// HALBORN

DeBridge - DLN EVM Bridge 1.3.0

Smart Contract Security Assessment

Prepared by: **Halborn**

Date of Engagement: **October 16th, 2023 - November 1st, 2023**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE
0.1	Document Creation	10/16/2023
0.2	Document Updates	10/31/2023
0.3	Draft Review	11/01/2023
0.4	Draft Review	11/01/2023
1.0	Remediation Plan	11/24/2023
1.1	Remediation Plan Review	11/24/2023

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 contracts beginning on October 16th, 2023 and ending on November 1st, 2023. The security assessment was scoped to the smart contracts provided in the following GitHub repositories:

- debridge-finance/dln-evm/.

1.2 ASSESSMENT SUMMARY

The team at Halborn was provided two weeks for the engagement and assigned a full-time security engineer to verify the security of the smart contracts. The security 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 assessment is to:

- Ensure that smart contract functions operate as intended.
- Identify potential security issues with the smart contracts.

In summary, Halborn identified some security risks that were mostly addressed by the DeBridge team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual 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 the code and can quickly identify items that do not follow the security best practices. The following phases and associated tools were used during the assessment:

- Research into architecture and purpose.
- Smart contract manual code review and walkthrough.
- Graphing out functionality and contract logic/connectivity/functions. (solgraph)
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes.
- Manual testing by custom scripts.
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment. (Foundry)

2. 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.

2.1 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:

Exploitability 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:

$$E = \prod m_e$$

2.2 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)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Integrity (I)	None (I:N)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Availability (A)	None (A:N)	0
	Low (A:L)	0.25
	Medium (A:M)	0.5
	High (A:H)	0.75
	Critical	1
Deposit (D)	None (D:N)	0
	Low (D:L)	0.25
	Medium (D:M)	0.5
	High (D:H)	0.75
	Critical (D:C)	1
Yield (Y)	None (Y:N)	0
	Low (Y:L)	0.25
	Medium: (Y:M)	0.5
	High: (Y:H)	0.75
	Critical (Y:H)	1

Impact I is calculated using the following formula:

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

2.3 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.

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

2.4 SCOPE

1. IN-SCOPE TREE & COMMIT :

The security assessment was scoped to the following PR:
GitHub repository: debridge-finance/dln-evm/
Audited Commit ID: 96973018b442c21c6f0ece4c300d97abd01bdf9b
Fix Commit ID: 649c00dd72bf19817d81a8c8320b0a558c304423

Smart contracts in scope:

- DlnBase.sol
- DlnDestination.sol
- DlnSource.sol
- DlnExternalCallAdapter.sol
- AAVECallExecutor.sol
- WidoCallExecutor.sol
- ExternalCallExecutor.sol
- ExternalCallExecutorBase.sol
- BytesLib.sol
- DlnExternalCallLib.sol
- DlnOrderLib.sol
- EncodeSolanaDlnMessage.sol
- SafeCast.sol

3. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	7	3

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) MULTIPLE ORDERS COULD GET PERMANENTLY STUCK IN THE DLNSOURCE CONTRACT DUE TO THE CLAIMBATCHUNLOCK() FUNCTION	Low (3.1)	RISK ACCEPTED
(HAL-02) EXTERNALCALLEXECUTOR PROHIBITED SELECTORS ARE NOT VALIDATED CORRECTLY	Low (2.5)	SOLVED - 11/24/2023
(HAL-03) MAKERS COULD ABUSE THE EXTERNAL CALL FEATURE TO FORCE TAKERS INTO FULFILLING EXTRA ORDERS ON THEIR BEHALF	Low (2.5)	RISK ACCEPTED
(HAL-04) INCOMPATIBILITY WITH REVERT ON ZERO TRANSFER TOKENS IN THE CLAIMBATCHUNLOCK() FUNCTION	Low (4.2)	SOLVED - 11/24/2023
(HAL-05) TAKERS COULD BE GAS GRIEFED BY MAKERS IF THEY DO NOT SET AN ACCURATE GAS LIMIT UPON FULFILLING AN ORDER	Low (2.5)	RISK ACCEPTED
(HAL-06) AFFILIATE FEE INCOMPATIBILITY WITH NON-STANDARD ERC20 TOKENS	Low (2.5)	RISK ACCEPTED
(HAL-07) SOME MULTISIGNATURE WALLETS REQUIRE MORE THAN 2300 GAS TO RECEIVE A NATIVE ASSET TRANSFER	Low (2.5)	SOLVED - 11/24/2023
(HAL-08) CHECK EFFECTS INTERACTION PATTERN IS NOT RESPECTED IN THE EXECUTECALL() FUNCTION	Informational (0.0)	SOLVED - 11/24/2023
(HAL-09) MAKER COULD TEMPORARY BLOCK ANY TAKER FROM FULFILLING HIS ORDERS	Informational (0.0)	ACKNOWLEDGED
(HAL-10) STATE VARIABLES MISSING IMMUTABLE MODIFIER	Informational (0.0)	ACKNOWLEDGED

FINDINGS & TECH DETAILS

4.1 (HAL-01) MULTIPLE ORDERS COULD GET PERMANENTLY STUCK IN THE DLNSOURCE CONTRACT DUE TO THE CLAIMBATCHUNLOCK() FUNCTION - LOW (3.1)

Commit IDs affected:
- 96973018b442c21c6f0ece4c300d97abd01bdf9b

Description:

In the DlnDestination contract, the function sendBatchEvmUnlock() is used to unlock multiple orders that have been filled on the EVM chain, but haven't been unlocked yet. This function sends a batch unlock request using the _orderIds provided:

Listing 1: DlnDestination.sol (Line 332)

```
311 function sendBatchEvmUnlock(
312     bytes32[] memory _orderIds,
313     address _beneficiary,
314     uint256 _executionFee
315 ) external payable nonReentrant whenNotPaused {
316     if (_orderIds.length == 0) revert UnexpectedBatchSize();
317     if (_orderIds.length > maxOrderCountPerBatchEvmUnlock) revert
 ↳ UnexpectedBatchSize();
318
319     uint256 giveChainId;
320     uint256 length = _orderIds.length;
321     for (uint256 i; i < length; ++i) {
322         uint256 currentGiveChainId = _prepareOrderStateForUnlock(
 ↳ _orderIds[i], DlnOrderLib.ChainEngine.EVM);
323         if (i == 0) {
324             giveChainId = currentGiveChainId;
325         }
326         else {
327             // giveChainId must be the same for all orders
```

```
328             if (giveChainId != currentGiveChainId) revert
 ↳ MismatchGiveChainId();
329         }
330     }
331     // encode function that will be called in target chain
332     bytes memory claimUnlockMethod = _encodeBatchClaimUnlock(
 ↳ _orderIds, _beneficiary);
333
334     //send crosschain message through deBridgeGate
335     bytes32 submissionId = _sendCrossChainMessage(
336         giveChainId, // _chainIdTo
337         abi.encodePacked(_beneficiary),
338         _executionFee,
339         claimUnlockMethod
340     );
341
342     for (uint256 i; i < length; ++i) {
343         emit SentOrderUnlock(_orderIds[i], abi.encodePacked(
 ↳ _beneficiary), submissionId);
344     }
345 }
```

After this call is executed, the different validators will generate the signatures, which can be used by the taker/keepers to call the DeBridgeGate.claim() function in the receiving chain. This function, internally, calls the DlnSource.claimBatchUnlock() function:

Listing 2: DlnSource.sol (Lines 273,300)

```
268 /**
269  * @dev Processes a batch of unlock orders originating from the
 ↳ order's take chain.
270  * @param _orderIds An array containing the IDs of the orders to
 ↳ be unlocked.
271  * @param _beneficiary The address that will receive the assets
 ↳ from the unlocked orders.
272  * # Restrictions
273  * This function can only be called through the debridge's
 ↳ external call mechanism,
274  * ensuring it's invoked by a validated native sender.
275  */
276 function claimBatchUnlock(bytes32[] memory _orderIds, address
 ↳ _beneficiary)
```

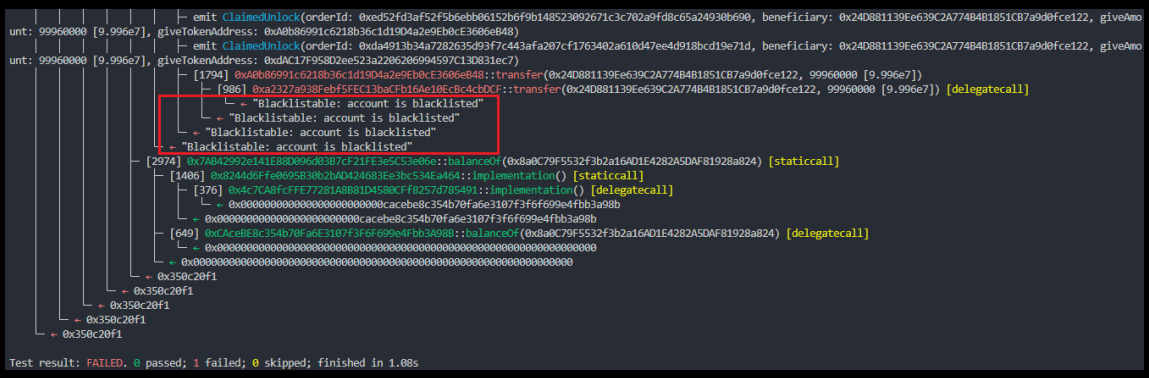
```
277     external
278     nonReentrant
279     whenNotPaused
280 {
281     uint256 submissionChainIdFrom = _onlyDlnDestinationAddress();
282     //  store the give token of the order in the batch. If next
 ↳ order has different give token we will pay to beneficiary
283     // and change distinctGiveToken to new one.
284     address distinctGiveToken;
285     uint256 distinctGiveTokenAmount;
286     uint256 length = _orderIds.length;
287     for (uint256 i; i < length; ++i) {
288         bytes32 orderId = _orderIds[i];
289         GiveOrderState storage orderState = giveOrders[orderId];
290 
291         // the give address of the first order is distinct
292         if (i == 0) {
293             distinctGiveToken = address(orderState.
 ↳ giveTokenAddress);
294         }
295 
296         uint256 amountToPay = _claimUnlock(orderId, _beneficiary,
 ↳ submissionChainIdFrom, false);
297         if (amountToPay != 0) {
298             // transfer tokens for beneficiary if token address
 ↳ changed
299             if (distinctGiveToken != address(orderState.
 ↳ giveTokenAddress)) {
300                 _safeTransferEthOrToken(distinctGiveToken,
 ↳ _beneficiary, distinctGiveTokenAmount);
301                 distinctGiveToken = address(orderState.
 ↳ giveTokenAddress);
302                 distinctGiveTokenAmount = amountToPay;
303             } else {
304                 distinctGiveTokenAmount += amountToPay;
305             }
306         }
307     }
308     if (distinctGiveTokenAmount != 0) {
309         _safeTransferEthOrToken(distinctGiveToken, _beneficiary,
 ↳ distinctGiveTokenAmount);
310     }
311 }
```

Although, if one single transfer fails, the whole transaction would revert. As such, we can think of the following scenario:

1. Maker creates 2 orders on the Ethereum mainnet.
   Order1:
   giveTokenAddress = address(contract_USDCETH);
   giveAmount = 100e6;
   takeTokenAddress = abi.encodePacked(contract_USDCARB);
   takeAmount = 100e6;
   takeChainId = 42161; (ARBITRUM)
   Order2:
   giveTokenAddress = address(contract_USDTETH);
   giveAmount = 100e6;
   takeTokenAddress = abi.encodePacked(contract_USDTARB);
   takeAmount = 100e6;
   takeChainId = 42161; (ARBITRUM)
2. Taker fulfills both orders in Arbitrum.
3. Taker calls DlnDestination.sendBatchEvmUnlock() in Arbitrum. Sent() event is emitted, and the signatures are created by the validators.
4. Taker calls in the Ethereum mainnet DeBridgeGateETH.claim(). It reverts with "Blacklistable: account is blacklisted" error as the Taker is blacklisted in the USDC contract.

Impact: Both order assets (in this case 100 USDC/USDT) are permanently stuck in the protocol. The taker has lost those funds.

Proof of Concept - Code Snippet



BVSS:

AO:A/AC:L/AX:L/C:N/I:L/A:N/D:L/Y:N/R:N/S:U (3.1)

Recommendation:

It is recommended to make use of a `try/catch` block to perform the token transfers. In case that one of the transfers fails, it will not block the others.

Another solution could be allowing the taker to initiate unlock batch again, excluding the problematic order from the batch.

Remediation Plan:

RISK ACCEPTED: The DeBridge team accepts this risk, stating that the taker has complete control over which tokens they execute, which includes the ability to conduct comprehensive checks before initiating the `sendBatchEvmUnlock()` process. Specifically, takers can verify the address of the unlock beneficiary against any blacklist or other security criteria. This preemptive checking mechanism significantly reduces the risk of assets getting stuck due to the unlocking of unverified or potentially problematic tokens.

In a future update, the DeBridge team will add functionality, so the taker can withdraw an unclaimed amount or initiate the unlock batch again.

4.2 (HAL-02) EXTERNALCALLEXECUTOR PROHIBITED SELECTORS ARE NOT VALIDATED CORRECTLY - LOW (2.5)

Commit IDs affected:
- 96973018b442c21c6f0ece4c300d97abd01bdf9b

Description:

In the ExternalCallExecutor contract, the function _isValideData() is used to validate that the callData passed in an external call does not include the following selectors:

Listing 3: ExternalCallExecutor.sol (Lines 118,121,122,123,124)

```
117 function _isValideData(bytes memory _data) internal returns (bool)
 ↳  {
118     bytes4 functionSelector = _toBytes4(_data, 0);
119
120     bytes4[4] memory prohibitedSelectors = [
121         bytes4(0x095ea7b3), // approve
122         bytes4(0x23b872dd), // transferFrom
123         bytes4(0xa9059cbb), // transfer
124         bytes4(0x39509351) // increaseAllowance
125     ];
126
127     for (uint256 i; i < prohibitedSelectors.length; ++i) {
128         if (prohibitedSelectors[i] == functionSelector) {
129             emit ProhibitedFunctionSelector(functionSelector);
130             return false;
131         }
132     }
133     return true;
134 }
```

The function selector is retrieved from the calldata using the _toBytes4() function:

Listing 4: ExternalCallExecutor.sol (Line 152)

```
136 /**
137  * @notice Converts the first 4 bytes starting at a specific
 ↳ position in a byte array to a bytes4 type.
138  * @dev This function uses assembly for efficient memory
 ↳ operations.
139  * @param _bytes The byte array to be converted.
140  * @param _start The position in the byte array to start the
 ↳ conversion.
141  * @return The first 4 bytes starting at the `_start` position in
 ↳ the byte array as a bytes4 type.
142  * throws If there are less than 4 bytes starting at the `_start`
 ↳ position in the byte array.
143  */
144 function _toBytes4(
145     bytes memory _bytes,
146     uint256 _start
147 ) internal pure returns (bytes4) {
148     require(_bytes.length >= _start + 4, "toBytes4_outOfBounds");
149     bytes4 tempUint;
150 
151     assembly {
152         tempUint := mload(add(add(_bytes, 0x4), _start))
153     }
154 
155     return tempUint;
156 }
```

This function, though, does not work as intended and does not correctly retrieve the function selector from the calldata. The `_toBytes4()` function will always return the `0x00000000` selector and, as such, the `_isValideData` will always return `true`. This occurs because the first 32 bytes of the `_bytes` parameter contains the length, and then the actual data follows. In order to read the actual selector, `0x4` should be changed for `0x20` (32).

