DeBridge - DLN External Call

Smart Contract Security Assessment

Prepared by: **Halborn**

Date of Engagement: **July 3rd, 2023 - July 21st, 2023**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	07/03/2023	Roberto Reigada
0.2	Document Updates	07/21/2023	Roberto Reigada
0.3	Draft Review	07/21/2023	Gokberk Gulgun
0.4	Draft Review	07/21/2023	Gabi Urrutia
1.0	Remediation Plan	08/31/2023	Roberto Reigada
1.1	Remediation Plan Review	09/04/2023	Gokberk Gulgun
1.2	Remediation Plan Review	09/04/2023	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Gokberk Gulgun	Halborn	Gokberk.Gulgun@halborn.com
Roberto Reigada	Halborn	Roberto.Reigada@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

DeBridge engaged Halborn to conduct a security assessment on their smart contracts beginning on July 3rd, 2023 and ending on July 21st, 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 assessment 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.
- Scanning of solidity files for vulnerabilities, security hot-spots or bugs. (MythX)
- 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)	1
	Specific (AO:S)	0.2
Attack Cost (AC)	Low (AC:L)	1
	Medium (AC:M)	0.67
	High (AC:H)	0.33
Attack Complexity (AX)	Low (AX:L)	1
	Medium (AX:M)	0.67
	High (AX:H)	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)	1
	Partial (R:P)	0.5
	Full (R:F)	0.25
Scope (s)	Changed (S:C)	1.25
	Unchanged (S:U)	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:

PR: #82
GitHub repository: debridge-finance/dln-evm/
Verified Commit ID: 8dd63480e9caeea4de11e7ba8912398f5d119dec
Fixed Commit ID: 5572654e906c95679f2b60f03796554d9d281335

Smart contracts in scope:

- DlnBase.sol
- DlnDestination.sol
- DlnSource.sol
- ExternalCallExecutor.sol
- DlnExternalCallAdapter.sol
- AAVECallExecutor.sol
- Executor.sol
- BytesLib.sol
- DlnExternalCallLib.sol
- DlnOrderLib.sol
- SafeCast.sol

3. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	2	0

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) INCREASEALLOWANCE SELECTOR IS MISSING IN PROHIBITED SELECTORS	Low (2.5)	SOLVED - 07/17/2023
(HAL-02) MISSING LOGIC TO DISTRIBUTE LIQUIDITY AMONG DIFFERENT WALLETS	Low (2.5)	FUTURE RELEASE

FINDINGS & TECH DETAILS

4.1 (HAL-01) INCREASEALLOWANCE SELECTOR IS MISSING IN PROHIBITED SELECTORS - LOW (2.5)

Commit IDs affected:
- 8dd63480e9caeea4de11e7ba8912398f5d119dec

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 1: ExternalCallExecutor.sol (Lines 119-121)

```
115 function _isValideData(bytes memory _data) internal returns (bool)
     ↳ {
116     bytes4 functionSelector = _toBytes4(_data, 0);
117 
118     bytes4[3] memory prohibitedSelectors = [
119         bytes4(0x095ea7b3), // approve
120         bytes4(0x23b872dd), // transferFrom
121         bytes4(0xa9059cbb) // transfer
122     ];
123 
124     for (uint256 i = 0; i < prohibitedSelectors.length; i++) {
125         if (prohibitedSelectors[i] == functionSelector) {
126             emit ProhibitedFunctionSelector(functionSelector);
127             return false;
128         }
129     }
130     return true;
131 }
```

Although, the increaseAllowance(address,uint256) selector, which is part of the ERC20 standard, is not present in the prohibitedSelectors array.

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 add the `increaseAllowance(address,uint256)` selector to the `ExternalCallExecutor` `prohibitedSelectors` array.

Remediation Plan:

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

Commit ID : 5572654e906c95679f2b60f03796554d9d281335.

4.2 (HAL-02) MISSING LOGIC TO DISTRIBUTE LIQUIDITY AMONG DIFFERENT WALLETS - LOW (2.5)

Commit IDs affected:
- 8dd63480e9caeea4de11e7ba8912398f5d119dec

Description:

One of the business requirements that should be met by the DLN External Call functionality is:

“As a DLN user, I want to be able to form an order, after fulfillment of which, my liquidity in the target chain would be, distributed among the wallets, a list of which I provide in the dln-extcall in a way I describe in the dln-extcall”

Although, the contracts do not implement anywhere this business requirement. The only way to achieve this currently is using some type of external Multisend contract.

BVSS:

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

Recommendation:

Consider adding logic to allow DLN users to form an order that after fulfillment the liquidity in the target chain would be distributed in a concrete way among a list of wallets provided in the dln-extcall.

Remediation Plan:

PENDING: The DeBridge team states that they plan to develop an executor with multi-send logic.

RECOMMENDATIONS OVERVIEW

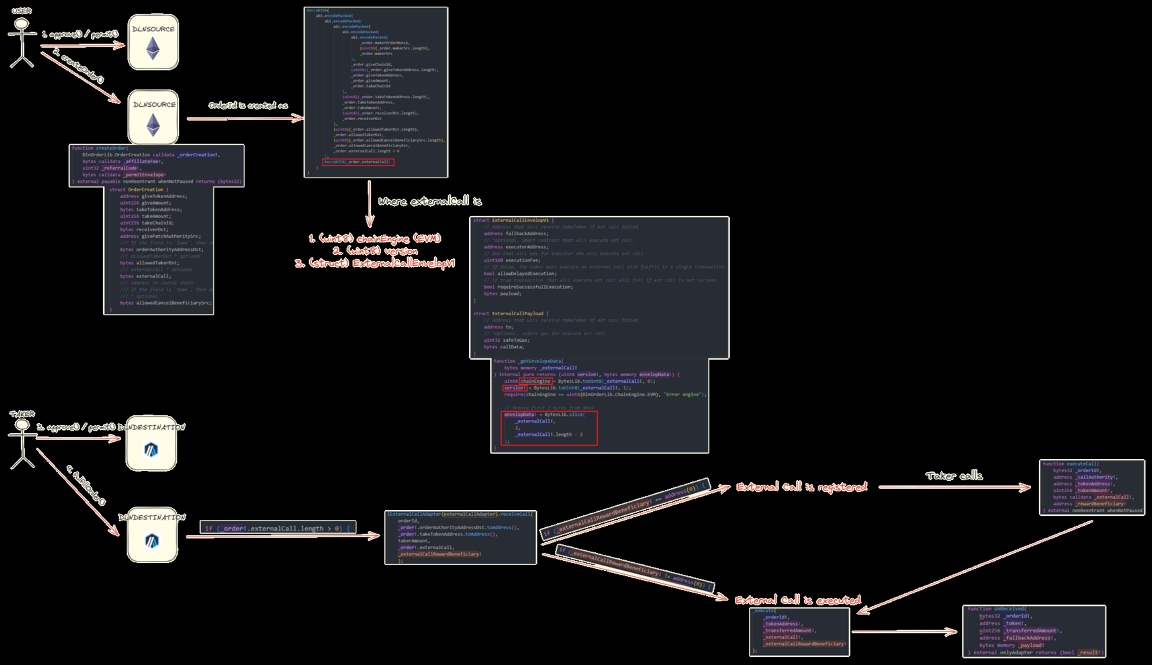
1. Add the `increaseAllowance(address,uint256)` selector to the `ExternalCallExecutor` `prohibitedSelectors` array.
2. Add logic to allow DLN users to form an order that after fulfillment the liquidity in the target chain would be distributed in a concrete way among a list of wallets provided in the dln-extcall.

