check will help in handling the exceptions better way.

Remediation Plan:

PENDING: The issue was acknowledged by the DeBridge team and will be fixed later.

3.4 (HAL-04) COGNITIVE COMPLEXITY OF FUNCTION IS TOO HIGH - INFORMATIONAL

Description:

Cognitive Complexity is a measure of how hard the control flow of a function is to understand. Functions with high Cognitive Complexity will be difficult to maintain.

CrossChainPathFindingService.ts
- refreshDst

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to refactor this function to reduce its cognitive complexity by separating it to different functions.

Remediation Plan:

ACKNOWLEDGED: The DeBridge team acknowledged this finding.

3.5 (HAL-05) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL

Description:

In public functions, array arguments are immediately copied to memory, while external functions can read directly from `calldata`. Reading `calldata` is cheaper than memory allocation. Public functions need to write the arguments to memory because public functions may be called internally. Internal calls are passed internally by pointers to memory. Thus, the function expects its arguments being located in memory when the compiler generates the code for an internal function.
Furthermore, methods do not necessarily have to be public if they are only called within the contract-in such case they should be marked `internal`.

Code Location:

LPConnector.sol
- getUnwrapToken

Listing 12: LPConnector.sol

```
51     function getUnwrapToken(address _wrappedToken) public view
↳ returns (address) {
52         return pools[_wrappedToken].jToken;
53     }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to mark this function as external, if it will not be called inside the contract.

Remediation Plan:

SOLVED: The issue was solved by the DeBridge team.

- Fix Commit

3.6 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance coverage of certain areas of the scoped contract. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their abi and binary formats. 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.

Results:

All of the findings were either detected during manual code review or false positive.

```
DummyPool._swap(IERC20,IERC20,address,address,uint256) (contracts/mock/DummyPool.sol#112-128) ignores return value by srcToken.transferFrom(from,address(this),amount) (contracts/mock/DummyPool.sol#122)
DummyPool._swap(IERC20,IERC20,address,address,uint256) (contracts/mock/DummyPool.sol#112-128) ignores return value by dstToken.transfer(to,amount - amount / 100) (contracts/mock/DummyPool.sol#124)
TargetTokenHolder.fallback(bytes) (contracts/mock/TargetTokenHolder.sol#22-27) ignores return value by IERC20(targetToken).transferFrom(msg.sender,address(this),amount) (contracts/mock/TargetTokenHolder.
sol#25)
TargetTokenHolder.obtain(address) (contracts/mock/TargetTokenHolder.sol#29-36) ignores return value by wrappedToken.transferFrom(msg.sender,address(this),wrappedToken.balanceOf(msg.sender)) (contracts/no
ck/TargetTokenHolder.sol#31-35)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

CrosschainForwarder._sendToBridge(address,uint256,uint256,bytes).autoParams (contracts/CrosschainForwarder.sol#195) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

LPConnector._swapFrom(IERC20,uint256,uint256) (contracts/LPConnector.sol#56-103) ignores return value by _tokenIn.approve(target.pool,_amountIn) (contracts/LPConnector.sol#77)
LPConnector._swapFrom(IERC20,uint256,uint256) (contracts/LPConnector.sol#56-103) ignores return value by ICurve(target.pool).exchange_underlying(target.i,target.j,_amountIn,_minAmountOut) (contracts/LPCo
nnector.sol#79-84)
LPConnector._swapFrom(IERC20,uint256,uint256) (contracts/LPConnector.sol#56-103) ignores return value by _tokenIn.approve(target.router,_amountIn) (contracts/LPConnector.sol#86)
LPConnector._swapFrom(IERC20,uint256,uint256) (contracts/LPConnector.sol#56-103) ignores return value by ICurve(target.router).exchange_underlying(target.pool,target.i,target.j,_amountIn,_minAmountOut,ad
dress(this)) (contracts/LPConnector.sol#88-95)
LPConnector._swapFrom(IERC20,uint256,uint256) (contracts/LPConnector.sol#56-103) ignores return value by _tokenIn.approve(target.pool,0) (contracts/LPConnector.sol#98)
DummyBridge.send(address,uint256,uint256,bytes,bytes,bool,uint32,bytes) (contracts/mock/DummyBridge.sol#26-84) ignores return value by callProxy.callERC20(_tokenAddress,fallbackAddress,receiver,autoParam
s.data,autoParams.flags,abi.encodePacked(msg.sender),getChainId()) (contracts/mock/DummyBridge.sol#68-76)
DummyCallProxy.callERC20(address,address,address,bytes,uint256,bytes,uint256) (contracts/mock/DummyCallProxy.sol#18-51) ignores return value by token.approve(_receiver,amount) (contracts/mock/DummyCallPr
oxy.sol#30)
DummyCallProxy.callERC20(address,address,address,bytes,uint256,bytes,uint256) (contracts/mock/DummyCallProxy.sol#18-51) ignores return value by token.approve(_receiver,0) (contracts/mock/DummyCallProxy.s
ol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