The impact of this incorrect validation is that any user could cause a Denial of Service of the `ExternalCallExecutor` for tokens like USDT. See the Proof of Concept below.

Proof of Concept:

Denial Of Service to the Compound USDT Market on Ethereum Mainnet.

Code Snippet

1. Attacker creates the following order in Arbitrum:

Listing 5

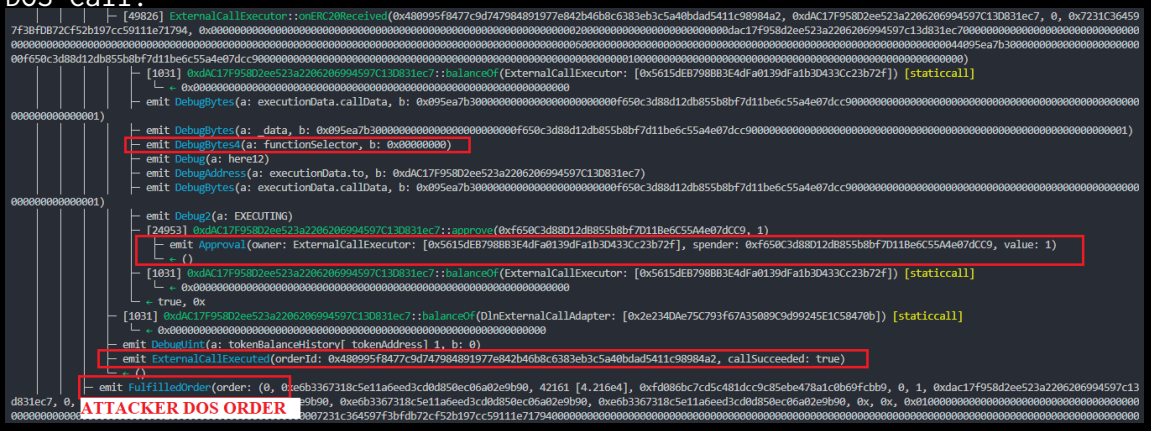
```
_orderCreation.giveTokenAddress = contract_USDTARB;
_orderCreation.giveAmount = 0;
_orderCreation.takeTokenAddress = abi.encodePacked(
↳ contract_USDTETH);
_orderCreation.takeAmount = 0;
_orderCreation.takeChainId = 1;
_orderCreation.receiverDst = abi.encodePacked(user1);
_orderCreation.givePatchAuthoritySrc = user1;
_orderCreation.orderAuthorityAddressDst = abi.encodePacked(user1);
_orderCreation.allowedTakerDst = '';
_orderCreation.externalCall = abi.encodePacked(uint8(1), abi.
↳ encode(_dataEnvelope));
_orderCreation.allowedCancelBeneficiarySrc = '';

DlnExternalCallLib.ExternalCallPayload memory _payload1;
_payload1.to = address(contract_USDTETH);
_payload1.txGas = uint32(0);
// Compound USDT Market address => 0
↳ xf650C3d88D12dB855b8bf7D11Be6C55A4e07dCC9
_payload1.callData = abi.encodePacked(abi.encodeWithSignature("
↳ approve(address,uint256)", address(0
↳ xf650C3d88D12dB855b8bf7D11Be6C55A4e07dCC9), 1), hex'');

DlnExternalCallLib.ExternalCallEnvelopV1 memory _dataEnvelope;
_dataEnvelope.fallbackAddress = user3;
_dataEnvelope.executorAddress = address(0);
_dataEnvelope.executionFee = uint160(0);
_dataEnvelope.allowDelayedExecution = false;
_dataEnvelope.requireSuccessfullExecution = true;
_dataEnvelope.payload = abi.encode(_payload1);
```

Notice that the _payload1.callData sets an allowance of 1 for the Compound

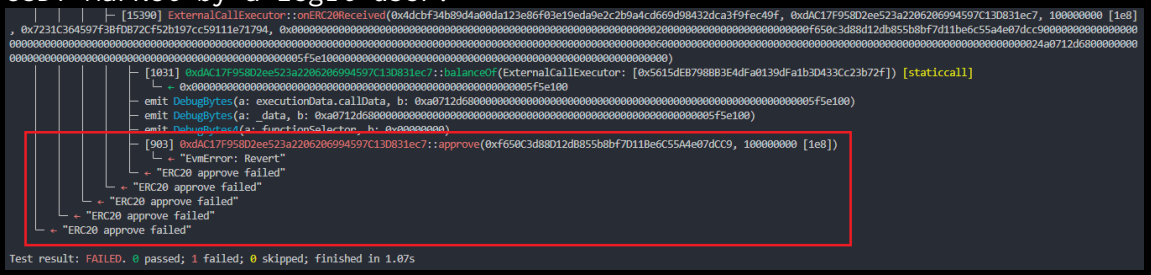
USDT Market address.

2. The order is fulfilled and afterward the ExternalCallExecutor contract has a remaining allowance of 1 with the Compound USDT Market.
3. Future orders that will interact with the Compound USDT Market through an external call will revert, as USDT requires having a zero allowance before setting a new allowance.

DOS Call:



Future order that external calls the mint() function of the Compound USDT Market by a legit user:



BVSS:

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

Recommendation:

It is recommended to update the _toBytes4() function as shown below:

Listing 6: ExternalCallExecutor.sol (Line 152)

```
136 /**
137  * @notice Converts the first 4 bytes starting at a specific
 ↳ position in a byte array to a bytes4 type.
138  * @dev This function uses assembly for efficient memory
 ↳ operations.
139  * @param _bytes The byte array to be converted.
140  * @param _start The position in the byte array to start the
 ↳ conversion.
141  * @return The first 4 bytes starting at the `_start` position in
 ↳ the byte array as a bytes4 type.
142  * throws If there are less than 4 bytes starting at the `_start`
 ↳ position in the byte array.
143  */
144 function _toBytes4(
145     bytes memory _bytes,
146     uint256 _start
147 ) internal pure returns (bytes4) {
148     require(_bytes.length >= _start + 4, "toBytes4_outOfBounds");
149     bytes4 tempUint;
150
151     assembly {
152         tempUint := mload(add(add(_bytes, 0x20), _start))
153     }
154
155     return tempUint;
156 }
```

Remediation Plan:

SOLVED: The DeBridge team solved the issue by implementing the recommended solution.

Commit ID : 826d21961da4f8bab83d52c8e7d81250694e7f5a.

4.3 (HAL-03) MAKERS COULD ABUSE THE EXTERNAL CALL FEATURE TO FORCE TAKERS INTO FULFILLING EXTRA ORDERS ON THEIR BEHALF - LOW (2.5)

Commit IDs affected:
- 96973018b442c21c6f0ece4c300d97abd01bdf9b

Description:

With the current external call implementation, the following scenario where makers could abuse takers to fulfill orders on their behalf is possible:

1. Order1 is created by a random maker. This order has no external call.
2. Malicious maker creates Order2. This order has allowDelayedExecution set to true and requireSuccessfullExecution also set to true and the following payload as the external call:

Listing 7

```
DlnExternalCallLib.ExternalCallPayload memory _payload;
_payload.to = address(contract_OrderFulfillerETH);
_payload.txGas = uint32(0);
_payload.callData = abi.encodeWithSignature("fulfillOrder()");
```

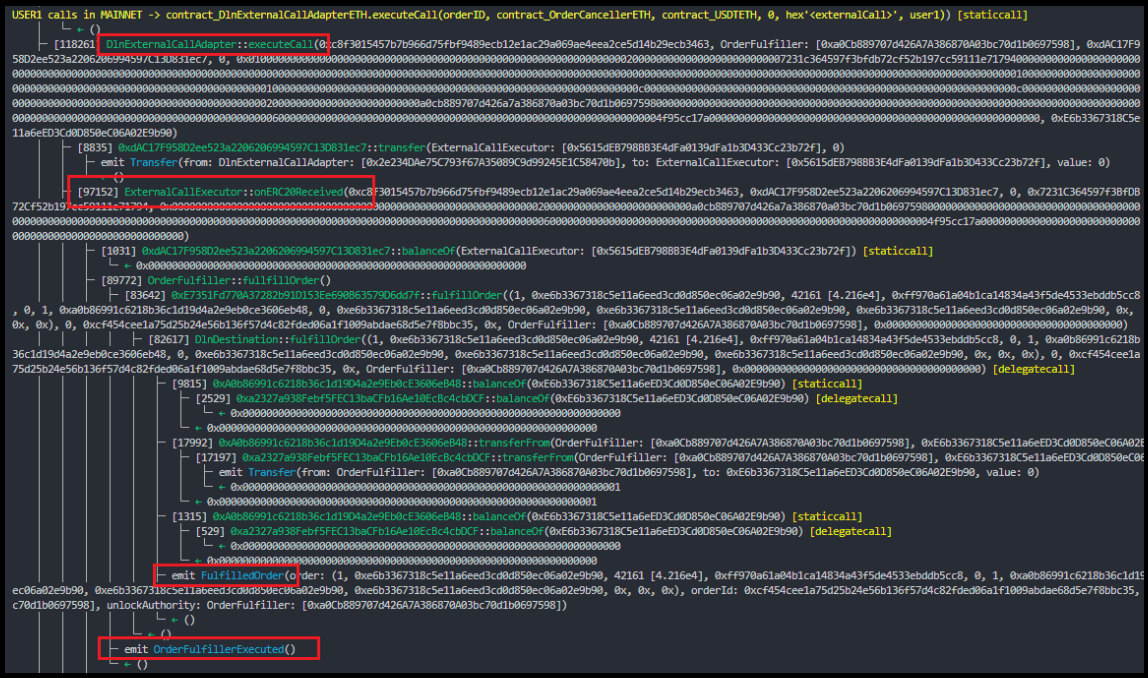
3. The contract_OrderFulfillerETH code can be found below:

Listing 8: (Lines 17,20)

```
pragma solidity ^0.8.17;

import {DlnOrderLib} from "@root/libraries/DlnOrderLib.sol";
```

```
4  import {DlnDestination} from "@root/DLN/DlnDestination.sol";
5
6  contract OrderFulfiller {
7
8      DlnDestination public dlnDestinationContract;
9      DlnOrderLib.Order public order;
10     uint256 public fulFillAmount;
11     bytes32 public orderId;
12     event OrderFulfillerExecuted();
13
14     constructor(){}
15
16     function fulfillOrder() public {
17         // NOTE THAT THIS CAN BE A LOOP THAT FULFILLS MULTIPLE
↳ ORDERS, NOT JUST ONE
18         dlnDestinationContract.fulfillOrder(order, fulFillAmount,
↳ orderId, '', address(this), address(0));
19         emit OrderFulfillerExecuted();
20         // NEW ORDER WITH ANOTHER EXTERNAL CALL THAT FULFILLS ALL
↳ THE OTHER ORDERS CAN BE CREATED HERE BY THIS CONTRACT
21     }
22
23     function setParameters(address _dlnDestinationContract,
↳ DlnOrderLib.Order memory _order, uint256 _fulFillAmount, bytes32
↳ _orderId) public {
24         dlnDestinationContract = DlnDestination(
↳ _dlnDestinationContract);
25         order = _order;
26         fulFillAmount = _fulFillAmount;
27         orderId = _orderId;
28     }
29 }
```

4. The malicious maker calls `contract_OrderFulfillerETH.setParameters()` to set all the parameters needed to fulfill the order.
5. Taker calls `fulfillOrder()`.
6. Taker then calls `executeCall()` receiving the execution fee for Order2, although the malicious maker will receive the give/take token profit for all the orders executed in the external call.



In this Proof of Concept a single order was fulfilled by the external call, but an "endless loop" could be achieved by creating an order with an external call that points to this contract function which:

1. Fulfills the rest of the orders of the protocol.
2. Creates a new order with an external call that points once again to this function.

BVSS:

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

Recommendation:

It is recommended to add an extra check to the fulfillOrder() function. Any call to this function should revert if it is coming from an executeCall() from the DlnExternalCallAdapter contract.

Remediation Plan:

RISK ACCEPTED: The DeBridge team accepted this risk, stating that they disagree with the notion of risks associated with executing a different

order through a dln external call. In a dln ext call, any interaction can be encapsulated, whether it's a swap through 1inch or an NFT purchase. The DeBridge team is confident that such behavior poses no threat to our services. Our services process orders/dln-ext calls based on an evaluation that includes gas expenses of transaction execution.

On the other hand, Halborn believes that this scenario can be abused to "defeat" or "abuse" the incentive system provided to the takers for fulfilling orders.

4.4 (HAL-04) INCOMPATIBILITY WITH REVERT ON ZERO TRANSFER TOKENS IN THE CLAIMBATCHUNLOCK() FUNCTION - LOW (4.2)

Commit IDs affected:
- 96973018b442c21c6f0ece4c300d97abd01bdf9b

Description:

As already mentioned, in the issue MULTIPLE ORDERS COULD GET PERMANENTLY STUCK IN THE DLNSOURCE CONTRACT DUE TO THE CLAIMBATCHUNLOCK()FUNCTION **if a single order claim fails, it will act as a blocker for the rest of the orders in the batch.**

Another problem related to this is that the claimBatchUnlock() function does not prevent zero value transfers:

Listing 9: DlnSource.sol (Line 300)

```
268 /**
269  * @dev Processes a batch of unlock orders originating from the
 ↳ order's take chain.
270  * @param _orderIds An array containing the IDs of the orders to
 ↳ be unlocked.
271  * @param _beneficiary The address that will receive the assets
 ↳ from the unlocked orders.
272  * # Restrictions
273  * This function can only be called through the debridge's
 ↳ external call mechanism,
274  * ensuring it's invoked by a validated native sender.
275  */
276 function claimBatchUnlock(bytes32[] memory _orderIds, address
 ↳ _beneficiary)
277     external
278     nonReentrant
279     whenNotPaused
280 {
```

```
281     uint256 submissionChainIdFrom = _onlyDlnDestinationAddress();
282     //  store the give token of the order in the batch. If next
 ↳ order has different give token we will pay to beneficiary
283     // and change distinctGiveToken to new one.
284     address distinctGiveToken;
285     uint256 distinctGiveTokenAmount;
286     uint256 length = _orderIds.length;
287     for (uint256 i; i < length; ++i) {
288         bytes32 orderId = _orderIds[i];
289         GiveOrderState storage orderState = giveOrders[orderId];
290
291         // the give address of the first order is distinct
292         if (i == 0) {
293             distinctGiveToken = address(orderState.
 ↳ giveTokenAddress);
294         }
295
296         uint256 amountToPay = _claimUnlock(orderId, _beneficiary,
 ↳ submissionChainIdFrom, false);
297         if (amountToPay != 0) {
298             // transfer tokens for beneficiary if token address
 ↳ changed
299             if (distinctGiveToken != address(orderState.
 ↳ giveTokenAddress)) {
300                 _safeTransferEthOrToken(distinctGiveToken,
 ↳ _beneficiary, distinctGiveTokenAmount);
301                 distinctGiveToken = address(orderState.
 ↳ giveTokenAddress);
302                 distinctGiveTokenAmount = amountToPay;
303             } else {
304                 distinctGiveTokenAmount += amountToPay;
305             }
306         }
307     }
308     if (distinctGiveTokenAmount != 0) {
309         _safeTransferEthOrToken(distinctGiveToken, _beneficiary,
 ↳ distinctGiveTokenAmount);
310     }
311 }
```

In the line 300, it is not checked that distinctGiveTokenAmount is a non-zero value.