MANUAL TESTING

A critical part of the manual testing was performed using code review and manual simulation of contract behavior. Particular focus was placed on five key areas:

1. The createOrder() function was tested thoroughly. The sequence of actions --- calling patchOrderTake() to incentivize takers to fulfill the order, front-running the taker transaction with a sendEvmOrderCancel() call plus attempting to create the same order through createOrder() --- was performed. Our tests confirmed that this exploit is prevented, as MasterNonce increases every time an order is created.
2. Recalling attempts of patchOrderTake() function with a lower _newSubtrahend were tested. Our review confirmed that such attempts are effectively prevented, as the code `if (takePatches[orderId] >= _newSubtrahend)revert WrongArgument();` reliably throws an exception for this scenario, ensuring appropriate subtrahend handling.
3. All external functions, especially the payable onReceived() function, were tested for reentrancy. The line of code `_result = _execute(executionData.to, msg.value, executionData.callData, executionData.safeTxGas);` was checked to ensure that it could not be exploited to re-enter and fulfill another order, which could potentially drain the contract's balance. Tests confirmed that the nonReentrant modifier effectively prevents any reentrancy, thus safeguarding the contract's assets.
4. Tests were performed to confirm that approvals are always reset back to 0 after use, preventing possible Denial of Service scenarios, especially with tokens like USDT.
5. Order Id Hashing: abi.encodePacked usage was checked. This aimed to ascertain that it was impossible to brute-force the same orderId hash with different data. Our manual review revealed that no variable-length arguments were used together.

Below, the normal DLN External Call flow can be found:



```
function createOrder(
    DlnOrderLib.OrderCreation calldata _orderCreation,
    bytes calldata _affiliateFee,
    uint32 _referralCode,
    bytes calldata _permitEnvelope
) external payable nonReentrant whenNotPaused returns (bytes32)
```

```
struct OrderCreation {
    address giveTokenAddress;
    uint256 giveAmount;
    bytes takeTokenAddress;
    uint256 takeAmount;
    uint256 takeChainId;
    bytes receiverDst;
    address givePatchAuthoritySrc;
    /// If the field is `Some`, then o
    bytes orderAuthorityAddressDst;
    /// allowedTakerDst * optional
    bytes allowedTakerDst;
    /// externalCall * optional
    bytes externalCall;
    /// Address in source chain
    /// If the field is `Some`, then o
    /// * optional
    bytes allowedCancelBeneficiarySrc;
}
```

```
keccak256(
    abi.encodePacked(
        abi.encodePacked(
            abi.encodePacked(
                abi.encodePacked(
                    abi.encodePacked(
                        _order.makerOrderNonce,
                        (uint8)(_order.makerSrc.length),
                        _order.makerSrc
                    ),
                    _order.giveChainId,
                    (uint8)(_order.giveTokenAddress.length),
                    _order.giveTokenAddress,
                    _order.giveAmount,
                    _order.takeChainId
                ),
                (uint8)(_order.takeTokenAddress.length),
                _order.takeTokenAddress,
                _order.takeAmount,
                (uint8)(_order.receiverDst.length),
                _order.receiverDst
            ),
            (uint8)(_order.allowedTakerDst.length),
            _order.allowedTakerDst,
            (uint8)(_order.allowedCancelBeneficiarySrc.length),
            _order.allowedCancelBeneficiarySrc,
            _order.externalCall.length > 0
        ),
        keccak256(_order.externalCall)
    )
)
```

```
struct ExternalCallEnvelopV1 {
    // Address that will receive takeToken if ext call failed
    address fallbackAddress;
    // *optional. Smart contract that will execute ext call.
    address executorAddress;
    // fee that will pay for executor who will execute ext call
    uint160 executionFee;
    // If false, the taker must execute an external call with fulfill in a single transaction.
    bool allowDelayedExecution;
    // if true transaction that will execute ext call will fail if ext call is not success
    bool requireSuccessfullExecution;
    bytes payload;
}

struct ExternalCallPayload {
    // Address that will receive takeToken if ext call failed
    address to;
    // *optional. safeTx gas for execute ext call
    uint32 safeTxGas;
    bytes callData;
}
```

```
function _getEnvelopeData(
    bytes memory _externalCall
) internal pure returns (uint8 version, bytes memory envelopData) {
    uint8 chainEngine = BytesLib.toUint8(_externalCall, 0);
    version = BytesLib.toUint8(_externalCall, 1);
    require(chainEngine == uint8(DlnOrderLib.ChainEngine.EVM), "Error engine");

    // Remove first 2 bytes from data
    envelopData = BytesLib.slice(
        _externalCall,
        2,
        _externalCall.length - 2
    );
}
```

```
if (_order.externalCall.length > 0) {
```

```
IExternalCallAdapter(externalCallAdapter).receiveCall(
    orderId,
    _order.orderAuthorityAddressDst.toAddress(),
    _order.takeTokenAddress.toAddress(),
    takerAmount,
    _order.externalCall,
    _externalCallRewardBeneficiary
);
```

```
if (_externalCallRewardBeneficiary == address(0)) {
```

```
if (_externalCallRewardBeneficiary != address(0)) {
```

```
function executeCall(
    bytes32 _orderId,
    address _callAuthority,
    address _tokenAddress,
    uint256 _tokenAmount,
    bytes calldata _externalCall,
    address _rewardBeneficiary
) external nonReentrant whenNotPaused
```

```
_execute(
    _orderId,
    _tokenAddress,
    _transferredAmount,
    _externalCall,
    _externalCallRewardBeneficiary
);
```

```
function onReceived(
    bytes32 _orderId,
    address _token,
    uint256 _transferredAmount,
    address _fallbackAddress,
    bytes memory _payload
) external onlyAdapter returns (bool _result)
```