DummyPool.constructor(address,address)._tokenA (contracts/mock/DummyPool.sol#18) lacks a zero-check on :
        - tokenA = _tokenA (contracts/mock/DummyPool.sol#19)
DummyPool.constructor(address,address)._tokenB (contracts/mock/DummyPool.sol#18) lacks a zero-check on :
        - tokenB = _tokenB (contracts/mock/DummyPool.sol#20)
TargetTokenHolder.constructor(address).targetToken_ (contracts/mock/TargetTokenHolder.sol#16) lacks a zero-check on :
        - targetToken = targetToken_ (contracts/mock/TargetTokenHolder.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in DummyCallProxy.callERC20(address,address,address,bytes,uint256,bytes,uint256) (contracts/mock/DummyCallProxy.sol#18-51):
        External calls:
        - token.approve(_receiver,amount) (contracts/mock/DummyCallProxy.sol#30)
        State variables written after the call(s):
        - _result = externalCall(_receiver,0,_data,_nativeSender,_chainIdFrom,_flags.getFlag(Flags.PROXY_WITH_SENDER)) (contracts/mock/DummyCallProxy.sol#32-39)
                - submissionChainIdFrom = _chainIdFrom (contracts/mock/DummyCallProxy.sol#64)
                - submissionChainIdFrom = 0 (contracts/mock/DummyCallProxy.sol#86)
        - _result = externalCall(_receiver,0,_data,_nativeSender,_chainIdFrom,_flags.getFlag(Flags.PROXY_WITH_SENDER)) (contracts/mock/DummyCallProxy.sol#32-39)
                - submissionNativeSender = _nativeSender (contracts/mock/DummyCallProxy.sol#65)
                - submissionNativeSender = (contracts/mock/DummyCallProxy.sol#87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#174-194) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#186-189)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#201-221) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#213-216)
ForwarderBase._externalCall(address,bytes,uint256) (contracts/ForwarderBase.sol#27-43) uses assembly
        - INLINE ASM (contracts/ForwarderBase.sol#32-42)
LPConnector.externalCall(address,bytes) (contracts/LPConnector.sol#105-123) uses assembly
        - INLINE ASM (contracts/LPConnector.sol#110-122)
CalldataUtils.getSig(bytes) (contracts/libraries/CalldataUtils.sol#7-11) uses assembly
        - INLINE ASM (contracts/libraries/CalldataUtils.sol#8-10)
CalldataUtils.slice(bytes,uint256,uint256) (contracts/libraries/CalldataUtils.sol#13-81) uses assembly
        - INLINE ASM (contracts/libraries/CalldataUtils.sol#22-78)
SignatureUtil.parseSignature(bytes,uint256) (contracts/libraries/SignatureUtil.sol#32-49) uses assembly
        - INLINE ASM (contracts/libraries/SignatureUtil.sol#41-45)
SignatureUtil.toUint256(bytes,uint256) (contracts/libraries/SignatureUtil.sol#51-61) uses assembly
        - INLINE ASM (contracts/libraries/SignatureUtil.sol#58-60)
SwapCalldataUtils.getAmount(bytes) (contracts/libraries/SwapCalldataUtils.sol#33-47) uses assembly
        - INLINE ASM (contracts/libraries/SwapCalldataUtils.sol#43-45)
DummyBridge.getChainId() (contracts/mock/DummyBridge.sol#20-24) uses assembly
        - INLINE ASM (contracts/mock/DummyBridge.sol#21-23)
DummyBridge.send(address,uint256,uint256,bytes,bytes,bool,uint32,bytes) (contracts/mock/DummyBridge.sol#26-84) uses assembly
        - INLINE ASM (contracts/mock/DummyBridge.sol#51-53)
        - INLINE ASM (contracts/mock/DummyBridge.sol#57-59)
DummyCallProxy.externalCall(address,uint256,bytes,bytes,uint256,bool) (contracts/mock/DummyCallProxy.sol#53-89) uses assembly
        - INLINE ASM (contracts/mock/DummyCallProxy.sol#68-82)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