Moreover, this check is also missing here, within the createOrder() call which would cause the initial order creation to revert:

Listing 10: DlnBase.sol (Line 142)

```
134 function _safeTransferFrom(
135     address _tokenAddress,
136     address _from,
137     address _to,
138     uint256 _amount
139 ) internal {
140     IERC20Upgradeable token = IERC20Upgradeable(_tokenAddress);
141     uint256 balanceBefore = token.balanceOf(_to);
142     token.safeTransferFrom(_from, _to, _amount);
143     // Received real amount
144     uint256 receivedAmount = token.balanceOf(_to) - balanceBefore;
145     if (_from != _to && _amount != receivedAmount) revert
 ↳ MismatchedTransferAmount();
146 }
```

Although, as any token can be used on the bridge since there is no whitelist, it is recommended to enforce here also this non-zero transfer amount check.

BVSS:

AO:A/AC:L/AX:M/C:N/I:M/A:N/D:M/Y:N/R:N/S:U (4.2)

Recommendation:

It is recommended to only perform the _safeTransferEthOrToken() call if distinctGiveTokenAmount is a non-zero value in order to prevent any revert caused by tokens that do not support zero value transfers.

Remediation Plan:

SOLVED: The DeBridge team solved the issue by implementing the recommended solution.

Commit ID : e91d1114d27d3bfb7a5f41507101fa90c3bbd111.

4.5 (HAL-05) TAKERS COULD BE GAS GRIEFED BY MAKERS IF THEY DO NOT SET AN ACCURATE GAS LIMIT UPON FULFILLING AN ORDER - LOW (2.5)

Commit IDs affected:
- 96973018b442c21c6f0ece4c300d97abd01bdf9b

Description:

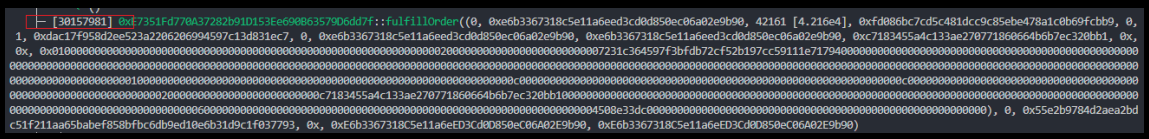
With the current implementation, a maker could create an order with `requireSuccessfullExecution` set to `true` and an external call that points to a contract with the following code:

Listing 11: GasDrainer.sol

```
1 contract GasDrainer {
2
3     uint256 public constant GAS_TO_DRAIN = 30000000;
4     uint256 public a;
5
6     constructor(){}
7
8     function gasDrain() public {
9         uint256 gasStart = gasleft();
10        while((gasStart - gasleft()) < GAS_TO_DRAIN){
11            a = gasStart - gasleft();
12        }
13    }
14 }
```

In this case, if the taker fulfills the external call without setting a proper gas limit, the maker could drain all the gas of the transaction.

Code Snippet



BVSS:

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

Recommendation:

It is recommended to advise takers to set accurate gas limits when fulfilling orders with external calls.

Remediation Plan:

RISK ACCEPTED: The DeBridge team accepts this risk, stating that their service process orders/dln-ext calls based on an evaluation that includes gas expenses of transaction execution.

4.6 (HAL-06) AFFILIATE FEE INCOMPATIBILITY WITH NON-STANDARD ERC20 TOKENS - LOW (2.5)

Commit IDs affected:
- 96973018b442c21c6f0ece4c300d97abd01bdf9b

Description:

In the DlnSource contract, the function _claimUnlock() sends the affiliateFee trhough a standard ERC20.transfer():

Listing 12: DlnSource.sol (Lines 591-593)

```
560 function _claimUnlock(bytes32 _orderId, address _beneficiary,
 ↳ uint256 _submissionChainIdFrom, bool _allowActualTransfer)
 ↳ internal returns (uint256 amountToPay) {
561     GiveOrderState storage orderState = giveOrders[_orderId];
562     if (orderState.status != OrderGiveStatus.Created) {
563         unexpectedOrderStatusForClaim[_orderId] = _beneficiary;
564         emit UnexpectedOrderStatusForClaim(_orderId, orderState.
 ↳ status, _beneficiary);
565         return 0;
566     }
567     // a circuit breaker in case DlnDestination has been
 ↳ compromised and is sending claim_unlock commands on behalf
568     // of another chain
569     if (orderState.takeChainId != _submissionChainIdFrom) {
570         emit CriticalMismatchChainId(_orderId, _beneficiary,
 ↳ orderState.takeChainId, _submissionChainIdFrom);
571         return 0;
572     }
573     amountToPay = orderState.giveAmount + givePatches[_orderId];
574     orderState.status = OrderGiveStatus.ClaimedUnlock;
575     address giveTokenAddress =  address(orderState.
 ↳ giveTokenAddress);
576     if (_allowActualTransfer) {
577         _safeTransferEthOrToken(giveTokenAddress, _beneficiary,
 ↳ amountToPay);
578     }
```

```
579     // send affiliateFee to affiliateFee beneficiary
580     if (orderState.affiliateAmount > 0) {
581         bool success;
582 
583         if (giveTokenAddress == address(0)) {
584             (success, ) = orderState.affiliateBeneficiary.call{
 ↳ value: orderState.affiliateAmount, gas: 2300}(new bytes(0));
585             if (!success) {
586                 unclaimedAffiliateETHFees[orderState.
 ↳ affiliateBeneficiary] += orderState.affiliateAmount;
587                 emit UnclaimedAffiliateFees(_orderId, address(0),
 ↳ orderState.affiliateAmount);
588             }
589         }
590         else {
591             try IERC20Upgradeable(giveTokenAddress).transfer(
592                 orderState.affiliateBeneficiary,
593                 orderState.affiliateAmount)
594             {
595                 // Successful transfer
596             }
597             catch (bytes memory /*lowLevelData*/)
598             {
599                 unclaimedERC20AffiliateFees[giveTokenAddress][
 ↳ orderState.affiliateBeneficiary] += orderState.affiliateAmount;
600                 emit UnclaimedAffiliateFees(_orderId,
 ↳ giveTokenAddress, orderState.affiliateAmount);
601             }
602 
603             success = true;
604         }
605 
606         if (success) {
607             emit AffiliateFeePaid(
608                 _orderId,
609                 orderState.affiliateBeneficiary,
610                 orderState.affiliateAmount,
611                 giveTokenAddress
612             );
613         }
614     }
615     emit ClaimedUnlock(
616         _orderId,
617         _beneficiary,
```

```
618             amountToPay,
619             giveTokenAddress
620         );
621         // Collected fee
622         collectedFee[giveTokenAddress] += orderState.percentFee;
623         collectedFee[address(0)] += orderState.nativeFixFee;
624     }
```

As such, this call will always revert with tokens that do not follow the ERC20 standard, like USDT (which does not return a boolean upon a transfer call).

Any affiliteFee that is paid with a token like USDT will fail. As this call is within a try catch block, this value will be stored in the unclaimedERC20AffiliateFees mapping. Although, even if this value is accounted in the smart contract, currently there is no functionality implemented to retrieve it.

BVSS:

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

Recommendation:

It is recommended to use safeTransfer() instead of transfer() in all the ERC20 transfer calls.

Remediation Plan:

RISK ACCEPTED: The DeBridge team accepts this risk, stating that safeTransfer() cannot be invoked in a try catch.

On the other hand, Halborn states that as safeTransfer() is an internal function it cannot be invoked directly in a try catch block, although it is possible to implement something like this in order to address this issue:

Listing 13: Example solution

```
try this.attemptTransfer(token, origin, beneficiary, amount) {}
↳ catch {
    // TODO FAIL LOGIC
}

function attemptTransfer(IERC20 token, address origin, address
↳ beneficiary, uint256 amount) external {
    require(msg.sender == address(this)); // this function should
↳ be called only by this contract
    token.safeTransferFrom(origin, beneficiary, amount);
}

```

4.7 (HAL-07) SOME MULTISIGNATURE WALLETS REQUIRE MORE THAN 2300 GAS TO RECEIVE A NATIVE ASSET TRANSFER - LOW (2.5)

Commit IDs affected:
- 96973018b442c21c6f0ece4c300d97abd01bdf9b

Description:

In the _claimUnlock() function, a native asset transfer limited to 2300 gas is performed in order to transfer the affiliate fee:

Listing 14: DlnSource.sol (Line 584)

```
560 function _claimUnlock(bytes32 _orderId, address _beneficiary,
 ↳ uint256 _submissionChainIdFrom, bool _allowActualTransfer)
 ↳ internal returns (uint256 amountToPay) {
561     GiveOrderState storage orderState = giveOrders[_orderId];
562     if (orderState.status != OrderGiveStatus.Created) {
563         unexpectedOrderStatusForClaim[_orderId] = _beneficiary;
564         emit UnexpectedOrderStatusForClaim(_orderId, orderState.
 ↳ status, _beneficiary);
565         return 0;
566     }
567     // a circuit breaker in case DlnDestination has been
 ↳ compromised and is sending claim_unlock commands on behalf
568     // of another chain
569     if (orderState.takeChainId != _submissionChainIdFrom) {
570         emit CriticalMismatchChainId(_orderId, _beneficiary,
 ↳ orderState.takeChainId, _submissionChainIdFrom);
571         return 0;
572     }
573     amountToPay = orderState.giveAmount + givePatches[_orderId];
574     orderState.status = OrderGiveStatus.ClaimedUnlock;
575     address giveTokenAddress =  address(orderState.
 ↳ giveTokenAddress);
576     if (_allowActualTransfer) {
```

```
577         _safeTransferEthOrToken(giveTokenAddress, _beneficiary,
 ↳ amountToPay);
578     }
579     // send affiliateFee to affiliateFee beneficiary
580     if (orderState.affiliateAmount > 0) {
581         bool success;
582 
583         if (giveTokenAddress == address(0)) {
584             (success, ) = orderState.affiliateBeneficiary.call{
 ↳ value: orderState.affiliateAmount, gas: 2300}(new bytes(0));
585             if (!success) {
586                 unclaimedAffiliateETHFees[orderState.
 ↳ affiliateBeneficiary] += orderState.affiliateAmount;
587                 emit UnclaimedAffiliateFees(_orderId, address(0),
 ↳ orderState.affiliateAmount);
588             }
589         }
590         else {
591             try IERC20Upgradeable(giveTokenAddress).transfer(
592                 orderState.affiliateBeneficiary,
593                 orderState.affiliateAmount)
594             {
595                 // Successful transfer
596             }
597             catch (bytes memory /*lowLevelData*/)
598             {
599                 unclaimedERC20AffiliateFees[giveTokenAddress][
 ↳ orderState.affiliateBeneficiary] += orderState.affiliateAmount;
600                 emit UnclaimedAffiliateFees(_orderId,
 ↳ giveTokenAddress, orderState.affiliateAmount);
601             }
602 
603             success = true;
604         }
605 
606         if (success) {
607             emit AffiliateFeePaid(
608                 _orderId,
609                 orderState.affiliateBeneficiary,
610                 orderState.affiliateAmount,
611                 giveTokenAddress
612             );
613         }
614     }
```

```
615     emit ClaimedUnlock(
616         _orderId,
617         _beneficiary,
618         amountToPay,
619         giveTokenAddress
620     );
621     // Collected fee
622     collectedFee[giveTokenAddress] += orderState.percentFee;
623     collectedFee[address(0)] += orderState.nativeFixFee;
624 }
```

Some multi-signature wallets require more than 2300 gas to receive a transfer of the native asset. For example, Gnosis Safe supports forwarding via fallback. Using transfer with a Safe's multisig can lead to gas depletion and failed transfers.

References:

- Article: why cannot I transfer Ether from a contract into a safe
- Blog post: Stop using solidity's transfer now

BVSS:

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

Recommendation:

It is recommended to increase the gas limit in the `orderState.affiliateBeneficiary.call()` in order to also support these kinds of wallets.

Remediation Plan:

SOLVED: The DeBridge team solved the issue by adding functionality to withdraw unclaimed fees.

Commit ID : 8c7c6f3712c2e8b4d73f6a250d51dc298a8bf829.

4.8 (HAL-08) CHECK EFFECTS INTERACTION PATTERN IS NOT RESPECTED IN THE EXECUTECALL() FUNCTION - INFORMATIONAL (0.0)

Commit IDs affected:
- 96973018b442c21c6f0ece4c300d97abd01bdf9b

Description:

In the contract DlnExternalCallAdapter, the function does not follow the check, effects, interaction pattern as the externalCallStatus[callId] is updated after the actual _execute() call:

Listing 15: DlnExternalCallAdapter.sol (Line 194)

```
169 function executeCall(
170     bytes32 _orderId,
171     address _callAuthority,
172     address _tokenAddress,
173     uint256 _tokenAmount,
174     bytes calldata _externalCall,
175     address _rewardBeneficiary
176 ) external nonReentrant whenNotPaused {
177     bytes32 callId = getCallId(
178         _orderId,
179         _callAuthority,
180         _tokenAddress,
181         _tokenAmount,
182         _externalCall
183     );
184
185     if (externalCallStatus[callId] != CallStatus.Created) revert
 ↳ InvalideState();
186
187     _execute(
188         _orderId,
189         _tokenAddress,
190         _tokenAmount,
```

```
191             _externalCall,
192             _rewardBeneficiary
193         );
194         externalCallStatus[callId] = CallStatus.Executed;
195     }
```

This does not entail any security risk, as all the functions of this contract contains the nonReentrant modifier. Although, it is still recommended to update the externalCallStatus[callId] mapping to CallStatus.Executed before the _execute() call.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:U (0.0)

Recommendation:

It is recommended to update the externalCallStatus[callId] mapping to CallStatus.Executed before the _execute() call in the DlnExternalCallAdapter.executeCall() function.

Remediation Plan:

SOLVED: The DeBridge team solved the issue by implementing the recommended solution.

Commit ID : 649c00dd72bf19817d81a8c8320b0a558c304423.

4.9 (HAL-09) MAKER COULD TEMPORARY BLOCK ANY TAKER FROM FULFILLING HIS ORDERS - INFORMATIONAL (0.0)

Commit IDs affected:
- 96973018b442c21c6f0ece4c300d97abd01bdf9b

Description:

As per the current implementation, makers can control if the takers can actually fulfill an order by setting `requireSuccessfullExecution` to `true` and `allowDelayedExecution` to `false` and setting an external call that points to a deployed contract in the take chain that the maker had previously deployed. Then, the maker can decide whether it reverts or not by, for example, updating the value of a state variable within the deployed contract.

This does not suppose any security risk, but could represent an unintended protocol functionality.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:U (0.0)

Recommendation:

Consider if this functionality is intended in the DLN EVM Bridge protocol.

Remediation Plan:

ACKNOWLEDGED: The DeBridge team acknowledged this finding.