MANUAL TESTING

AUTOMATED TESTING

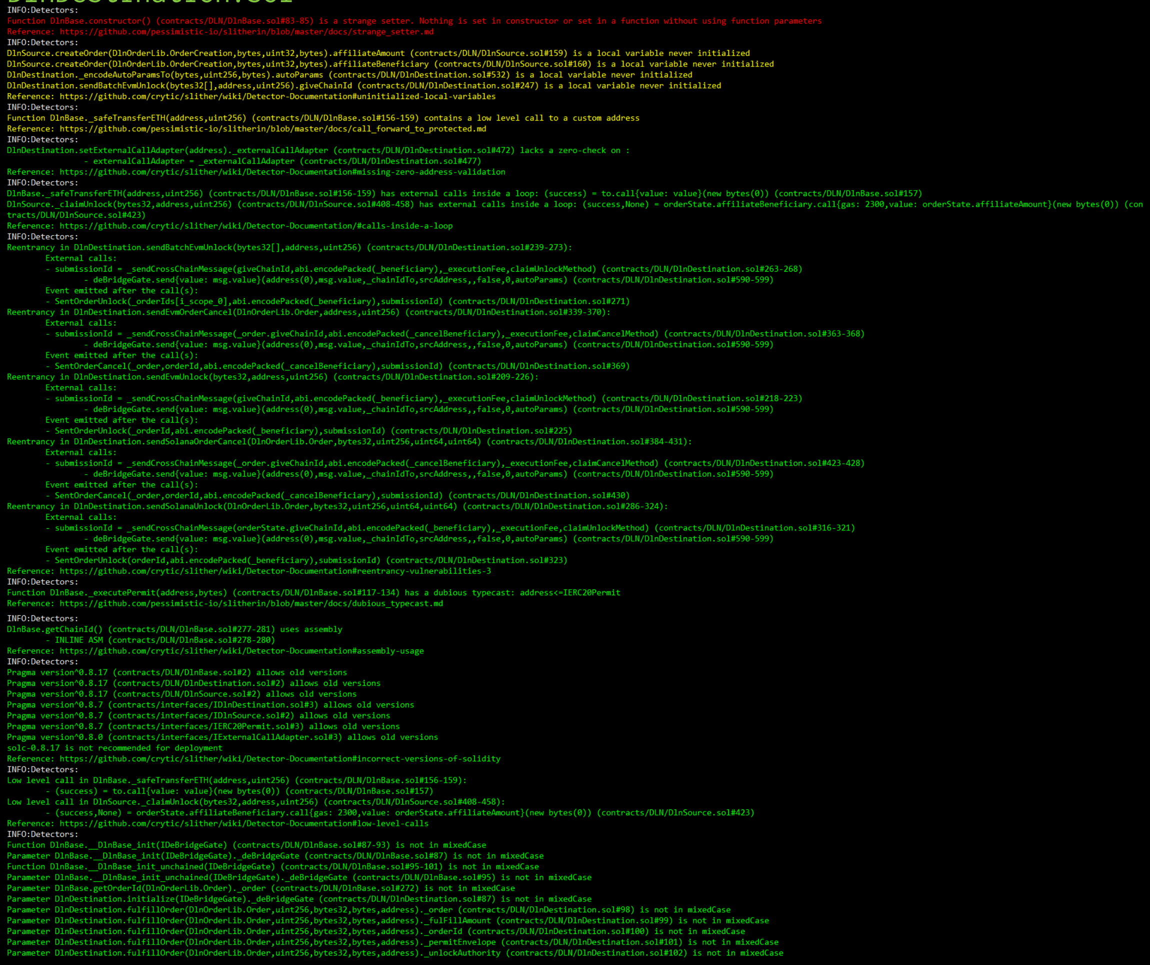
7.1 STATIC ANALYSIS REPORT

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:
Function DlnBase.constructor() (contracts/DLN/DlnBase.sol#83-85) 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:
DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes).affiliateAmount (contracts/DLN/DlnSource.sol#159) is a local variable never initialized
DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes).affiliateBeneficiary (contracts/DLN/DlnSource.sol#160) is a local variable never initialized
DlnDestination._encodeAutoParamsTo(bytes,uint256,bytes).autoParams (contracts/DLN/DlnDestination.sol#532) is a local variable never initialized
DlnDestination.sendBatchEvmUnlock(bytes32[],address,uint256).giveChainId (contracts/DLN/DlnDestination.sol#247) 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#156-159) 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#472) lacks a zero-check on :
                - externalCallAdapter = _externalCallAdapter (contracts/DLN/DlnDestination.sol#477)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
DlnBase._safeTransferETH(address,uint256) (contracts/DLN/DlnBase.sol#156-159) has external calls inside a loop: (success) = to.call{value: value}(new bytes(0)) (contracts/DLN/DlnBase.sol#157)
DlnSource._claimUnlock(bytes32,address,uint256) (contracts/DLN/DlnSource.sol#408-458) has external calls inside a loop: (success,None) = orderState.affiliateBeneficiary.call{gas: 2300,value: orderState.affiliateAmount}(new bytes(0)) (con
tracts/DLN/DlnSource.sol#423)
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#239-273):
        External calls:
        - submissionId = _sendCrossChainMessage(giveChainId,abi.encodePacked(_beneficiary),_executionFee,claimUnlockMethod) (contracts/DLN/DlnDestination.sol#263-268)
                - deBridgeGate.send{value: msg.value}(address(0),msg.value,_chainIdTo,srcAddress,,false,0,autoParams) (contracts/DLN/DlnDestination.sol#590-599)
        Event emitted after the call(s):
        - SentOrderUnlock(_orderIds[i_scope_0],abi.encodePacked(_beneficiary),submissionId) (contracts/DLN/DlnDestination.sol#271)
Reentrancy in DlnDestination.sendEvmOrderCancel(DlnOrderLib.Order,address,uint256) (contracts/DLN/DlnDestination.sol#339-370):
        External calls:
        - submissionId = _sendCrossChainMessage(_order.giveChainId,abi.encodePacked(_cancelBeneficiary),_executionFee,claimCancelMethod) (contracts/DLN/DlnDestination.sol#363-368)
                - deBridgeGate.send{value: msg.value}(address(0),msg.value,_chainIdTo,srcAddress,,false,0,autoParams) (contracts/DLN/DlnDestination.sol#590-599)
        Event emitted after the call(s):
        - SentOrderCancel(_order,orderId,abi.encodePacked(_cancelBeneficiary),submissionId) (contracts/DLN/DlnDestination.sol#369)
Reentrancy in DlnDestination.sendEvmUnlock(bytes32,address,uint256) (contracts/DLN/DlnDestination.sol#209-226):
        External calls:
        - submissionId = _sendCrossChainMessage(giveChainId,abi.encodePacked(_beneficiary),_executionFee,claimUnlockMethod) (contracts/DLN/DlnDestination.sol#218-223)
                - deBridgeGate.send{value: msg.value}(address(0),msg.value,_chainIdTo,srcAddress,,false,0,autoParams) (contracts/DLN/DlnDestination.sol#590-599)
        Event emitted after the call(s):
        - SentOrderUnlock(_orderId,abi.encodePacked(_beneficiary),submissionId) (contracts/DLN/DlnDestination.sol#225)
Reentrancy in DlnDestination.sendSolanaOrderCancel(DlnOrderLib.Order,bytes32,uint256,uint64,uint64) (contracts/DLN/DlnDestination.sol#384-431):
        External calls:
        - submissionId = _sendCrossChainMessage(_order.giveChainId,abi.encodePacked(_cancelBeneficiary),_executionFee,claimCancelMethod) (contracts/DLN/DlnDestination.sol#423-428)
                - deBridgeGate.send{value: msg.value}(address(0),msg.value,_chainIdTo,srcAddress,,false,0,autoParams) (contracts/DLN/DlnDestination.sol#590-599)
        Event emitted after the call(s):
        - SentOrderCancel(_order,orderId,abi.encodePacked(_cancelBeneficiary),submissionId) (contracts/DLN/DlnDestination.sol#430)
Reentrancy in DlnDestination.sendSolanaUnlock(DlnOrderLib.Order,bytes32,uint256,uint64,uint64) (contracts/DLN/DlnDestination.sol#286-324):
        External calls:
        - submissionId = _sendCrossChainMessage(orderState.giveChainId,abi.encodePacked(_beneficiary),_executionFee,claimUnlockMethod) (contracts/DLN/DlnDestination.sol#316-321)
                - deBridgeGate.send{value: msg.value}(address(0),msg.value,_chainIdTo,srcAddress,,false,0,autoParams) (contracts/DLN/DlnDestination.sol#590-599)
        Event emitted after the call(s):
        - SentOrderUnlock(orderId,abi.encodePacked(_beneficiary),submissionId) (contracts/DLN/DlnDestination.sol#323)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Function DlnBase._executePermit(address,bytes) (contracts/DLN/DlnBase.sol#117-134) has a dubious typecast: address<=IERC20Permit
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md