FINDINGS & TECH DETAILS

```
Different versions of Solidity is used:
	- Version used: ['0.8.7', '^0.8.0', '^0.8.1', '^0.8.7']
	- ^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/proxy/utils/Initializable.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/utils/SafeERC20Upgradeable.sol#4)
	- ^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.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/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-IERC20Permit.sol#4)
	- ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4)
	- ^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4)
	- ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
	- 0.8.7 (contracts/CrosschainForwarder.sol#2)
	- 0.8.7 (contracts/ForwarderBase.sol#2)
	- ^0.8.7 (contracts/LPConnector.sol#2)
	- 0.8.7 (contracts/ReceivingForwarder.sol#2)
	- ^0.8.7 (contracts/interfaces/ICrossChainForwarder.sol#2)
	- ^0.8.7 (contracts/interfaces/ICurve.sol#2)
	- 0.8.7 (contracts/interfaces/IDeBridgeGate.sol#2)
	- ^0.8.7 (contracts/interfaces/IReceivingForwarder.sol#2)
	- ^0.8.7 (contracts/libraries/CalldataUtils.sol#2)
	- 0.8.7 (contracts/libraries/Flags.sol#2)
	- 0.8.7 (contracts/libraries/SignatureUtil.sol#2)
	- ^0.8.7 (contracts/libraries/SwapCalldataUtils.sol#2)
	- ^0.8.7 (contracts/mock/DummyBridge.sol#1)
	- ^0.8.7 (contracts/mock/DummyCallProxy.sol#1)
	- ^0.8.7 (contracts/mock/DummyPool.sol#1)
	- ^0.8.7 (contracts/mock/DummyToken.sol#1)
	- ^0.8.7 (contracts/mock/SwapCalldataMock.sol#2)
	- ^0.8.7 (contracts/mock/TargetTokenHolder.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```