4.10 (HAL-10) STATE VARIABLES MISSING IMMUTABLE MODIFIER - INFORMATIONAL (0.0)

Commit IDs affected:
- 96973018b442c21c6f0ece4c300d97abd01bdf9b

Description:

The immutable keyword was added to Solidity in 0.6.5. State variables can be marked immutable which causes them to be read-only, but only assignable in the constructor. The following state variables are missing the immutable modifier:

WidoCallExecutor.sol
- Line 15: address public widoRouter;
- Line 16: address public widoTokenManager;

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:C (0.0)

Recommendation:

It is recommended to add the immutable modifier to the state variables mentioned.

Remediation Plan:

ACKNOWLEDGED: The DeBridge team acknowledged this finding.

RECOMMENDATIONS OVERVIEW

1. It is recommended to make use of a `try/catch` block to perform the token transfers in the `claimBatchUnlock()` function. Another possible solution could be allowing the taker to initiate the unlock batch again, excluding the problematic order from the batch.
2. Update the `ExternalCallExecutor._toBytes4()` function as suggested.
3. Add an extra check to the `fulfillOrder()` function. Any call to this function should revert if it is coming from an `executeCall()` from the `DlnExternalCallAdapter` contract.
4. Only perform the `_safeTransferEthOrToken()` call if `distinctGiveTokenAmount` is a non-zero value in the `claimBatchUnlock()` function.
5. Advice takers to set accurate gas limits when fulfilling orders with external calls.
6. Use `safeTransfer()` instead of `transfer()` in all the `ERC20` transfer calls.
7. Increase the gas limit in the `orderState.affiliateBeneficiary.call()` in order to support all types of multisignature wallets that may need more than 2300 gas to receive a transfer of a native asset.
8. Update the `externalCallStatus[callId]` mapping to `CallStatus.Executed` before the `_execute()` call in the `DlnExternalCallAdapter.executeCall()` function.
9. Add the `immutable` modifier to the state variables mentioned in the `WidoCallExecutor` contract.

UPGRADE TO 1.3.0 VERSION CHECKS

1. Lack of any storage collision:

The following state variable was added to the DlnSource contract respecting the storage layout:
- unclaimedERC20AffiliateFees

DlnSource storage layout, current version. It can be found here

Label	Slot (Offset)	Visibility	Type	Value/Items
BPS_DENOMINATOR	constant	public	uint256	10000
DEFAULT_ADMIN_ROLE	constant	public	bytes32	0x0000000000000000000000000000000000000000000000000
EVM_ADDRESS_LENGTH	constant	public	uint256	20
GOVMONITORING_ROLE	constant	public	bytes32	0x2b36fa99e118fa8485d488becf749a974743fbeb6
MAX_ADDRESS_LENGTH	constant	public	uint256	255
SOLANA_ADDRESS_LENGTH	constant	public	uint256	32
_ENTERED	constant	private	uint256	2
_NOT_ENTERED	constant	private	uint256	1
_initialized	0x00…00	--	uint8	255
_initializing	0x00…00 (1)	--	bool	false
__gap	0x00…01	--	uint256[50]	
__gap	0x00…33	--	uint256[50]	
_roles	0x00…65	--	mapping(bytes32 => struct AccessControlUpgradeable.RoleData)	
__gap	0x00…66	--	uint256[49]	
_paused	0x00…97	--	bool	false
__gap	0x00…98	--	uint256[49]	
chainEngines	0x00…c9	--	mapping(uint256 => enum DlnOrderLib.ChainEngine)	
deBridgeGate	0x00…ca	--	contract IDeBridgeGate	0x0000000000000000000000000000000000000000
_status	0x00…cb	--	uint256	0
__gap	0x00…cc	--	uint256[49]	
globalFixedNativeFee	0x00…fd	--	uint88	0
globalTransferFeeBps	0x00…fd (11)	--	uint16	0
dlnDestinationAddresses	0x00…fe	--	mapping(uint256 => bytes)	
giveOrders	0x00…ff	--	mapping(bytes32 => struct DlnSource.GiveOrderState)	
givePatches	0x01…00	--	mapping(bytes32 => uint256)	
masterNonce	0x01…01	--	mapping(address => uint256)	
collectedFee	0x01…02	--	mapping(address => uint256)	
unexpectedOrderStatusForClaim	0x01…03	--	mapping(bytes32 => address)	
unexpectedOrderStatusForCancel	0x01…04	--	mapping(bytes32 => address)	
unclaimedAffiliateETHFees	0x01…05	--	mapping(address => uint256)	

.

UPGRADE TO 1.3.0 VERSION CHECKS

DlnSource storage layout, updated version

Name	Type	Slot	Offset	Bytes	Contract
_initialized	uint8	0	0	1	contracts/DLN/DlnSource.sol:DlnSource
_initializing	bool	0	1	1	contracts/DLN/DlnSource.sol:DlnSource
__gap	uint256[50]	1	0	1600	contracts/DLN/DlnSource.sol:DlnSource
__gap	uint256[50]	51	0	1600	contracts/DLN/DlnSource.sol:DlnSource
_roles	mapping(bytes32 => struct AccessControlUpgradeable.RoleData)	101	0	32	contracts/DLN/DlnSource.sol:DlnSource
__gap	uint256[49]	102	0	1568	contracts/DLN/DlnSource.sol:DlnSource
_paused	bool	151	0	1	contracts/DLN/DlnSource.sol:DlnSource
__gap	uint256[49]	152	0	1568	contracts/DLN/DlnSource.sol:DlnSource
chainEngines	mapping(uint256 => enum DlnOrderLib.ChainEngine)	201	0	32	contracts/DLN/DlnSource.sol:DlnSource
deBridgeGate	contract IDeBridgeGate	202	0	20	contracts/DLN/DlnSource.sol:DlnSource
_status	uint256	203	0	32	contracts/DLN/DlnSource.sol:DlnSource
__gap	uint256[49]	204	0	1568	contracts/DLN/DlnSource.sol:DlnSource
globalFixedNativeFee	uint88	253	0	11	contracts/DLN/DlnSource.sol:DlnSource
globalTransferFeeBps	uint16	253	11	2	contracts/DLN/DlnSource.sol:DlnSource
dlnDestinationAddresses	mapping(uint256 => bytes)	254	0	32	contracts/DLN/DlnSource.sol:DlnSource
giveOrders	mapping(bytes32 => struct DlnSource.GiveOrderState)	255	0	32	contracts/DLN/DlnSource.sol:DlnSource
givePatches	mapping(bytes32 => uint256)	256	0	32	contracts/DLN/DlnSource.sol:DlnSource
masterNonce	mapping(address => uint256)	257	0	32	contracts/DLN/DlnSource.sol:DlnSource
collectedFee	mapping(address => uint256)	258	0	32	contracts/DLN/DlnSource.sol:DlnSource
unexpectedOrderStatusForClaim	mapping(bytes32 => address)	259	0	32	contracts/DLN/DlnSource.sol:DlnSource
unexpectedOrderStatusForCancel	mapping(bytes32 => address)	260	0	32	contracts/DLN/DlnSource.sol:DlnSource
unclaimedAffiliateETHFees	mapping(address => uint256)	261	0	32	contracts/DLN/DlnSource.sol:DlnSource
unclaimedERC20AffiliateFees	mapping(address => mapping(address => uint256))	262	0	32	contracts/DLN/DlnSource.sol:DlnSource

The following state variable was added to the DlnDestination contract respecting the storage layout:
- externalCallAdapter

DlnDestination storage layout, current version. It can be found here

Label	Slot (Offset)	Visibility	Type	Value/Items
BPS_DENOMINATOR	constant	public	uint256	10000
CANCEL_DESCRIMINATOR	constant	public	bytes8	0x0000000000000000000000000000000000000000
CLAIM_DESCRIMINATOR	constant	public	bytes8	0x0000000000000000000000000000000000000000
DEFAULT_ADMIN_ROLE	constant	public	bytes32	0x0000000000000000000000000000000000000000
EVM_ADDRESS_LENGTH	constant	public	uint256	20
GOVMONITORING_ROLE	constant	public	bytes32	0x2b36fa99e118fa8485d488becf749a974743fbeb
MAX_ADDRESS_LENGTH	constant	public	uint256	255
NATIVE_AMOUNT_DIVIDER_FOR_TRANSFER_TO_SOLANA	constant	public	uint256	10000000000
PROXY_WITH_SENDER	constant	public	uint256	2
REVERT_IF_EXTERNAL_FAIL	constant	public	uint256	1
SOLANA_ADDRESS_LENGTH	constant	public	uint256	32
_ENTERED	constant	private	uint256	2
_NOT_ENTERED	constant	private	uint256	1
_initialized	0x00_00	--	uint8	255
_initializing	0x00_00 (1)	--	bool	false
__gap	0x00_01	--	uint256[50]	
__gap	0x00_33	--	uint256[50]	
_roles	0x00_65	--	mapping(bytes32 => struct AccessControlUpgradeable.RoleData)	
__gap	0x00_66	--	uint256[49]	
_paused	0x00_97	--	bool	false
__gap	0x00_98	--	uint256[49]	
chainEngines	0x00_c9	--	mapping(uint256 => enum DlnBase.ChainEngine)	
deBridgeGate	0x00_ca	--	contract IDeBridgeGate	0x0000000000000000000000000000000000000000
_status	0x00_cb	--	uint256	0
__gap	0x00_cc	--	uint256[49]	
dlnSourceAddresses	0x00_fd	--	mapping(uint256 => bytes)	
takeOrders	0x00_fe	--	mapping(bytes32 => struct DlnDestination.OrderTakeState)	
takePatches	0x00_ff	--	mapping(bytes32 => uint256)	
maxOrderCountPerBatchEvmUnlock	0x01_00	--	uint256	0
maxOrderCountPerBatchSolanaUnlock	0x01_01	--	uint256	0

DlnDestination storage layout, updated version

Name	Type	Slot	Offset	Bytes	Contract
_initialized	uint8	0	0	1	contracts/DLN/DlnDestination.sol
_initializing	bool	0	1	1	contracts/DLN/DlnDestination.sol
__gap	uint256[50]	1	0	1600	contracts/DLN/DlnDestination.sol
__gap	uint256[50]	51	0	1600	contracts/DLN/DlnDestination.sol
_roles	mapping(bytes32 => struct AccessControlUpgradeable.RoleData)	101	0	32	contracts/DLN/DlnDestination.sol
__gap	uint256[49]	102	0	1568	contracts/DLN/DlnDestination.sol
_paused	bool	151	0	1	contracts/DLN/DlnDestination.sol
__gap	uint256[49]	152	0	1568	contracts/DLN/DlnDestination.sol
chainEngines	mapping(uint256 => enum DlnOrderLib.ChainEngine)	201	0	32	contracts/DLN/DlnDestination.sol
deBridgeGate	contract IDeBridgeGate	202	0	20	contracts/DLN/DlnDestination.sol
_status	uint256	203	0	32	contracts/DLN/DlnDestination.sol
__gap	uint256[49]	204	0	1568	contracts/DLN/DlnDestination.sol
dlnSourceAddresses	mapping(uint256 => bytes)	253	0	32	contracts/DLN/DlnDestination.sol
takeOrders	mapping(bytes32 => struct DlnDestination.OrderTakeState)	254	0	32	contracts/DLN/DlnDestination.sol
takePatches	mapping(bytes32 => uint256)	255	0	32	contracts/DLN/DlnDestination.sol
maxOrderCountPerBatchEvmUnlock	uint256	256	0	32	contracts/DLN/DlnDestination.sol
maxOrderCountPerBatchSolanaUnlock	uint256	257	0	32	contracts/DLN/DlnDestination.sol
externalCallAdapter	address	258	0	20	contracts/DLN/DlnDestination.sol

2. Upgrade steps:

- Deploy new contract implementations of the DlnSource and DlnDestination.
- Call upgradeTo with the ProxyAdmin address in the DlnSource and DlnDestination contracts in order to point the proxies to their new implementations:

Listing 16

```
// Upgrade to 1.3.0
vm.startPrank(proxyAdminETH);
TransparentUpgradeableProxy(payable(address(contract_DlnSourceETH)
↳ )).upgradeTo(address(contract_DlnSourceETHImpl));
TransparentUpgradeableProxy(payable(address(
↳ contract_DlnDestinationETH))).upgradeTo(address(
↳ contract_DlnDestinationETHImpl));
vm.stopPrank();
```

- Call DlnDestination.setExternalCallAdapter() to set the External-CallAdapter in the DlnDestination contract.

Full upgrade steps can be found here.

3. All the contracts and parent contracts are correctly initialized.

AUTOMATED TESTING

7.1 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.

Slither results:

DlnDestination.sol

```
INFO:Detectors:
DlnSource._claimUnlock(bytes32,address,uint256,bool) (contracts/DLN/DlnSource.sol#561-625) ignores return value by IERC20Upgradeable(giveTokenAddress).transfer(orderState.affiliateBeneficiary,orderState.affiliateAmount) (contracts/DLN/DlnSource.sol#592-602)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
DlnSource.claimBatchUnlock(bytes32[],address).distinctGiveToken (contracts/DLN/DlnSource.sol#285) is a local variable never initialized
DlnSource.claimBatchUnlock(bytes32[],address).distinctGiveTokenAmount (contracts/DLN/DlnSource.sol#286) is a local variable never initialized
DlnDestination.sendBatchSolanaUnlock(DlnOrderLib.Order[],bytes32,uint256,uint64,uint64).i (contracts/DLN/DlnDestination.sol#384) is a local variable never initialized
DlnDestination.sendBatchSolanaUnlock(DlnOrderLib.Order[],bytes32,uint256,uint64,uint64).solanaSrcProgramId (contracts/DLN/DlnDestination.sol#381) is a local variable never initialized
DlnSource.claimBatchCancel(bytes32[],address).i (contracts/DLN/DlnSource.sol#353) is a local variable never initialized
DlnDestination.sendBatchSolanaUnlock(DlnOrderLib.Order[],bytes32,uint256,uint64,uint64).giveChainId (contracts/DLN/DlnDestination.sol#379) is a local variable never initialized
DlnDestination.sendBatchSolanaUnlock(DlnOrderLib.Order[],bytes32,uint256,uint64,uint64).giveTokenAddress (contracts/DLN/DlnDestination.sol#380) is a local variable never initialized
DlnSource.withdrawFee(address[],address).i (contracts/DLN/DlnSource.sol#459) is a local variable never initialized
DlnDestination._encodeAutoParamsTo(bytes,uint256,bytes).autoParams (contracts/DLN/DlnDestination.sol#757) is a local variable never initialized
DlnSource._createSaltedOrder(DlnOrderLib.OrderCreation,uint64,bytes,uint32,bytes,bytes).affiliateAmount (contracts/DLN/DlnSource.sol#220) is a local variable never initialized
DlnDestination.sendBatchSolanaUnlock(DlnOrderLib.Order[],bytes32,uint256,uint64,uint64).i_scope_0 (contracts/DLN/DlnDestination.sol#442) is a local variable never initialized
DlnDestination.sendBatchEvmUnlock(bytes32[],address,uint256).i (contracts/DLN/DlnDestination.sol#321) is a local variable never initialized
DlnSource.claimBatchUnlock(bytes32[],address).i (contracts/DLN/DlnSource.sol#288) is a local variable never initialized
DlnDestination.sendBatchEvmUnlock(bytes32[],address,uint256).giveChainId (contracts/DLN/DlnDestination.sol#319) is a local variable never initialized
DlnDestination.sendBatchEvmUnlock(bytes32[],address,uint256).i_scope_0 (contracts/DLN/DlnDestination.sol#342) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
Function DlnBase._safeTransferETH(address,uint256) (contracts/DLN/DlnBase.sol#155-158) contains a low level call to a custom address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/call_forward_to_protected.md
INFO:Detectors:
DlnDestination.setExternalCallAdapter(address)._externalCallAdapter (contracts/DLN/DlnDestination.sol#678) lacks a zero-check on :
        - externalCallAdapter = _externalCallAdapter (contracts/DLN/DlnDestination.sol#683)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
DlnBase._safeTransferETH(address,uint256) (contracts/DLN/DlnBase.sol#155-158) has external calls inside a loop: (success) = to.call{value: value}(new bytes(0)) (contracts/DLN/DlnBase.sol#156)
DlnSource._claimUnlock(bytes32,address,uint256,bool) (contracts/DLN/DlnSource.sol#561-625) has external calls inside a loop: (success,None) = orderState.affiliateBeneficiary.call{gas: 2300,value: orderState.affiliateAmount}(new bytes(0))
 (contracts/DLN/DlnSource.sol#585)
DlnSource._claimUnlock(bytes32,address,uint256,bool) (contracts/DLN/DlnSource.sol#561-625) has external calls inside a loop: IERC20Upgradeable(giveTokenAddress).transfer(orderState.affiliateBeneficiary,orderState.affiliateAmount) (contracts/DLN/DlnSource.sol#592-602)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in DlnDestination.sendBatchEvmUnlock(bytes32[],address,uint256) (contracts/DLN/DlnDestination.sol#311-345):
        External calls:
        - submissionId = _sendCrossChainMessage(giveChainId,abi.encodePacked(_beneficiary),_executionFee,claimUnlockMethod) (contracts/DLN/DlnDestination.sol#335-340)
                - deBridgeGate.send{value: msg.value}(address(0),msg.value,_chainIdTo,srcAddress,,false,0,autoParams) (contracts/DLN/DlnDestination.sol#817-826)
        Event emitted after the call(s):
        - SentOrderUnlock(_orderIds[i_scope_0],abi.encodePacked(_beneficiary),submissionId) (contracts/DLN/DlnDestination.sol#343)
Reentrancy in DlnDestination.sendBatchSolanaUnlock(DlnOrderLib.Order[],bytes32,uint256,uint64,uint64) (contracts/DLN/DlnDestination.sol#367-445):
        External calls:
        - submissionId = _sendCrossChainMessage(giveChainId,abi.encodePacked(_beneficiary),_executionFee,instructionsData) (contracts/DLN/DlnDestination.sol#434-439)
                - deBridgeGate.send{value: msg.value}(address(0),msg.value,_chainIdTo,srcAddress,,false,0,autoParams) (contracts/DLN/DlnDestination.sol#817-826)
        Event emitted after the call(s):
        - SentOrderUnlock{orderIds[i_scope_0],abi.encodePacked(_beneficiary),submissionId) (contracts/DLN/DlnDestination.sol#443)
Reentrancy in DlnDestination.sendEvmOrderCancel(DlnOrderLib.Order,address,uint256) (contracts/DLN/DlnDestination.sol#524-555):
        External calls:
        - submissionId = _sendCrossChainMessage(_order.giveChainId,abi.encodePacked(_cancelBeneficiary),_executionFee,claimCancelMethod) (contracts/DLN/DlnDestination.sol#548-553)
                - deBridgeGate.send{value: msg.value}(address(0),msg.value,_chainIdTo,srcAddress,,false,0,autoParams) (contracts/DLN/DlnDestination.sol#817-826)
        Event emitted after the call(s):
        - SentOrderCancel(_order,orderId,abi.encodePacked(_cancelBeneficiary),submissionId) (contracts/DLN/DlnDestination.sol#554)
Reentrancy in DlnDestination.sendEvmUnlock(bytes32,address,uint256) (contracts/DLN/DlnDestination.sol#272-289):
        External calls:
        - submissionId = _sendCrossChainMessage(giveChainId,abi.encodePacked(_beneficiary),_executionFee,claimUnlockMethod) (contracts/DLN/DlnDestination.sol#281-286)
                - deBridgeGate.send{value: msg.value}(address(0),msg.value,_chainIdTo,srcAddress,,false,0,autoParams) (contracts/DLN/DlnDestination.sol#817-826)
        Event emitted after the call(s):
        - SentOrderUnlock(_orderId,abi.encodePacked(_beneficiary),submissionId) (contracts/DLN/DlnDestination.sol#288)
Reentrancy in DlnDestination.sendSolanaOrderCancel(DlnOrderLib.Order,bytes32,uint256,uint64,uint64) (contracts/DLN/DlnDestination.sol#578-632):
        External calls:
        - submissionId = _sendCrossChainMessage(_order.giveChainId,abi.encodePacked(_cancelBeneficiary),_executionFee,claimCancelMethod) (contracts/DLN/DlnDestination.sol#624-629)
                - deBridgeGate.send{value: msg.value}(address(0),msg.value,_chainIdTo,srcAddress,,false,0,autoParams) (contracts/DLN/DlnDestination.sol#817-826)
        Event emitted after the call(s):
        - SentOrderCancel(_order,orderId,abi.encodePacked(_cancelBeneficiary),submissionId) (contracts/DLN/DlnDestination.sol#631)
Reentrancy in DlnDestination.sendSolanaUnlock(DlnOrderLib.Order,bytes32,uint256,uint64,uint64) (contracts/DLN/DlnDestination.sol#466-503):
        External calls:
        - submissionId = _sendCrossChainMessage(giveChainId,abi.encodePacked(_beneficiary),_executionFee,instructionsData) (contracts/DLN/DlnDestination.sol#495-500)
                - deBridgeGate.send{value: msg.value}(address(0),msg.value,_chainIdTo,srcAddress,,false,0,autoParams) (contracts/DLN/DlnDestination.sol#817-826)
        Event emitted after the call(s):
        - SentOrderUnlock(orderId,abi.encodePacked(_beneficiary),submissionId) (contracts/DLN/DlnDestination.sol#502)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Function DlnBase._executePermit(address,bytes) (contracts/DLN/DlnBase.sol#114-131) has a dubious typecast: address=IERC20Permit
Function DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes) (contracts/DLN/DlnSource.sol#174-188) has a dubious typecast: bytes<=uint64
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO:Detectors:
DlnBase.getChainId() (contracts/DLN/DlnBase.sol#180-184) uses assembly
        - INLINE ASM (contracts/DLN/DlnBase.sol#181-183)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
DlnDestination._fulfillOrder(bytes,DlnOrderLib.Order,uint256,bytes32,address,address) (contracts/DLN/DlnDestination.sol#180-251) has a high cyclomatic complexity (12).
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cyclomatic-complexity
```

```
INFO:Detectors:
Pragma version^0.8.17 (contracts/DLN/DlnBase.sol#2) allows old versions
Pragma version^0.8.17 (contracts/DLN/DlnDestination.sol#2) allows old versions
Pragma version^0.8.17 (contracts/DLN/DlnSource.sol#2) allows old versions
Pragma version^0.8.7 (contracts/interfaces/IDlnDestination.sol#3) allows old versions
Pragma version^0.8.7 (contracts/interfaces/IDlnSource.sol#2) allows old versions
Pragma version^0.8.7 (contracts/interfaces/IERC20Permit.sol#3) allows old versions
Pragma version^0.8.0 (contracts/interfaces/IExternalCallAdapter.sol#3) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in DlnBase._safeTransferETH(address,uint256) (contracts/DLN/DlnBase.sol#155-158):
        - (success) = to.call{value: value}(new bytes(0)) (contracts/DLN/DlnBase.sol#156)
Low level call in DlnSource._claimUnlock(bytes32,address,uint256,bool) (contracts/DLN/DlnSource.sol#561-625):
        - (success,None) = orderState.affiliateBeneficiary.call{gas: 2300,value: orderState.affiliateAmount}(new bytes(0)) (contracts/DLN/DlnSource.sol#585)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function DlnBase.__DlnBase_init(IDeBridgeGate) (contracts/DLN/DlnBase.sol#84-90) is not in mixedCase
Parameter DlnBase.__DlnBase_init(IDeBridgeGate)._deBridgeGate (contracts/DLN/DlnBase.sol#84) is not in mixedCase
Function DlnBase.__DlnBase_init_unchained(IDeBridgeGate) (contracts/DLN/DlnBase.sol#92-98) is not in mixedCase
Parameter DlnBase.__DlnBase_init_unchained(IDeBridgeGate)._deBridgeGate (contracts/DLN/DlnBase.sol#92) is not in mixedCase
Parameter DlnBase.getOrderId(DlnOrderLib.Order)._order (contracts/DLN/DlnBase.sol#175) is not in mixedCase
Parameter DlnDestination.initialize(IDeBridgeGate)._deBridgeGate (contracts/DLN/DlnDestination.sol#132) is not in mixedCase
Parameter DlnDestination.fulfillOrder(DlnOrderLib.Order,uint256,bytes32,bytes,address)._order (contracts/DLN/DlnDestination.sol#143) is not in mixedCase
Parameter DlnDestination.fulfillOrder(DlnOrderLib.Order,uint256,bytes32,bytes,address)._fulfillAmount (contracts/DLN/DlnDestination.sol#144) is not in mixedCase
Parameter DlnDestination.fulfillOrder(DlnOrderLib.Order,uint256,bytes32,bytes,address)._orderId (contracts/DLN/DlnDestination.sol#145) is not in mixedCase
Parameter DlnDestination.fulfillOrder(DlnOrderLib.Order,uint256,bytes32,bytes,address)._permitEnvelope (contracts/DLN/DlnDestination.sol#146) is not in mixedCase
Parameter DlnDestination.fulfillOrder(DlnOrderLib.Order,uint256,bytes32,bytes,address)._unlockAuthority (contracts/DLN/DlnDestination.sol#147) is not in mixedCase
Parameter DlnDestination.fulfillOrder(DlnOrderLib.Order,uint256,bytes32,bytes,address,address)._order (contracts/DLN/DlnDestination.sol#163) is not in mixedCase
Parameter DlnDestination.fulfillOrder(DlnOrderLib.Order,uint256,bytes32,bytes,address,address)._fulFillAmount (contracts/DLN/DlnDestination.sol#164) is not in mixedCase
Parameter DlnDestination.fulfillOrder(DlnOrderLib.Order,uint256,bytes32,bytes,address,address)._orderId (contracts/DLN/DlnDestination.sol#165) is not in mixedCase
Parameter DlnDestination.fulfillOrder(DlnOrderLib.Order,uint256,bytes32,bytes,address,address)._permitEnvelope (contracts/DLN/DlnDestination.sol#166) is not in mixedCase
Parameter DlnDestination.fulfillOrder(DlnOrderLib.Order,uint256,bytes32,bytes,address,address)._unlockAuthority (contracts/DLN/DlnDestination.sol#167) is not in mixedCase
Parameter DlnDestination.fulfillOrder(DlnOrderLib.Order,uint256,bytes32,bytes,address,address)._externalCallRewardBeneficiary (contracts/DLN/DlnDestination.sol#168) is not in mixedCase
Parameter DlnDestination.sendEvmUnlock(bytes32,address,uint256)._orderId (contracts/DLN/DlnDestination.sol#273) is not in mixedCase
Parameter DlnDestination.sendEvmUnlock(bytes32,address,uint256)._beneficiary (contracts/DLN/DlnDestination.sol#274) is not in mixedCase
Parameter DlnDestination.sendEvmUnlock(bytes32,address,uint256)._executionFee (contracts/DLN/DlnDestination.sol#275) is not in mixedCase
Parameter DlnDestination.sendBatchEvmUnlock(bytes32[],address,uint256)._orderIds (contracts/DLN/DlnDestination.sol#312) is not in mixedCase
Parameter DlnDestination.sendBatchEvmUnlock(bytes32[],address,uint256)._beneficiary (contracts/DLN/DlnDestination.sol#313) is not in mixedCase
Parameter DlnDestination.sendBatchEvmUnlock(bytes32[],address,uint256)._executionFee (contracts/DLN/DlnDestination.sol#314) is not in mixedCase
Parameter DlnDestination.sendBatchSolanaUnlock(DlnOrderLib.Order[],bytes32,uint256,uint64,uint64)._orders (contracts/DLN/DlnDestination.sol#368) is not in mixedCase
Parameter DlnDestination.sendBatchSolanaUnlock(DlnOrderLib.Order[],bytes32,uint256,uint64,uint64)._beneficiary (contracts/DLN/DlnDestination.sol#369) is not in mixedCase
Parameter DlnDestination.sendBatchSolanaUnlock(DlnOrderLib.Order[],bytes32,uint256,uint64,uint64)._executionFee (contracts/DLN/DlnDestination.sol#370) is not in mixedCase
Parameter DlnDestination.sendBatchSolanaUnlock(DlnOrderLib.Order[],bytes32,uint256,uint64,uint64)._initWalletIfNeededInstructionReward (contracts/DLN/DlnDestination.sol#371) is not in mixedCase
Parameter DlnDestination.sendBatchSolanaUnlock(DlnOrderLib.Order[],bytes32,uint256,uint64,uint64)._claimUnlockInstructionReward (contracts/DLN/DlnDestination.sol#372) is not in mixedCase
Parameter DlnDestination.sendSolanaUnlock(DlnOrderLib.Order,bytes32,uint256,uint64,uint64)._order (contracts/DLN/DlnDestination.sol#467) is not in mixedCase
Parameter DlnDestination.sendSolanaUnlock(DlnOrderLib.Order,bytes32,uint256,uint64,uint64)._beneficiary (contracts/DLN/DlnDestination.sol#468) is not in mixedCase
Parameter DlnDestination.sendSolanaUnlock(DlnOrderLib.Order,bytes32,uint256,uint64,uint64)._executionFee (contracts/DLN/DlnDestination.sol#469) is not in mixedCase
Parameter DlnDestination.sendSolanaUnlock(DlnOrderLib.Order,bytes32,uint256,uint64,uint64)._initWalletIfNeededInstructionReward (contracts/DLN/DlnDestination.sol#470) is not in mixedCase
Parameter DlnDestination.sendSolanaUnlock(DlnOrderLib.Order,bytes32,uint256,uint64,uint64)._claimUnlockInstructionReward (contracts/DLN/DlnDestination.sol#471) is not in mixedCase
Parameter DlnDestination.sendEvmOrderCancel(DlnOrderLib.Order,address,uint256)._order (contracts/DLN/DlnDestination.sol#525) is not in mixedCase
Parameter DlnDestination.sendEvmOrderCancel(DlnOrderLib.Order,address,uint256)._cancelBeneficiary (contracts/DLN/DlnDestination.sol#526) is not in mixedCase
Parameter DlnDestination.sendEvmOrderCancel(DlnOrderLib.Order,address,uint256)._executionFee (contracts/DLN/DlnDestination.sol#527) is not in mixedCase
Parameter DlnDestination.sendSolanaOrderCancel(DlnOrderLib.Order,bytes32,uint256,uint64,uint64)._order (contracts/DLN/DlnDestination.sol#579) is not in mixedCase
Parameter DlnDestination.sendSolanaOrderCancel(DlnOrderLib.Order,bytes32,uint256,uint64,uint64)._cancelBeneficiary (contracts/DLN/DlnDestination.sol#580) is not in mixedCase
Parameter DlnDestination.sendSolanaOrderCancel(DlnOrderLib.Order,bytes32,uint256,uint64,uint64)._executionFee (contracts/DLN/DlnDestination.sol#581) is not in mixedCase
Parameter DlnDestination.sendSolanaOrderCancel(DlnOrderLib.Order,bytes32,uint256,uint64,uint64)._initWalletIfNeededInstructionReward (contracts/DLN/DlnDestination.sol#582) is not in mixedCase
Parameter DlnDestination.sendSolanaOrderCancel(DlnOrderLib.Order,bytes32,uint256,uint64,uint64)._claimCancelInstructionReward (contracts/DLN/DlnDestination.sol#583) is not in mixedCase
Parameter DlnDestination.patchOrderTake(DlnOrderLib.Order,uint256)._order (contracts/DLN/DlnDestination.sol#648) is not in mixedCase
Parameter DlnDestination.patchOrderTake(DlnOrderLib.Order,uint256)._newSubtrahend (contracts/DLN/DlnDestination.sol#648) is not in mixedCase
Parameter DlnDestination.setDlnSourceAddress(uint256,bytes,DlnOrderLib.ChainEngine)._chainIdFrom (contracts/DLN/DlnDestination.sol#668) is not in mixedCase
Parameter DlnDestination.setDlnSourceAddress(uint256,bytes,DlnOrderLib.ChainEngine)._dlnSourceAddress (contracts/DLN/DlnDestination.sol#668) is not in mixedCase
Parameter DlnDestination.setDlnSourceAddress(uint256,bytes,DlnOrderLib.ChainEngine)._chainEngine (contracts/DLN/DlnDestination.sol#668) is not in mixedCase
Parameter DlnDestination.setExternalCallAdapter(address)._externalCallAdapter (contracts/DLN/DlnDestination.sol#678) is not in mixedCase
Parameter DlnDestination.setMaxOrderCountsPerBatch(uint256,uint256)._newEvmCount (contracts/DLN/DlnDestination.sol#687) is not in mixedCase
Parameter DlnDestination.setMaxOrderCountsPerBatch(uint256,uint256)._newSolanaCount (contracts/DLN/DlnDestination.sol#687) is not in mixedCase
Parameter DlnSource.initialize(IDeBridgeGate,uint88,uint16)._deBridgeGate (contracts/DLN/DlnSource.sol#158) is not in mixedCase
Parameter DlnSource.initialize(IDeBridgeGate,uint88,uint16)._globalFixedNativeFee (contracts/DLN/DlnSource.sol#159) is not in mixedCase
Parameter DlnSource.initialize(IDeBridgeGate,uint88,uint16)._globalTransferFeeBps (contracts/DLN/DlnSource.sol#160) is not in mixedCase
Parameter DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes)._orderCreation (contracts/DLN/DlnSource.sol#175) is not in mixedCase
Parameter DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes)._affiliateFee (contracts/DLN/DlnSource.sol#176) is not in mixedCase
Parameter DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes)._referralCode (contracts/DLN/DlnSource.sol#177) is not in mixedCase
Parameter DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes)._permitEnvelope (contracts/DLN/DlnSource.sol#178) is not in mixedCase
Parameter DlnSource.createSaltedOrder(DlnOrderLib.OrderCreation,uint64,bytes,uint32,bytes,bytes)._orderCreation (contracts/DLN/DlnSource.sol#194) is not in mixedCase
Parameter DlnSource.createSaltedOrder(DlnOrderLib.OrderCreation,uint64,bytes,uint32,bytes,bytes)._salt (contracts/DLN/DlnSource.sol#195) is not in mixedCase
Parameter DlnSource.createSaltedOrder(DlnOrderLib.OrderCreation,uint64,bytes,uint32,bytes,bytes)._affiliateFee (contracts/DLN/DlnSource.sol#196) is not in mixedCase
Parameter DlnSource.createSaltedOrder(DlnOrderLib.OrderCreation,uint64,bytes,uint32,bytes,bytes)._referralCode (contracts/DLN/DlnSource.sol#197) is not in mixedCase
Parameter DlnSource.createSaltedOrder(DlnOrderLib.OrderCreation,uint64,bytes,uint32,bytes,bytes)._permitEnvelope (contracts/DLN/DlnSource.sol#198) is not in mixedCase
Parameter DlnSource.createSaltedOrder(DlnOrderLib.OrderCreation,uint64,bytes,uint32,bytes,bytes)._metadata (contracts/DLN/DlnSource.sol#199) is not in mixedCase
Parameter DlnSource.claimBatchUnlock(bytes32[],address)._orderIds (contracts/DLN/DlnSource.sol#277) is not in mixedCase
Parameter DlnSource.claimBatchUnlock(bytes32[],address)._beneficiary (contracts/DLN/DlnSource.sol#277) is not in mixedCase
Parameter DlnSource.claimUnlock(bytes32,address)._orderId (contracts/DLN/DlnSource.sol#322) is not in mixedCase
Parameter DlnSource.claimUnlock(bytes32,address)._beneficiary (contracts/DLN/DlnSource.sol#322) is not in mixedCase
Parameter DlnSource.claimBatchCancel(bytes32[],address)._orderIds (contracts/DLN/DlnSource.sol#346) is not in mixedCase
Parameter DlnSource.claimBatchCancel(bytes32[],address)._beneficiary (contracts/DLN/DlnSource.sol#346) is not in mixedCase
Parameter DlnSource.claimCancel(bytes32,address)._orderId (contracts/DLN/DlnSource.sol#371) is not in mixedCase
Parameter DlnSource.claimCancel(bytes32,address)._beneficiary (contracts/DLN/DlnSource.sol#371) is not in mixedCase
Parameter DlnSource.patchOrderGive(DlnOrderLib.Order,uint256,bytes)._order (contracts/DLN/DlnSource.sol#398) is not in mixedCase
Parameter DlnSource.patchOrderGive(DlnOrderLib.Order,uint256,bytes)._addGiveAmount (contracts/DLN/DlnSource.sol#399) is not in mixedCase
Parameter DlnSource.patchOrderGive(DlnOrderLib.Order,uint256,bytes)._permitEnvelope (contracts/DLN/DlnSource.sol#400) is not in mixedCase
Parameter DlnSource.setDlnDestinationAddress(uint256,bytes,DlnOrderLib.ChainEngine)._chainIdTo (contracts/DLN/DlnSource.sol#439) is not in mixedCase
Parameter DlnSource.setDlnDestinationAddress(uint256,bytes,DlnOrderLib.ChainEngine)._dlnDestinationAddress (contracts/DLN/DlnSource.sol#440) is not in mixedCase
Parameter DlnSource.setDlnDestinationAddress(uint256,bytes,DlnOrderLib.ChainEngine)._chainEngine (contracts/DLN/DlnSource.sol#441) is not in mixedCase
Parameter DlnSource.withdrawFee(address[],address)._tokens (contracts/DLN/DlnSource.sol#454) is not in mixedCase
Parameter DlnSource.withdrawFee(address[],address)._beneficiary (contracts/DLN/DlnSource.sol#454) is not in mixedCase
Parameter DlnSource.updateGlobalFee(uint88,uint16)._globalFixedNativeFee (contracts/DLN/DlnSource.sol#474) is not in mixedCase
Parameter DlnSource.updateGlobalFee(uint88,uint16)._globalTransferFeeBps (contracts/DLN/DlnSource.sol#475) is not in mixedCase
Parameter DlnSource.validateCreationOrder(DlnOrderLib.OrderCreation,address)._orderCreation (contracts/DLN/DlnSource.sol#489) is not in mixedCase
Parameter DlnSource.validateCreationOrder(DlnOrderLib.OrderCreation,address)._signer (contracts/DLN/DlnSource.sol#489) is not in mixedCase
Parameter DlnSource.validateCreationOrder(DlnOrderLib.OrderCreation,address,uint64)._orderCreation (contracts/DLN/DlnSource.sol#497) is not in mixedCase
Parameter DlnSource.validateCreationOrder(DlnOrderLib.OrderCreation,address,uint64)._signer (contracts/DLN/DlnSource.sol#497) is not in mixedCase
Parameter DlnSource.validateCreationOrder(DlnOrderLib.OrderCreation,address,uint64)._salt (contracts/DLN/DlnSource.sol#497) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable DlnDestination.setDlnSourceAddress(uint256,bytes,DlnOrderLib.ChainEngine)._chainEngine (contracts/DLN/DlnDestination.sol#668) is too similar to DlnBase.chainEngines (contracts/DLN/DlnBase.sol#41)
Variable DlnDestination._validateSolanaRewards(uint256,uint256,uint64,uint64)._solanaExternalCallReward1 (contracts/DLN/DlnDestination.sol#780) is too similar to DlnDestination._validateSolanaRewards(uint256,uint256,uint64,uint64)._solanaExternalCallReward2 (contracts/DLN/DlnDestination.sol#781)
Variable DlnDestination._prepareOrderStateForUnlock(bytes32,DlnOrderLib.ChainEngine)._chainEngine (contracts/DLN/DlnDestination.sol#711) is too similar to DlnBase.chainEngines (contracts/DLN/DlnBase.sol#41)
Variable DlnSource.setDlnDestinationAddress(uint256,bytes,DlnOrderLib.ChainEngine)._chainEngine (contracts/DLN/DlnSource.sol#441) is too similar to DlnBase.chainEngines (contracts/DLN/DlnBase.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
Function DlnBase._executePermit(address,bytes) (contracts/DLN/DlnBase.sol#114-131) contains magic numbers: 32, 64

Function DlnSource._createSaltedOrder(DlnOrderLib.OrderCreation,uint64,bytes,uint32,bytes,bytes) (contracts/DLN/DlnSource.sol#211-266) contains magic numbers: 52, 20
Function DlnSource._claimUnlock(bytes32,address,uint256,bool) (contracts/DLN/DlnSource.sol#561-625) contains magic number: 2300
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
In a function DlnSource._pullTokens(DlnOrderLib.OrderCreation,DlnOrderLib.Order,bytes) (contracts/DLN/DlnSource.sol#534-551) variable DlnSource.globalFixedNativeFee (contracts/DLN/DlnSource.sol#18) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

DlnSource.sol