INFO:Detectors:
DlnBase.getChainId() (contracts/DLN/DlnBase.sol#277-281) uses assembly
        - INLINE ASM (contracts/DLN/DlnBase.sol#278-280)
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/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#156-159):
        - (success) = to.call{value: value}(new bytes(0)) (contracts/DLN/DlnBase.sol#157)
Low level call in DlnSource._claimUnlock(bytes32,address,uint256) (contracts/DLN/DlnSource.sol#408-458):
        - (success,None) = orderState.affiliateBeneficiary.call{gas: 2300,value: orderState.affiliateAmount}(new bytes(0)) (contracts/DLN/DlnSource.sol#423)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function DlnBase.__DlnBase_init(IDeBridgeGate) (contracts/DLN/DlnBase.sol#87-93) is not in mixedCase
Parameter DlnBase.__DlnBase_init(IDeBridgeGate)._deBridgeGate (contracts/DLN/DlnBase.sol#87) is not in mixedCase
Function DlnBase.__DlnBase_init_unchained(IDeBridgeGate) (contracts/DLN/DlnBase.sol#95-101) is not in mixedCase
Parameter DlnBase.__DlnBase_init_unchained(IDeBridgeGate)._deBridgeGate (contracts/DLN/DlnBase.sol#95) is not in mixedCase
Parameter DlnBase.getOrderId(DlnOrderLib.Order)._order (contracts/DLN/DlnBase.sol#272) is not in mixedCase
Parameter DlnDestination.initialize(IDeBridgeGate)._deBridgeGate (contracts/DLN/DlnDestination.sol#87) is not in mixedCase
Parameter DlnDestination.fulfillOrder(DlnOrderLib.Order,uint256,bytes32,bytes,address)._order (contracts/DLN/DlnDestination.sol#98) is not in mixedCase
Parameter DlnDestination.fulfillOrder(DlnOrderLib.Order,uint256,bytes32,bytes,address)._fulFillAmount (contracts/DLN/DlnDestination.sol#99) is not in mixedCase
Parameter DlnDestination.fulfillOrder(DlnOrderLib.Order,uint256,bytes32,bytes,address)._orderId (contracts/DLN/DlnDestination.sol#100) is not in mixedCase
Parameter DlnDestination.fulfillOrder(DlnOrderLib.Order,uint256,bytes32,bytes,address)._permitEnvelope (contracts/DLN/DlnDestination.sol#101) is not in mixedCase
Parameter DlnDestination.fulfillOrder(DlnOrderLib.Order,uint256,bytes32,bytes,address)._unlockAuthority (contracts/DLN/DlnDestination.sol#102) is not in mixedCase
Parameter DlnDestination.fulfillOrder(DlnOrderLib.Order,uint256,bytes32,bytes,address,address)._order (contracts/DLN/DlnDestination.sol#115) is not in mixedCase
Parameter DlnDestination.fulfillOrder(DlnOrderLib.Order,uint256,bytes32,bytes,address,address)._fulFillAmount (contracts/DLN/DlnDestination.sol#116) is not in mixedCase
Parameter DlnDestination.fulfillOrder(DlnOrderLib.Order,uint256,bytes32,bytes,address,address)._orderId (contracts/DLN/DlnDestination.sol#117) is not in mixedCase
Parameter DlnDestination.fulfillOrder(DlnOrderLib.Order,uint256,bytes32,bytes,address,address)._permitEnvelope (contracts/DLN/DlnDestination.sol#118) is not in mixedCase
Parameter DlnDestination.fulfillOrder(DlnOrderLib.Order,uint256,bytes32,bytes,address,address)._unlockAuthority (contracts/DLN/DlnDestination.sol#119) is not in mixedCase
Parameter DlnDestination.fulfillOrder(DlnOrderLib.Order,uint256,bytes32,bytes,address,address)._externalCallRewardBeneficiary (contracts/DLN/DlnDestination.sol#120) is not in mixedCase
Parameter DlnDestination.sendEvmUnlock(bytes32,address,uint256)._orderId (contracts/DLN/DlnDestination.sol#210) is not in mixedCase
Parameter DlnDestination.sendEvmUnlock(bytes32,address,uint256)._beneficiary (contracts/DLN/DlnDestination.sol#211) is not in mixedCase
Parameter DlnDestination.sendEvmUnlock(bytes32,address,uint256)._executionFee (contracts/DLN/DlnDestination.sol#212) is not in mixedCase
Parameter DlnDestination.sendBatchEvmUnlock(bytes32[],address,uint256)._orderIds (contracts/DLN/DlnDestination.sol#240) is not in mixedCase
Parameter DlnDestination.sendBatchEvmUnlock(bytes32[],address,uint256)._beneficiary (contracts/DLN/DlnDestination.sol#241) is not in mixedCase
Parameter DlnDestination.sendBatchEvmUnlock(bytes32[],address,uint256)._executionFee (contracts/DLN/DlnDestination.sol#242) is not in mixedCase
Parameter DlnDestination.sendSolanaUnlock(DlnOrderLib.Order,bytes32,uint256,uint64,uint64)._order (contracts/DLN/DlnDestination.sol#287) is not in mixedCase
Parameter DlnDestination.sendSolanaUnlock(DlnOrderLib.Order,bytes32,uint256,uint64,uint64)._beneficiary (contracts/DLN/DlnDestination.sol#288) is not in mixedCase
Parameter DlnDestination.sendSolanaUnlock(DlnOrderLib.Order,bytes32,uint256,uint64,uint64)._executionFee (contracts/DLN/DlnDestination.sol#289) is not in mixedCase
Parameter DlnDestination.sendSolanaUnlock(DlnOrderLib.Order,bytes32,uint256,uint64,uint64)._solanaExternalCallReward1 (contracts/DLN/DlnDestination.sol#290) is not in mixedCase
Parameter DlnDestination.sendSolanaUnlock(DlnOrderLib.Order,bytes32,uint256,uint64,uint64)._solanaExternalCallReward2 (contracts/DLN/DlnDestination.sol#291) is not in mixedCase
Parameter DlnDestination.sendEvmOrderCancel(DlnOrderLib.Order,address,uint256)._order (contracts/DLN/DlnDestination.sol#340) is not in mixedCase
Parameter DlnDestination.sendEvmOrderCancel(DlnOrderLib.Order,address,uint256)._cancelBeneficiary (contracts/DLN/DlnDestination.sol#341) is not in mixedCase