```
Flags.setFlag(uint256,uint256,bool) (contracts/libraries/Flags.sol#30-37) is never used and should be removed
LPConnector._addUnwrapPool(address,int128,address,int128,address) (contracts/LPConnector.sol#21-33) is never used and should be removed
LPConnector._addUnwrapRouter(address,address,int128,address,int128,address) (contracts/LPConnector.sol#35-49) is never used and should be removed
LPConnector._swapFrom(IERC20,uint256,uint256) (contracts/LPConnector.sol#56-103) is never used and should be removed
LPConnector.externalCall(address,bytes) (contracts/LPConnector.sol#105-123) is never used and should be removed
SignatureUtil.getUnsignedMsg(bytes32) (contracts/libraries/SignatureUtil.sol#13-15) is never used and should be removed
SignatureUtil.splitSignature(bytes) (contracts/libraries/SignatureUtil.sol#19-30) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/IAccessControlUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65):
	- (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#63)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#128-139):
	- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#137)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166):
	- (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#164)
Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#60-65):
	- (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#63)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#128-139):
	- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#157-166):
	- (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#164)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#184-193):
	- (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
Low level call in ForwarderBase._safeTransferETH(address,uint256) (contracts/ForwarderBase.sol#50-53):
	- (success) = to.call{value: value}(new bytes(0)) (contracts/ForwarderBase.sol#51)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
Function AccessControlUpgradeable.__AccessControl_init() (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#51-52) is not in mixedCase
Function AccessControlUpgradeable.__AccessControl_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#54-55) is not in mixedCase
Variable AccessControlUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#235) is not in mixedCase
Function ContextUpgradeable.__Context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is not in mixedCase
Variable ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#36) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#24-25) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#27-28) is not in mixedCase
Variable ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#41) is not in mixedCase
Function IERC20Permit.DOMAIN_SEPARATOR() (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol#59) is not in mixedCase
Parameter CrosschainForwarder.initialize(IDeBridgeGate)._deBridgeGate (contracts/CrosschainForwarder.sol#39) is not in mixedCase
Parameter CrosschainForwarder.swapAndSend(address,uint256,bytes,address,bytes,address,bytes)._srcTokenIn (contracts/CrosschainForwarder.sol#47) is not in mixedCase
Parameter CrosschainForwarder.swapAndSend(address,uint256,bytes,address,bytes,address,bytes)._srcAmountIn (contracts/CrosschainForwarder.sol#48) is not in mixedCase
Parameter CrosschainForwarder.swapAndSend(address,uint256,bytes,address,bytes,address,bytes)._srcTokenInPermit (contracts/CrosschainForwarder.sol#49) is not in mixedCase
Parameter CrosschainForwarder.swapAndSend(address,uint256,bytes,address,bytes,address,bytes)._srcSwapRouter (contracts/CrosschainForwarder.sol#50) is not in mixedCase
Parameter CrosschainForwarder.swapAndSend(address,uint256,bytes,address,bytes,address,bytes)._srcSwapCalldata (contracts/CrosschainForwarder.sol#51) is not in mixedCase
Parameter CrosschainForwarder.swapAndSend(address,uint256,bytes,address,bytes,address,bytes)._srcTokenOut (contracts/CrosschainForwarder.sol#52) is not in mixedCase
Parameter CrosschainForwarder.swapAndSend(address,uint256,bytes,address,bytes,address,bytes)._dstDetails (contracts/CrosschainForwarder.sol#53) is not in mixedCase
Parameter CrosschainForwarder.updateSupportedRouter(address,bool)._srcSwapRouter (contracts/CrosschainForwarder.sol#225) is not in mixedCase