```
INFO:Detectors:
DlnSource._claimUnlock(bytes32,address,uint256,bool) (contracts/DLN/DlnSource.sol#561-625) ignores return value by IERC20Upgradeable(giveTokenAddress).transfer(orderState.affiliateBeneficiary,orderState.affiliateAmount) (contracts/DLN/DlnSource.sol#592-602)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
DlnSource._createSaltedOrder(DlnOrderLib.OrderCreation,uint64,bytes,uint32,bytes,bytes).affiliateAmount (contracts/DLN/DlnSource.sol#220) is a local variable never initialized
DlnSource.claimBatchUnlock(bytes32[],address).i (contracts/DLN/DlnSource.sol#288) is a local variable never initialized
DlnSource.claimBatchCancel(bytes32[],address).i (contracts/DLN/DlnSource.sol#353) is a local variable never initialized
DlnSource.claimBatchUnlock(bytes32[],address).distinctGiveToken (contracts/DLN/DlnSource.sol#285) is a local variable never initialized
DlnSource.claimBatchUnlock(bytes32[],address).distinctGiveTokenAmount (contracts/DLN/DlnSource.sol#286) is a local variable never initialized
DlnSource.withdrawFee(address[],address).i (contracts/DLN/DlnSource.sol#459) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
Function DlnBase._safeTransferETH(address,uint256) (contracts/DLN/DlnBase.sol#155-158) contains a low level call to a custom address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/call_forward_to_protected.md
INFO:Detectors:
DlnBase._safeTransferETH(address,uint256) (contracts/DLN/DlnBase.sol#155-158) has external calls inside a loop: (success) = to.call{value: value}(new bytes(0)) (contracts/DLN/DlnBase.sol#156)
DlnSource._claimUnlock(bytes32,address,uint256,bool) (contracts/DLN/DlnSource.sol#561-625) has external calls inside a loop: (success,None) = orderState.affiliateBeneficiary.call{gas: 2300,value: orderState.affiliateAmount}(new bytes(0)) (contracts/DLN/DlnSource.sol#585)
DlnSource._claimUnlock(bytes32,address,uint256,bool) (contracts/DLN/DlnSource.sol#561-625) has external calls inside a loop: IERC20Upgradeable(giveTokenAddress).transfer(orderState.affiliateBeneficiary,orderState.affiliateAmount) (contracts/DLN/DlnSource.sol#592-602)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#calls-inside-a-loop
INFO:Detectors:
Function DlnBase._executePermit(address,bytes) (contracts/DLN/DlnBase.sol#114-131) has a dubious typecast: address<=IERC20Permit
Function DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes) (contracts/DLN/DlnSource.sol#174-188) has a dubious typecast: bytes<=uint64
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO:Detectors:
DlnBase.getChainId() (contracts/DLN/DlnBase.sol#180-184) uses assembly
        - INLINE ASM (contracts/DLN/DlnBase.sol#181-183)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
INFO:Detectors:
Pragma version^0.8.17 (contracts/DLN/DlnBase.sol#2) allows old versions
Pragma version^0.8.17 (contracts/DLN/DlnSource.sol#2) allows old versions
Pragma version^0.8.7 (contracts/interfaces/IDlnSource.sol#2) allows old versions
Pragma version^0.8.7 (contracts/interfaces/IERC20Permit.sol#3) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in DlnBase._safeTransferETH(address,uint256) (contracts/DLN/DlnBase.sol#155-158):
        - (success) = to.call{value: value}(new bytes(0)) (contracts/DLN/DlnBase.sol#156)
Low level call in DlnSource._claimUnlock(bytes32,address,uint256,bool) (contracts/DLN/DlnSource.sol#561-625):
        - (success,None) = orderState.affiliateBeneficiary.call{gas: 2300,value: orderState.affiliateAmount}(new bytes(0)) (contracts/DLN/DlnSource.sol#585)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function DlnBase.__DlnBase_init(IDeBridgeGate) (contracts/DLN/DlnBase.sol#84-90) is not in mixedCase
Parameter DlnBase.__DlnBase_init(IDeBridgeGate)._deBridgeGate (contracts/DLN/DlnBase.sol#84) is not in mixedCase
Function DlnBase.__DlnBase_init_unchained(IDeBridgeGate) (contracts/DLN/DlnBase.sol#92-98) is not in mixedCase
Parameter DlnBase.__DlnBase_init_unchained(IDeBridgeGate)._deBridgeGate (contracts/DLN/DlnBase.sol#92) is not in mixedCase
Parameter DlnBase.getOrderId(DlnOrderLib.Order)._order (contracts/DLN/DlnBase.sol#175) is not in mixedCase
Parameter DlnSource.initialize(IDeBridgeGate,uint88,uint16)._deBridgeGate (contracts/DLN/DlnSource.sol#158) is not in mixedCase
Parameter DlnSource.initialize(IDeBridgeGate,uint88,uint16)._globalFixedNativeFee (contracts/DLN/DlnSource.sol#159) is not in mixedCase
Parameter DlnSource.initialize(IDeBridgeGate,uint88,uint16)._globalTransferFeeBps (contracts/DLN/DlnSource.sol#160) is not in mixedCase
Parameter DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes)._orderCreation (contracts/DLN/DlnSource.sol#175) is not in mixedCase
Parameter DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes)._affiliateFee (contracts/DLN/DlnSource.sol#176) is not in mixedCase
Parameter DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes)._referralCode (contracts/DLN/DlnSource.sol#177) is not in mixedCase
Parameter DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes)._permitEnvelope (contracts/DLN/DlnSource.sol#178) is not in mixedCase
Parameter DlnSource.createSaltedOrder(DlnOrderLib.OrderCreation,uint64,bytes,uint32,bytes,bytes)._orderCreation (contracts/DLN/DlnSource.sol#194) is not in mixedCase
Parameter DlnSource.createSaltedOrder(DlnOrderLib.OrderCreation,uint64,bytes,uint32,bytes,bytes)._salt (contracts/DLN/DlnSource.sol#195) is not in mixedCase
Parameter DlnSource.createSaltedOrder(DlnOrderLib.OrderCreation,uint64,bytes,uint32,bytes,bytes)._affiliateFee (contracts/DLN/DlnSource.sol#196) is not in mixedCase
Parameter DlnSource.createSaltedOrder(DlnOrderLib.OrderCreation,uint64,bytes,uint32,bytes,bytes)._referralCode (contracts/DLN/DlnSource.sol#197) is not in mixedCase
Parameter DlnSource.createSaltedOrder(DlnOrderLib.OrderCreation,uint64,bytes,uint32,bytes,bytes)._permitEnvelope (contracts/DLN/DlnSource.sol#198) is not in mixedCase
Parameter DlnSource.createSaltedOrder(DlnOrderLib.OrderCreation,uint64,bytes,uint32,bytes,bytes)._metadata (contracts/DLN/DlnSource.sol#199) is not in mixedCase
Parameter DlnSource.claimBatchUnlock(bytes32[],address)._orderIds (contracts/DLN/DlnSource.sol#277) is not in mixedCase
Parameter DlnSource.claimBatchUnlock(bytes32[],address)._beneficiary (contracts/DLN/DlnSource.sol#277) is not in mixedCase
Parameter DlnSource.claimUnlock(bytes32,address)._orderId (contracts/DLN/DlnSource.sol#322) is not in mixedCase
Parameter DlnSource.claimUnlock(bytes32,address)._beneficiary (contracts/DLN/DlnSource.sol#322) is not in mixedCase
Parameter DlnSource.claimBatchCancel(bytes32[],address)._orderIds (contracts/DLN/DlnSource.sol#346) is not in mixedCase
Parameter DlnSource.claimBatchCancel(bytes32[],address)._beneficiary (contracts/DLN/DlnSource.sol#346) is not in mixedCase
Parameter DlnSource.claimCancel(bytes32,address)._orderId (contracts/DLN/DlnSource.sol#371) is not in mixedCase
Parameter DlnSource.claimCancel(bytes32,address)._beneficiary (contracts/DLN/DlnSource.sol#371) is not in mixedCase
Parameter DlnSource.patchOrderGive(DlnOrderLib.Order,uint256,bytes)._order (contracts/DLN/DlnSource.sol#398) is not in mixedCase
Parameter DlnSource.patchOrderGive(DlnOrderLib.Order,uint256,bytes)._addGiveAmount (contracts/DLN/DlnSource.sol#399) is not in mixedCase
Parameter DlnSource.patchOrderGive(DlnOrderLib.Order,uint256,bytes)._permitEnvelope (contracts/DLN/DlnSource.sol#400) is not in mixedCase
Parameter DlnSource.setDlnDestinationAddress(uint256,bytes,DlnOrderLib.ChainEngine)._chainIdTo (contracts/DLN/DlnSource.sol#439) is not in mixedCase
Parameter DlnSource.setDlnDestinationAddress(uint256,bytes,DlnOrderLib.ChainEngine)._dlnDestinationAddress (contracts/DLN/DlnSource.sol#440) is not in mixedCase
Parameter DlnSource.setDlnDestinationAddress(uint256,bytes,DlnOrderLib.ChainEngine)._chainEngine (contracts/DLN/DlnSource.sol#441) is not in mixedCase
Parameter DlnSource.withdrawFee(address[],address)._tokens (contracts/DLN/DlnSource.sol#454) is not in mixedCase
Parameter DlnSource.withdrawFee(address[],address)._beneficiary (contracts/DLN/DlnSource.sol#454) is not in mixedCase
Parameter DlnSource.updateGlobalFee(uint88,uint16)._globalFixedNativeFee (contracts/DLN/DlnSource.sol#474) is not in mixedCase
Parameter DlnSource.updateGlobalFee(uint88,uint16)._globalTransferFeeBps (contracts/DLN/DlnSource.sol#475) is not in mixedCase
Parameter DlnSource.validateCreationOrder(DlnOrderLib.OrderCreation,address)._orderCreation (contracts/DLN/DlnSource.sol#489) is not in mixedCase
Parameter DlnSource.validateCreationOrder(DlnOrderLib.OrderCreation,address)._signer (contracts/DLN/DlnSource.sol#489) is not in mixedCase
Parameter DlnSource.validateCreationOrder(DlnOrderLib.OrderCreation,address,uint64)._orderCreation (contracts/DLN/DlnSource.sol#497) is not in mixedCase
Parameter DlnSource.validateCreationOrder(DlnOrderLib.OrderCreation,address,uint64)._signer (contracts/DLN/DlnSource.sol#497) is not in mixedCase
Parameter DlnSource.validateCreationOrder(DlnOrderLib.OrderCreation,address,uint64)._salt (contracts/DLN/DlnSource.sol#497) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