Parameter DlnDestination.sendEvmOrderCancel(DlnOrderLib.Order,address,uint256)._executionFee (contracts/DLN/DlnDestination.sol#342) is not in mixedCase
Parameter DlnDestination.sendSolanaOrderCancel(DlnOrderLib.Order,bytes32,uint256,uint64,uint64)._order (contracts/DLN/DlnDestination.sol#385) is not in mixedCase
Parameter DlnDestination.sendSolanaOrderCancel(DlnOrderLib.Order,bytes32,uint256,uint64,uint64)._cancelBeneficiary (contracts/DLN/DlnDestination.sol#386) is not in mixedCase
Parameter DlnDestination.sendSolanaOrderCancel(DlnOrderLib.Order,bytes32,uint256,uint64,uint64)._executionFee (contracts/DLN/DlnDestination.sol#387) is not in mixedCase
Parameter DlnDestination.sendSolanaOrderCancel(DlnOrderLib.Order,bytes32,uint256,uint64,uint64)._reward1 (contracts/DLN/DlnDestination.sol#388) is not in mixedCase
Parameter DlnDestination.sendSolanaOrderCancel(DlnOrderLib.Order,bytes32,uint256,uint64,uint64)._reward2 (contracts/DLN/DlnDestination.sol#389) is not in mixedCase
Parameter DlnDestination.patchOrderTake(DlnOrderLib.Order,uint256)._order (contracts/DLN/DlnDestination.sol#442) is not in mixedCase
Parameter DlnDestination.patchOrderTake(DlnOrderLib.Order,uint256)._newSubtrahend (contracts/DLN/DlnDestination.sol#442) is not in mixedCase
Parameter DlnDestination.setDlnSourceAddress(uint256,bytes,DlnOrderLib.ChainEngine)._chainIdFrom (contracts/DLN/DlnDestination.sol#462) is not in mixedCase
Parameter DlnDestination.setDlnSourceAddress(uint256,bytes,DlnOrderLib.ChainEngine)._dlnSourceAddress (contracts/DLN/DlnDestination.sol#462) is not in mixedCase
Parameter DlnDestination.setDlnSourceAddress(uint256,bytes,DlnOrderLib.ChainEngine)._chainEngine (contracts/DLN/DlnDestination.sol#462) is not in mixedCase
Parameter DlnDestination.setExternalCallAdapter(address)._externalCallAdapter (contracts/DLN/DlnDestination.sol#472) is not in mixedCase
Parameter DlnSource.initialize(IDeBridgeGate,uint88,uint16)._deBridgeGate (contracts/DLN/DlnSource.sol#136) is not in mixedCase
Parameter DlnSource.initialize(IDeBridgeGate,uint88,uint16)._globalFixedNativeFee (contracts/DLN/DlnSource.sol#137) is not in mixedCase
Parameter DlnSource.initialize(IDeBridgeGate,uint88,uint16)._globalTransferFeeBps (contracts/DLN/DlnSource.sol#138) is not in mixedCase
Parameter DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes)._orderCreation (contracts/DLN/DlnSource.sol#153) is not in mixedCase
Parameter DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes)._affiliateFee (contracts/DLN/DlnSource.sol#154) is not in mixedCase
Parameter DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes)._referralCode (contracts/DLN/DlnSource.sol#155) is not in mixedCase
Parameter DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes)._permitEnvelope (contracts/DLN/DlnSource.sol#156) is not in mixedCase
Parameter DlnSource.claimBatchUnlock(bytes32[],address)._orderIds (contracts/DLN/DlnSource.sol#223) is not in mixedCase
Parameter DlnSource.claimBatchUnlock(bytes32[],address)._beneficiary (contracts/DLN/DlnSource.sol#223) is not in mixedCase
Parameter DlnSource.claimUnlock(bytes32,address)._orderId (contracts/DLN/DlnSource.sol#240) is not in mixedCase
Parameter DlnSource.claimUnlock(bytes32,address)._beneficiary (contracts/DLN/DlnSource.sol#240) is not in mixedCase
Parameter DlnSource.claimBatchCancel(bytes32[],address)._orderIds (contracts/DLN/DlnSource.sol#254) is not in mixedCase
Parameter DlnSource.claimBatchCancel(bytes32[],address)._beneficiary (contracts/DLN/DlnSource.sol#254) is not in mixedCase
Parameter DlnSource.claimCancel(bytes32,address)._orderId (contracts/DLN/DlnSource.sol#271) is not in mixedCase
Parameter DlnSource.claimCancel(bytes32,address)._beneficiary (contracts/DLN/DlnSource.sol#271) is not in mixedCase
Parameter DlnSource.patchOrderGive(DlnOrderLib.Order,uint256,bytes)._order (contracts/DLN/DlnSource.sol#290) is not in mixedCase
Parameter DlnSource.patchOrderGive(DlnOrderLib.Order,uint256,bytes)._addGiveAmount (contracts/DLN/DlnSource.sol#291) is not in mixedCase
Parameter DlnSource.patchOrderGive(DlnOrderLib.Order,uint256,bytes)._permitEnvelope (contracts/DLN/DlnSource.sol#292) is not in mixedCase
Parameter DlnSource.setDlnDestinationAddress(uint256,bytes,DlnOrderLib.ChainEngine)._chainIdTo (contracts/DLN/DlnSource.sol#327) is not in mixedCase
Parameter DlnSource.setDlnDestinationAddress(uint256,bytes,DlnOrderLib.ChainEngine)._dlnDestinationAddress (contracts/DLN/DlnSource.sol#327) is not in mixedCase
Parameter DlnSource.setDlnDestinationAddress(uint256,bytes,DlnOrderLib.ChainEngine)._chainEngine (contracts/DLN/DlnSource.sol#327) is not in mixedCase
Parameter DlnSource.withdrawFee(address[],address)._tokens (contracts/DLN/DlnSource.sol#340) is not in mixedCase
Parameter DlnSource.withdrawFee(address[],address)._beneficiary (contracts/DLN/DlnSource.sol#340) is not in mixedCase
Parameter DlnSource.updateGlobalFee(uint88,uint16)._globalFixedNativeFee (contracts/DLN/DlnSource.sol#355) is not in mixedCase
Parameter DlnSource.updateGlobalFee(uint88,uint16)._globalTransferFeeBps (contracts/DLN/DlnSource.sol#356) is not in mixedCase