Parameter CrosschainForwarder.updateSupportedRouter(address,bool)._isSupported (contracts/CrosschainForwarder.sol#226) is not in mixedCase
Parameter LPConnector.getUnwrapToken(address)._wrappedToken (contracts/LPConnector.sol#51) is not in mixedCase
Parameter ReceivingForwarder.forward(address,address,bytes,address,address)._dstTokenIn (contracts/ReceivingForwarder.sol#30) is not in mixedCase
Parameter ReceivingForwarder.forward(address,address,bytes,address,address)._router (contracts/ReceivingForwarder.sol#31) is not in mixedCase
Parameter ReceivingForwarder.forward(address,address,bytes,address,address)._routerCalldata (contracts/ReceivingForwarder.sol#32) is not in mixedCase
Parameter ReceivingForwarder.forward(address,address,bytes,address,address)._dstTokenOut (contracts/ReceivingForwarder.sol#33) is not in mixedCase
Parameter ReceivingForwarder.forward(address,address,bytes,address,address)._fallbackAddress (contracts/ReceivingForwarder.sol#34) is not in mixedCase
Function ICurve.exchange_underlying(int128,int128,uint256,uint256) (contracts/interfaces/ICurve.sol#5-10) is not in mixedCase
Parameter ICurve.exchange_underlying(int128,int128,uint256,uint256)._min_dy (contracts/interfaces/ICurve.sol#9) is not in mixedCase
Function ICurve.exchange_underlying(address,int128,int128,uint256,uint256,address) (contracts/interfaces/ICurve.sol#22-29) is not in mixedCase
Parameter ICurve.exchange_underlying(address,int128,int128,uint256,uint256,address)._min_dy (contracts/interfaces/ICurve.sol#27) is not in mixedCase
Parameter CalldataUtils.getSig(bytes)._data (contracts/libraries/CalldataUtils.sol#7) is not in mixedCase
Parameter CalldataUtils.slice(bytes,uint256,uint256)._bytes (contracts/libraries/CalldataUtils.sol#14) is not in mixedCase
Parameter CalldataUtils.slice(bytes,uint256,uint256)._start (contracts/libraries/CalldataUtils.sol#15) is not in mixedCase
Parameter CalldataUtils.slice(bytes,uint256,uint256)._length (contracts/libraries/CalldataUtils.sol#16) is not in mixedCase
Parameter Flags.getFlag(uint256,uint256)._packedFlags (contracts/libraries/Flags.sol#17) is not in mixedCase
Parameter Flags.getFlag(uint256,uint256)._flag (contracts/libraries/Flags.sol#17) is not in mixedCase
Parameter Flags.setFlag(uint256,uint256,bool)._packedFlags (contracts/libraries/Flags.sol#31) is not in mixedCase
Parameter Flags.setFlag(uint256,uint256,bool)._flag (contracts/libraries/Flags.sol#32) is not in mixedCase
Parameter Flags.setFlag(uint256,uint256,bool)._value (contracts/libraries/Flags.sol#33) is not in mixedCase
Parameter SignatureUtil.getUnsignedMsg(bytes32)._submissionId (contracts/libraries/SignatureUtil.sol#13) is not in mixedCase
Parameter SignatureUtil.splitSignature(bytes)._signature (contracts/libraries/SignatureUtil.sol#19) is not in mixedCase
Parameter SignatureUtil.parseSignature(bytes,uint256)._signatures (contracts/libraries/SignatureUtil.sol#32) is not in mixedCase
Parameter SignatureUtil.toUint256(bytes,uint256)._bytes (contracts/libraries/SignatureUtil.sol#51) is not in mixedCase
Parameter SignatureUtil.toUint256(bytes,uint256)._offset (contracts/libraries/SignatureUtil.sol#51) is not in mixedCase
Parameter SwapCalldataUtils.patch(bytes,uint256)._data (contracts/libraries/SwapCalldataUtils.sol#15) is not in mixedCase
Parameter SwapCalldataUtils.getAmount(bytes)._data (contracts/libraries/SwapCalldataUtils.sol#33) is not in mixedCase
Parameter DummyBridge.send(address,uint256,uint256,bytes,bytes,bool,uint32,bytes)._tokenAddress (contracts/mock/DummyBridge.sol#27) is not in mixedCase
Parameter DummyBridge.send(address,uint256,uint256,bytes,bytes,bool,uint32,bytes)._amount (contracts/mock/DummyBridge.sol#28) is not in mixedCase
Parameter DummyBridge.send(address,uint256,uint256,bytes,bytes,bool,uint32,bytes)._receiver (contracts/mock/DummyBridge.sol#30) is not in mixedCase
Parameter DummyBridge.send(address,uint256,uint256,bytes,bytes,bool,uint32,bytes)._autoParams (contracts/mock/DummyBridge.sol#34) is not in mixedCase
Parameter DummyCallProxy.callERC20(address,address,address,bytes,uint256,bytes,uint256)._token (contracts/mock/DummyCallProxy.sol#19) is not in mixedCase
Parameter DummyCallProxy.callERC20(address,address,address,bytes,uint256,bytes,uint256)._reserveAddress (contracts/mock/DummyCallProxy.sol#20) is not in mixedCase
```

THANK YOU FOR CHOOSING

// HALBORN