INFO:Detectors:
Variable DlnSource.setDlnDestinationAddress(uint256,bytes,DlnOrderLib.ChainEngine)._chainEngine (contracts/DLN/DlnSource.sol#441) is too similar to DlnBase.chainEngines (contracts/DLN/DlnBase.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
Function DlnBase._executePermit(address,bytes) (contracts/DLN/DlnBase.sol#114-131) contains magic numbers: 32, 64
Function DlnSource._createSaltedOrder(DlnOrderLib.OrderCreation,uint64,bytes,uint32,bytes,bytes) (contracts/DLN/DlnSource.sol#211-266) contains magic numbers: 52, 20
Function DlnSource._claimUnlock(bytes32,address,uint256,bool) (contracts/DLN/DlnSource.sol#561-625) contains magic number: 2300
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
In a function DlnSource._pullTokens(DlnOrderLib.OrderCreation,DlnOrderLib.Order,bytes) (contracts/DLN/DlnSource.sol#534-551) variable DlnSource.globalFixedNativeFee (contracts/DLN/DlnSource.sol#18) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

DlnExternalCallAdapter.sol

```
INFO:Detectors:
DlnExternalCallAdapter._safeTransferETH(address,uint256) (contracts/adapters/DlnExternalCallAdapter.sol#417-420) sends eth to arbitrary user
        Dangerous calls:
        - (success) = to.call{value: value}(new bytes(0)) (contracts/adapters/DlnExternalCallAdapter.sol#418)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Function DlnExternalCallAdapter.constructor() (contracts/adapters/DlnExternalCallAdapter.sol#92-94) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/strange_setter.md
INFO:Detectors:
Function DlnExternalCallAdapter.onlyAdmin() (contracts/adapters/DlnExternalCallAdapter.sol#79-82) is an unprotected initializer.
Function DlnExternalCallAdapter.onlyDlnDestination() (contracts/adapters/DlnExternalCallAdapter.sol#84-87) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initialize.md
INFO:Detectors:
DlnExternalCallAdapter._execute(bytes32,address,uint256,bytes,address).executionStatus (contracts/adapters/DlnExternalCallAdapter.sol#255) is a local variable never initialized
DlnExternalCallAdapter._execute(bytes32,address,uint256,bytes,address).callResult (contracts/adapters/DlnExternalCallAdapter.sol#256) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
Function DlnExternalCallAdapter._safeTransferETH(address,uint256) (contracts/adapters/DlnExternalCallAdapter.sol#417-420) contains a low level call to a custom address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/call_forward_to_protected.md
INFO:Detectors:
DlnExternalCallAdapter.initialize(address,address) (contracts/adapters/DlnExternalCallAdapter.sol#96-104) should emit an event for:
        - dlnDestination = _dlnDestination (contracts/adapters/DlnExternalCallAdapter.sol#100)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control
INFO:Detectors:
DlnExternalCallAdapter.initialize(address,address)._dlnDestination (contracts/adapters/DlnExternalCallAdapter.sol#96) lacks a zero-check on :
        - dlnDestination = _dlnDestination (contracts/adapters/DlnExternalCallAdapter.sol#100)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Function DlnExternalCallAdapter._getBalance(address) (contracts/adapters/DlnExternalCallAdapter.sol#350-360) has a dubious typecast: address<=IERC20Upgradeable
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO:Detectors:
Pragma version^0.8.17 (contracts/adapters/DlnExternalCallAdapter.sol#2) allows old versions
Pragma version^0.8.0 (contracts/interfaces/IExternalCallAdapter.sol#3) allows old versions
Pragma version^0.8.0 (contracts/interfaces/IExternalCallExecutor.sol#3) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in DlnExternalCallAdapter._safeTransferETH(address,uint256) (contracts/adapters/DlnExternalCallAdapter.sol#417-420):
        - (success) = to.call{value: value}(new bytes(0)) (contracts/adapters/DlnExternalCallAdapter.sol#418)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter DlnExternalCallAdapter.initialize(address,address)._dlnDestination (contracts/adapters/DlnExternalCallAdapter.sol#96) is not in mixedCase
Parameter DlnExternalCallAdapter.initialize(address,address)._executor (contracts/adapters/DlnExternalCallAdapter.sol#96) is not in mixedCase
Parameter DlnExternalCallAdapter.receiveCall(bytes32,address,address,uint256,bytes,address)._orderId (contracts/adapters/DlnExternalCallAdapter.sol#120) is not in mixedCase
Parameter DlnExternalCallAdapter.receiveCall(bytes32,address,address,uint256,bytes,address)._callAuthority (contracts/adapters/DlnExternalCallAdapter.sol#121) is not in mixedCase
Parameter DlnExternalCallAdapter.receiveCall(bytes32,address,address,uint256,bytes,address)._tokenAddress (contracts/adapters/DlnExternalCallAdapter.sol#122) is not in mixedCase
Parameter DlnExternalCallAdapter.receiveCall(bytes32,address,address,uint256,bytes,address)._transferredAmount (contracts/adapters/DlnExternalCallAdapter.sol#123) is not in mixedCase
Parameter DlnExternalCallAdapter.receiveCall(bytes32,address,address,uint256,bytes,address)._externalCall (contracts/adapters/DlnExternalCallAdapter.sol#124) is not in mixedCase
Parameter DlnExternalCallAdapter.receiveCall(bytes32,address,address,uint256,bytes,address)._externalCallRewardBeneficiary (contracts/adapters/DlnExternalCallAdapter.sol#125) is not in mixedCase
Parameter DlnExternalCallAdapter.executeCall(bytes32,address,address,uint256,bytes,address)._orderId (contracts/adapters/DlnExternalCallAdapter.sol#179) is not in mixedCase
Parameter DlnExternalCallAdapter.executeCall(bytes32,address,address,uint256,bytes,address)._callAuthority (contracts/adapters/DlnExternalCallAdapter.sol#180) is not in mixedCase
Parameter DlnExternalCallAdapter.executeCall(bytes32,address,address,uint256,bytes,address)._tokenAddress (contracts/adapters/DlnExternalCallAdapter.sol#181) is not in mixedCase
Parameter DlnExternalCallAdapter.executeCall(bytes32,address,address,uint256,bytes,address)._tokenAmount (contracts/adapters/DlnExternalCallAdapter.sol#182) is not in mixedCase
Parameter DlnExternalCallAdapter.executeCall(bytes32,address,address,uint256,bytes,address)._externalCall (contracts/adapters/DlnExternalCallAdapter.sol#183) is not in mixedCase
Parameter DlnExternalCallAdapter.executeCall(bytes32,address,address,uint256,bytes,address)._rewardBeneficiary (contracts/adapters/DlnExternalCallAdapter.sol#184) is not in mixedCase
Parameter DlnExternalCallAdapter.cancelCall(bytes32,address,address,uint256,address,bytes32)._orderId (contracts/adapters/DlnExternalCallAdapter.sol#207) is not in mixedCase

Parameter DlnExternalCallAdapter.cancelCall(bytes32,address,address,uint256,address,bytes32)._callAuthority (contracts/adapters/DlnExternalCallAdapter.sol#208) is not in mixedCase
Parameter DlnExternalCallAdapter.cancelCall(bytes32,address,address,uint256,address,bytes32)._tokenAddress (contracts/adapters/DlnExternalCallAdapter.sol#209) is not in mixedCase
Parameter DlnExternalCallAdapter.cancelCall(bytes32,address,address,uint256,address,bytes32)._tokenAmount (contracts/adapters/DlnExternalCallAdapter.sol#210) is not in mixedCase
Parameter DlnExternalCallAdapter.cancelCall(bytes32,address,address,uint256,address,bytes32)._recipient (contracts/adapters/DlnExternalCallAdapter.sol#211) is not in mixedCase
Parameter DlnExternalCallAdapter.cancelCall(bytes32,address,address,uint256,address,bytes32)._externalCallHash (contracts/adapters/DlnExternalCallAdapter.sol#212) is not in mixedCase
Parameter DlnExternalCallAdapter.updateExecutor(address)._newExecutor (contracts/adapters/DlnExternalCallAdapter.sol#241) is not in mixedCase
Parameter DlnExternalCallAdapter.getCallId(bytes32,address,address,uint256,bytes)._orderId (contracts/adapters/DlnExternalCallAdapter.sol#375) is not in mixedCase
Parameter DlnExternalCallAdapter.getCallId(bytes32,address,address,uint256,bytes)._callAuthority (contracts/adapters/DlnExternalCallAdapter.sol#376) is not in mixedCase
Parameter DlnExternalCallAdapter.getCallId(bytes32,address,address,uint256,bytes)._tokenAddress (contracts/adapters/DlnExternalCallAdapter.sol#377) is not in mixedCase
Parameter DlnExternalCallAdapter.getCallId(bytes32,address,address,uint256,bytes)._transferredAmount (contracts/adapters/DlnExternalCallAdapter.sol#378) is not in mixedCase
Parameter DlnExternalCallAdapter.getCallId(bytes32,address,address,uint256,bytes)._externalCall (contracts/adapters/DlnExternalCallAdapter.sol#379) is not in mixedCase
Parameter DlnExternalCallAdapter.getCallId(bytes32,address,address,uint256,bytes32)._orderId (contracts/adapters/DlnExternalCallAdapter.sol#394) is not in mixedCase
Parameter DlnExternalCallAdapter.getCallId(bytes32,address,address,uint256,bytes32)._callAuthority (contracts/adapters/DlnExternalCallAdapter.sol#395) is not in mixedCase
Parameter DlnExternalCallAdapter.getCallId(bytes32,address,address,uint256,bytes32)._tokenAddress (contracts/adapters/DlnExternalCallAdapter.sol#396) is not in mixedCase
Parameter DlnExternalCallAdapter.getCallId(bytes32,address,address,uint256,bytes32)._transferredAmount (contracts/adapters/DlnExternalCallAdapter.sol#397) is not in mixedCase
Parameter DlnExternalCallAdapter.getCallId(bytes32,address,address,uint256,bytes32)._externalCallHash (contracts/adapters/DlnExternalCallAdapter.sol#398) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Function DlnExternalCallAdapter.receiveCall(bytes32,address,address,uint256,bytes,address) (contracts/adapters/DlnExternalCallAdapter.sol#119-176) contains magic numbers: 3, 4
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
In a function DlnExternalCallAdapter.receiveCall(bytes32,address,address,uint256,bytes,address) (contracts/adapters/DlnExternalCallAdapter.sol#119-176) variable DlnExternalCallAdapter.tokenBalanceHistory (contracts/adapters/DlnExternalCa
llAdapter.sol#31) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

AAVECallExecutor.sol