Parameter DlnSource.validateCreationOrder(DlnOrderLib.OrderCreation,address)._orderCreation (contracts/DLN/DlnSource.sol#367) is not in mixedCase
Parameter DlnSource.validateCreationOrder(DlnOrderLib.OrderCreation,address)._sender (contracts/DLN/DlnSource.sol#367) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable DlnDestination._validateSolanaRewards(uint256,uint256,uint64,uint64)._solanaExternalCallReward1 (contracts/DLN/DlnDestination.sol#553) is too similar to DlnDestination.sendSolanaUnlock(DlnOrderLib.Order,bytes32,uint256,uint64,ui
nt64)._solanaExternalCallReward2 (contracts/DLN/DlnDestination.sol#291)
Variable DlnDestination.sendSolanaUnlock(DlnOrderLib.Order,bytes32,uint256,uint64,uint64)._solanaExternalCallReward1 (contracts/DLN/DlnDestination.sol#290) is too similar to DlnDestination.sendSolanaUnlock(DlnOrderLib.Order,bytes32,uint2
56,uint64,uint64)._solanaExternalCallReward2 (contracts/DLN/DlnDestination.sol#291)
Variable DlnDestination.sendSolanaUnlock(DlnOrderLib.Order,bytes32,uint256,uint64,uint64)._solanaExternalCallReward1 (contracts/DLN/DlnDestination.sol#290) is too similar to DlnDestination._validateSolanaRewards(uint256,uint256,uint64,ui
nt64)._solanaExternalCallReward2 (contracts/DLN/DlnDestination.sol#554)
Variable DlnDestination._validateSolanaRewards(uint256,uint256,uint64,uint64)._solanaExternalCallReward1 (contracts/DLN/DlnDestination.sol#553) is too similar to DlnDestination._validateSolanaRewards(uint256,uint256,uint64,uint64)._solan
aExternalCallReward2 (contracts/DLN/DlnDestination.sol#554)
Variable DlnDestination.setDlnSourceAddress(uint256,bytes,DlnOrderLib.ChainEngine)._chainEngine (contracts/DLN/DlnDestination.sol#462) is too similar to DlnBase.chainEngines (contracts/DLN/DlnBase.sol#43)
Variable DlnSource.setDlnDestinationAddress(uint256,bytes,DlnOrderLib.ChainEngine)._chainEngine (contracts/DLN/DlnSource.sol#327) is too similar to DlnBase.chainEngines (contracts/DLN/DlnBase.sol#43)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
Function DlnBase._executePermit(address,bytes) (contracts/DLN/DlnBase.sol#117-134) contains magic numbers: 32, 64
Function DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes) (contracts/DLN/DlnSource.sol#152-216) contains magic numbers: 52, 20
Function DlnSource._claimUnlock(bytes32,address,uint256) (contracts/DLN/DlnSource.sol#408-458) contains magic number: 2300
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
In a function DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes) (contracts/DLN/DlnSource.sol#152-216) 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:
Function DlnBase.constructor() (contracts/DLN/DlnBase.sol#83-85) 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:
DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes).affiliateAmount (contracts/DLN/DlnSource.sol#159) is a local variable never initialized
DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes).affiliateBeneficiary (contracts/DLN/DlnSource.sol#160) 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#156-159) 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#156-159) has external calls inside a loop: (success) = to.call{value: value}(new bytes(0)) (contracts/DLN/DlnBase.sol#157)
DlnSource._claimUnlock(bytes32,address,uint256) (contracts/DLN/DlnSource.sol#408-458) has external calls inside a loop: (success,None) = orderState.affiliateBeneficiary.call{gas: 2300,value: orderState.affiliateAmount}(new bytes(0)) (con
tracts/DLN/DlnSource.sol#423)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Function DlnBase._executePermit(address,bytes) (contracts/DLN/DlnBase.sol#117-134) has a dubious typecast: address<=IERC20Permit
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO:Detectors:
DlnBase.getChainId() (contracts/DLN/DlnBase.sol#277-281) uses assembly
        - INLINE ASM (contracts/DLN/DlnBase.sol#278-280)
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#156-159):
        - (success) = to.call{value: value}(new bytes(0)) (contracts/DLN/DlnBase.sol#157)
Low level call in DlnSource._claimUnlock(bytes32,address,uint256) (contracts/DLN/DlnSource.sol#408-458):
        - (success,None) = orderState.affiliateBeneficiary.call{gas: 2300,value: orderState.affiliateAmount}(new bytes(0)) (contracts/DLN/DlnSource.sol#423)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function DlnBase.__DlnBase_init(IDeBridgeGate) (contracts/DLN/DlnBase.sol#87-93) is not in mixedCase
Parameter DlnBase.__DlnBase_init(IDeBridgeGate)._deBridgeGate (contracts/DLN/DlnBase.sol#87) is not in mixedCase
Function DlnBase.__DlnBase_init_unchained(IDeBridgeGate) (contracts/DLN/DlnBase.sol#95-101) is not in mixedCase
Parameter DlnBase.__DlnBase_init_unchained(IDeBridgeGate)._deBridgeGate (contracts/DLN/DlnBase.sol#95) is not in mixedCase
Parameter DlnBase.getOrderId(DlnOrderLib.Order)._order (contracts/DLN/DlnBase.sol#272) is not in mixedCase
Parameter DlnSource.initialize(IDeBridgeGate,uint88,uint16)._deBridgeGate (contracts/DLN/DlnSource.sol#136) is not in mixedCase
Parameter DlnSource.initialize(IDeBridgeGate,uint88,uint16)._globalFixedNativeFee (contracts/DLN/DlnSource.sol#137) is not in mixedCase
Parameter DlnSource.initialize(IDeBridgeGate,uint88,uint16)._globalTransferFeeBps (contracts/DLN/DlnSource.sol#138) is not in mixedCase
Parameter DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes)._orderCreation (contracts/DLN/DlnSource.sol#153) is not in mixedCase
Parameter DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes)._affiliateFee (contracts/DLN/DlnSource.sol#154) is not in mixedCase
Parameter DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes)._referralCode (contracts/DLN/DlnSource.sol#155) is not in mixedCase
Parameter DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes)._permitEnvelope (contracts/DLN/DlnSource.sol#156) is not in mixedCase
Parameter DlnSource.claimBatchUnlock(bytes32[],address)._orderIds (contracts/DLN/DlnSource.sol#223) is not in mixedCase
Parameter DlnSource.claimBatchUnlock(bytes32[],address)._beneficiary (contracts/DLN/DlnSource.sol#223) is not in mixedCase
Parameter DlnSource.claimUnlock(bytes32,address)._orderId (contracts/DLN/DlnSource.sol#240) is not in mixedCase
Parameter DlnSource.claimUnlock(bytes32,address)._beneficiary (contracts/DLN/DlnSource.sol#240) is not in mixedCase
Parameter DlnSource.claimBatchCancel(bytes32[],address)._orderIds (contracts/DLN/DlnSource.sol#254) is not in mixedCase
Parameter DlnSource.claimBatchCancel(bytes32[],address)._beneficiary (contracts/DLN/DlnSource.sol#254) is not in mixedCase
Parameter DlnSource.claimCancel(bytes32,address)._orderId (contracts/DLN/DlnSource.sol#271) is not in mixedCase
Parameter DlnSource.claimCancel(bytes32,address)._beneficiary (contracts/DLN/DlnSource.sol#271) is not in mixedCase

Parameter DlnSource.patchOrderGive(DlnOrderLib.Order,uint256,bytes)._order (contracts/DLN/DlnSource.sol#290) is not in mixedCase
Parameter DlnSource.patchOrderGive(DlnOrderLib.Order,uint256,bytes)._addGiveAmount (contracts/DLN/DlnSource.sol#291) is not in mixedCase
Parameter DlnSource.patchOrderGive(DlnOrderLib.Order,uint256,bytes)._permitEnvelope (contracts/DLN/DlnSource.sol#292) is not in mixedCase
Parameter DlnSource.setDlnDestinationAddress(uint256,bytes,DlnOrderLib.ChainEngine)._chainIdTo (contracts/DLN/DlnSource.sol#327) is not in mixedCase
Parameter DlnSource.setDlnDestinationAddress(uint256,bytes,DlnOrderLib.ChainEngine)._dlnDestinationAddress (contracts/DLN/DlnSource.sol#327) is not in mixedCase
Parameter DlnSource.setDlnDestinationAddress(uint256,bytes,DlnOrderLib.ChainEngine)._chainEngine (contracts/DLN/DlnSource.sol#327) is not in mixedCase
Parameter DlnSource.withdrawFee(address[],address)._tokens (contracts/DLN/DlnSource.sol#340) is not in mixedCase
Parameter DlnSource.withdrawFee(address[],address)._beneficiary (contracts/DLN/DlnSource.sol#340) is not in mixedCase
Parameter DlnSource.updateGlobalFee(uint88,uint16)._globalFixedNativeFee (contracts/DLN/DlnSource.sol#355) is not in mixedCase
Parameter DlnSource.updateGlobalFee(uint88,uint16)._globalTransferFeeBps (contracts/DLN/DlnSource.sol#356) is not in mixedCase
Parameter DlnSource.validateCreationOrder(DlnOrderLib.OrderCreation,address)._orderCreation (contracts/DLN/DlnSource.sol#367) is not in mixedCase
Parameter DlnSource.validateCreationOrder(DlnOrderLib.OrderCreation,address)._sender (contracts/DLN/DlnSource.sol#367) 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#327) is too similar to DlnBase.chainEngines (contracts/DLN/DlnBase.sol#43)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
Function DlnBase._executePermit(address,bytes) (contracts/DLN/DlnBase.sol#117-134) contains magic numbers: 32, 64
Function DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes) (contracts/DLN/DlnSource.sol#152-216) contains magic numbers: 52, 20
Function DlnSource._claimUnlock(bytes32,address,uint256) (contracts/DLN/DlnSource.sol#408-458) contains magic number: 2300
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
In a function DlnSource.createOrder(DlnOrderLib.OrderCreation,bytes,uint32,bytes) (contracts/DLN/DlnSource.sol#152-216) 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
```

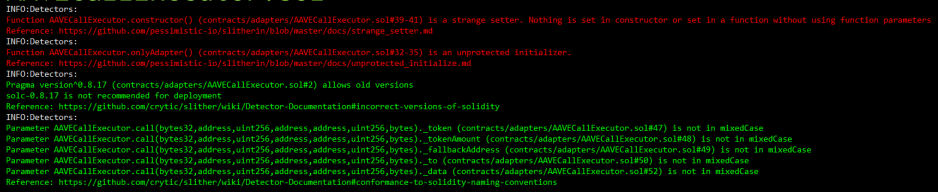
ExternalCallExecutor.sol