```
INFO:Detectors:
ExternalCallExecutorBase._safeTransferETH(address,uint256) (contracts/adapters/ExternalCallExecutorBase.sol#128-131) sends eth to arbitrary user
        Dangerous calls:
        - (success) = to.call{value: value}(new bytes(0)) (contracts/adapters/ExternalCallExecutorBase.sol#129)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Function ExternalCallExecutorBase.constructor() (contracts/adapters/ExternalCallExecutorBase.sol#37-39) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Function AAVECallExecutor.constructor(address) (contracts/adapters/examples/AAVECallExecutor.sol#17-19) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/strange_setter.md
INFO:Detectors:
Function ExternalCallExecutorBase._execute(address,uint256,bytes,uint256) (contracts/adapters/ExternalCallExecutorBase.sol#80-96) contains a low level call to a custom address
Function ExternalCallExecutorBase._safeTransferETH(address,uint256) (contracts/adapters/ExternalCallExecutorBase.sol#128-131) contains a low level call to a custom address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/call_forward_to_protected.md
INFO:Detectors:
ExternalCallExecutorBase._execute(address,uint256,bytes,uint256) (contracts/adapters/ExternalCallExecutorBase.sol#80-96) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.7 (contracts/adapters/ExternalCallExecutorBase.sol#2) allows old versions
Pragma version^0.8.17 (contracts/adapters/examples/AAVECallExecutor.sol#2) allows old versions
Pragma version^0.8.0 (contracts/interfaces/IExternalCallExecutor.sol#3) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in ExternalCallExecutorBase._execute(address,uint256,bytes,uint256) (contracts/adapters/ExternalCallExecutorBase.sol#80-96):
        - (callSucceeded,callResult) = _to.call{gas: _txGas,value: _value}(_data) (contracts/adapters/ExternalCallExecutorBase.sol#90)
        - (callSucceeded,callResult) = _to.call{value: _value}(_data) (contracts/adapters/ExternalCallExecutorBase.sol#93)
Low level call in ExternalCallExecutorBase._safeTransferETH(address,uint256) (contracts/adapters/ExternalCallExecutorBase.sol#128-131):
        - (success) = to.call{value: value}(new bytes(0)) (contracts/adapters/ExternalCallExecutorBase.sol#129)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter ExternalCallExecutorBase.rescueFunds(address,address,uint256)._token (contracts/adapters/ExternalCallExecutorBase.sol#52) is not in mixedCase
Parameter ExternalCallExecutorBase.rescueFunds(address,address,uint256)._recipient (contracts/adapters/ExternalCallExecutorBase.sol#53) is not in mixedCase
Parameter ExternalCallExecutorBase.rescueFunds(address,address,uint256)._amount (contracts/adapters/ExternalCallExecutorBase.sol#54) is not in mixedCase
Parameter AAVECallExecutor.onERC20Received(bytes32,address,uint256,address,bytes)._token (contracts/adapters/examples/AAVECallExecutor.sol#41) is not in mixedCase
Parameter AAVECallExecutor.onERC20Received(bytes32,address,uint256,address,bytes)._transferredAmount (contracts/adapters/examples/AAVECallExecutor.sol#42) is not in mixedCase
Parameter AAVECallExecutor.onERC20Received(bytes32,address,uint256,address,bytes)._payload (contracts/adapters/examples/AAVECallExecutor.sol#44) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Function ExternalCallExecutorBase._execute(address,uint256,bytes,uint256) (contracts/adapters/ExternalCallExecutorBase.sol#80-96) contains magic number: 64
Function AAVECallExecutor.onEtherReceived(bytes32,address,bytes) (contracts/adapters/examples/AAVECallExecutor.sol#26-33) contains magic number: 1337
Function AAVECallExecutor.onERC20Received(bytes32,address,uint256,address,bytes) (contracts/adapters/examples/AAVECallExecutor.sol#39-58) contains magic number: 20
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
```

WidoCallExecutor.sol

```
INFO:Detectors:
ExternalCallExecutorBase._safeTransferETH(address,uint256) (contracts/adapters/ExternalCallExecutorBase.sol#128-131) sends eth to arbitrary user
        Dangerous calls:
        - (success) = to.call{value: value}(new bytes(0)) (contracts/adapters/ExternalCallExecutorBase.sol#129)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Function ExternalCallExecutorBase.constructor() (contracts/adapters/ExternalCallExecutorBase.sol#37-39) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/strange_setter.md
INFO:Detectors:
WidoCallExecutor.onERC20Received(bytes32,address,uint256,address,bytes) (contracts/adapters/examples/WidoCallExecutor.sol#50-74) ignores return value by widoRouter.call(_payload) (contracts/adapters/examples/WidoCallExecutor.sol#65)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-low-level-calls
INFO:Detectors:
Function ExternalCallExecutorBase._execute(address,uint256,bytes,uint256) (contracts/adapters/ExternalCallExecutorBase.sol#80-96) contains a low level call to a custom address
Function ExternalCallExecutorBase._safeTransferETH(address,uint256) (contracts/adapters/ExternalCallExecutorBase.sol#128-131) contains a low level call to a custom address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/call_forward_to_protected.md
INFO:Detectors:
WidoCallExecutor.constructor(address,address,address)._widoRouter (contracts/adapters/examples/WidoCallExecutor.sol#25) lacks a zero-check on :
                - widoRouter = _widoRouter (contracts/adapters/examples/WidoCallExecutor.sol#29)
WidoCallExecutor.constructor(address,address,address)._widoTokenManager (contracts/adapters/examples/WidoCallExecutor.sol#26) lacks a zero-check on :
                - widoTokenManager = _widoTokenManager (contracts/adapters/examples/WidoCallExecutor.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
ExternalCallExecutorBase._execute(address,uint256,bytes,uint256) (contracts/adapters/ExternalCallExecutorBase.sol#80-96) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.7 (contracts/adapters/ExternalCallExecutorBase.sol#2) allows old versions
Pragma version^0.8.17 (contracts/adapters/examples/WidoCallExecutor.sol#2) allows old versions
Pragma version^0.8.0 (contracts/interfaces/IExternalCallExecutor.sol#3) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in ExternalCallExecutorBase._execute(address,uint256,bytes,uint256) (contracts/adapters/ExternalCallExecutorBase.sol#80-96):
        - (callSucceeded,callResult) = _to.call{gas: _txGas,value: _value}(_data) (contracts/adapters/ExternalCallExecutorBase.sol#90)
        - (callSucceeded,callResult) = _to.call{value: _value}(_data) (contracts/adapters/ExternalCallExecutorBase.sol#93)
Low level call in ExternalCallExecutorBase._safeTransferETH(address,uint256) (contracts/adapters/ExternalCallExecutorBase.sol#128-131):
        - (success) = to.call{value: value}(new bytes(0)) (contracts/adapters/ExternalCallExecutorBase.sol#129)
Low level call in WidoCallExecutor.onERC20Received(bytes32,address,uint256,address,bytes) (contracts/adapters/examples/WidoCallExecutor.sol#50-74):
        - widoRouter.call(_payload) (contracts/adapters/examples/WidoCallExecutor.sol#65)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter ExternalCallExecutorBase.rescueFunds(address,address,uint256)._token (contracts/adapters/ExternalCallExecutorBase.sol#52) is not in mixedCase
Parameter ExternalCallExecutorBase.rescueFunds(address,address,uint256)._recipient (contracts/adapters/ExternalCallExecutorBase.sol#53) is not in mixedCase
Parameter ExternalCallExecutorBase.rescueFunds(address,address,uint256)._amount (contracts/adapters/ExternalCallExecutorBase.sol#54) is not in mixedCase
Parameter WidoCallExecutor.onERC20Received(bytes32,address,uint256,address,bytes)._token (contracts/adapters/examples/WidoCallExecutor.sol#52) is not in mixedCase
Parameter WidoCallExecutor.onERC20Received(bytes32,address,uint256,address,bytes)._fallbackAddress (contracts/adapters/examples/WidoCallExecutor.sol#54) is not in mixedCase
Parameter WidoCallExecutor.onERC20Received(bytes32,address,uint256,address,bytes)._payload (contracts/adapters/examples/WidoCallExecutor.sol#55) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Function ExternalCallExecutorBase._execute(address,uint256,bytes,uint256) (contracts/adapters/ExternalCallExecutorBase.sol#80-96) contains magic number: 64
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
WidoCallExecutor.widoRouter (contracts/adapters/examples/WidoCallExecutor.sol#15) should be immutable
WidoCallExecutor.widoTokenManager (contracts/adapters/examples/WidoCallExecutor.sol#16) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

ExternalCallExecutor.sol

```
INFO:Detectors:
ExternalCallExecutorBase._safeTransferETH(address,uint256) (contracts/adapters/ExternalCallExecutorBase.sol#128-131) sends eth to arbitrary user
        Dangerous calls:
        - (success) = to.call{value: value}(new bytes(0)) (contracts/adapters/ExternalCallExecutorBase.sol#129)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Function ExternalCallExecutorBase.constructor() (contracts/adapters/ExternalCallExecutorBase.sol#37-39) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Function ExternalCallExecutor.constructor() (contracts/adapters/ExternalCallExecutor.sol#26) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/strange_setter.md
INFO:Detectors:
ExternalCallExecutor._isValideData(bytes).i (contracts/adapters/ExternalCallExecutor.sol#139) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
Function ExternalCallExecutorBase._execute(address,uint256,bytes,uint256) (contracts/adapters/ExternalCallExecutorBase.sol#80-96) contains a low level call to a custom address
Function ExternalCallExecutorBase._safeTransferETH(address,uint256) (contracts/adapters/ExternalCallExecutorBase.sol#128-131) contains a low level call to a custom address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/call_forward_to_protected.md
INFO:Detectors:
Reentrancy in ExternalCallExecutor.onERC20Received(bytes32,address,uint256,address,bytes) (contracts/adapters/ExternalCallExecutor.sol#72-115):
        External calls:
        - token.safeTransfer(_fallbackAddress,_transferredAmount) (contracts/adapters/ExternalCallExecutor.sol#91)
        Event emitted after the call(s):
        - Debug(here11) (contracts/adapters/ExternalCallExecutor.sol#92)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Function ExternalCallExecutor._isValideData(bytes) (contracts/adapters/ExternalCallExecutor.sol#127-146) has a dubious typecast: bytes4<=bytes4
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO:Detectors:
ExternalCallExecutor._toBytes4(bytes,uint256) (contracts/adapters/ExternalCallExecutor.sol#156-168) uses assembly
        - INLINE ASM (contracts/adapters/ExternalCallExecutor.sol#163-165)
ExternalCallExecutor._toBytes5(bytes) (contracts/adapters/ExternalCallExecutor.sol#170-176) uses assembly
        - INLINE ASM (contracts/adapters/ExternalCallExecutor.sol#175)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
ExternalCallExecutor._toBytes5(bytes) (contracts/adapters/ExternalCallExecutor.sol#170-176) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.17 (contracts/adapters/ExternalCallExecutor.sol#2) allows old versions
Pragma version^0.8.7 (contracts/adapters/ExternalCallExecutorBase.sol#2) allows old versions
Pragma version^0.8.0 (contracts/interfaces/IExternalCallExecutor.sol#3) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
INFO:Detectors:
Low level call in ExternalCallExecutorBase._execute(address,uint256,bytes,uint256) (contracts/adapters/ExternalCallExecutorBase.sol#80-96):
        - (callSucceeded,callResult) = _to.call{gas: _txGas,value: _value}(_data) (contracts/adapters/ExternalCallExecutorBase.sol#90)
        - (callSucceeded,callResult) = _to.call{value: _value}(_data) (contracts/adapters/ExternalCallExecutorBase.sol#93)
Low level call in ExternalCallExecutorBase._safeTransferETH(address,uint256) (contracts/adapters/ExternalCallExecutorBase.sol#128-131):
        - (success) = to.call{value: value}(new bytes(0)) (contracts/adapters/ExternalCallExecutorBase.sol#129)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter ExternalCallExecutor.onEtherReceived(bytes32,address,bytes)._fallbackAddress (contracts/adapters/ExternalCallExecutor.sol#35) is not in mixedCase
Parameter ExternalCallExecutor.onEtherReceived(bytes32,address,bytes)._payload (contracts/adapters/ExternalCallExecutor.sol#36) is not in mixedCase
Parameter ExternalCallExecutor.onERC20Received(bytes32,address,uint256,address,bytes)._token (contracts/adapters/ExternalCallExecutor.sol#74) is not in mixedCase
Parameter ExternalCallExecutor.onERC20Received(bytes32,address,uint256,address,bytes)._transferredAmount (contracts/adapters/ExternalCallExecutor.sol#75) is not in mixedCase
Parameter ExternalCallExecutor.onERC20Received(bytes32,address,uint256,address,bytes)._fallbackAddress (contracts/adapters/ExternalCallExecutor.sol#76) is not in mixedCase
Parameter ExternalCallExecutor.onERC20Received(bytes32,address,uint256,address,bytes)._payload (contracts/adapters/ExternalCallExecutor.sol#77) is not in mixedCase
Parameter ExternalCallExecutorBase.rescueFunds(address,address,uint256)._token (contracts/adapters/ExternalCallExecutorBase.sol#52) is not in mixedCase
Parameter ExternalCallExecutorBase.rescueFunds(address,address,uint256)._recipient (contracts/adapters/ExternalCallExecutorBase.sol#53) is not in mixedCase
Parameter ExternalCallExecutorBase.rescueFunds(address,address,uint256)._amount (contracts/adapters/ExternalCallExecutorBase.sol#54) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Function ExternalCallExecutorBase._execute(address,uint256,bytes,uint256) (contracts/adapters/ExternalCallExecutorBase.sol#80-96) contains magic number: 64
Function ExternalCallExecutor._toBytes4(bytes,uint256) (contracts/adapters/ExternalCallExecutor.sol#156-168) contains magic number: 4
Function ExternalCallExecutor._toBytes5(bytes) (contracts/adapters/ExternalCallExecutor.sol#170-176) contains magic number: 4
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
```

- The unprotected initialize and strange setter issues were checked individually and are false positives.
- `DLNExternalCallAdapter` and `ExternalCallExecutorBase` send native assets to an arbitrary destination, although this is intended and implemented correctly.
- No major issues found by Slither.

THANK YOU FOR CHOOSING

// HALBORN