```
INFO:Detectors:
Function ExternalCallExecutor.constructor() (contracts/adapters/ExternalCallExecutor.sol#43-45) 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 ExternalCallExecutor.onlyAdapter() (contracts/adapters/ExternalCallExecutor.sol#36-39) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initialize.md
INFO:Detectors:
ExternalCallExecutor.onReceived(bytes32,address,bytes).success_scope_0 (contracts/adapters/ExternalCallExecutor.sol#73) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
Function Executor._execute(address,uint256,bytes,uint256) (contracts/base/Executor.sol#19-33) contains a low level call to a custom address
Function ExternalCallExecutor.onReceived(bytes32,address,bytes) (contracts/adapters/ExternalCallExecutor.sol#49-78) 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:
ExternalCallExecutor.onReceived(bytes32,address,bytes)._fallbackAddress (contracts/adapters/ExternalCallExecutor.sol#51) lacks a zero-check on :
                - (success) = _fallbackAddress.call{value: amount}(new bytes(0)) (contracts/adapters/ExternalCallExecutor.sol#62-64)
                - (success) = _fallbackAddress.call{value: amount}(new bytes(0)) (contracts/adapters/ExternalCallExecutor.sol#73-75)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Function ExternalCallExecutor._isValideData(bytes) (contracts/adapters/ExternalCallExecutor.sol#115-131) 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#151-164) uses assembly
        - INLINE ASM (contracts/adapters/ExternalCallExecutor.sol#159-161)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Pragma version^0.8.17 (contracts/adapters/ExternalCallExecutor.sol#2) allows old versions
Pragma version^0.8.7 (contracts/base/Executor.sol#2) 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 ExternalCallExecutor.onReceived(bytes32,address,bytes) (contracts/adapters/ExternalCallExecutor.sol#49-78):
        - (success) = _fallbackAddress.call{value: amount}(new bytes(0)) (contracts/adapters/ExternalCallExecutor.sol#62-64)
        - (success) = _fallbackAddress.call{value: amount}(new bytes(0)) (contracts/adapters/ExternalCallExecutor.sol#73-75)
Low level call in Executor._execute(address,uint256,bytes,uint256) (contracts/base/Executor.sol#19-33):
        - (success,None) = _to.call{gas: _txGas,value: _value}(_data) (contracts/base/Executor.sol#28)
        - (success,None) = _to.call{value: _value}(_data) (contracts/base/Executor.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter ExternalCallExecutor.onReceived(bytes32,address,bytes)._fallbackAddress (contracts/adapters/ExternalCallExecutor.sol#51) is not in mixedCase
Parameter ExternalCallExecutor.onReceived(bytes32,address,bytes)._payload (contracts/adapters/ExternalCallExecutor.sol#52) is not in mixedCase
Parameter ExternalCallExecutor.onReceived(bytes32,address,uint256,address,bytes)._token (contracts/adapters/ExternalCallExecutor.sol#82) is not in mixedCase
Parameter ExternalCallExecutor.onReceived(bytes32,address,uint256,address,bytes)._fallbackAddress (contracts/adapters/ExternalCallExecutor.sol#84) is not in mixedCase
Parameter ExternalCallExecutor.onReceived(bytes32,address,uint256,address,bytes)._payload (contracts/adapters/ExternalCallExecutor.sol#85) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Function ExternalCallExecutor._toBytes4(bytes,uint256) (contracts/adapters/ExternalCallExecutor.sol#151-164) contains magic number: 4
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
```

DlnExternalCallAdapter.sol

```
INFO:Detectors:
DlnExternalCallAdapter._safeTransferETH(address,uint256) (contracts/adapters/DlnExternalCallAdapter.sol#367-370) sends eth to arbitrary user
        Dangerous calls:
        - (success) = to.call{value: value}(new bytes(0)) (contracts/adapters/DlnExternalCallAdapter.sol#368)
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#84-86) 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#71-74) 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#234) 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#367-370) 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#88-96) should emit an event for:
        - dlnDestination = _dlnDestination (contracts/adapters/DlnExternalCallAdapter.sol#92)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control
INFO:Detectors:
DlnExternalCallAdapter.initialize(address,address)._dlnDestination (contracts/adapters/DlnExternalCallAdapter.sol#88) lacks a zero-check on :
                - dlnDestination = _dlnDestination (contracts/adapters/DlnExternalCallAdapter.sol#92)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Function DlnExternalCallAdapter._getBalance(address) (contracts/adapters/DlnExternalCallAdapter.sol#319-329) 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#367-370):
        - (success) = to.call{value: value}(new bytes(0)) (contracts/adapters/DlnExternalCallAdapter.sol#368)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter DlnExternalCallAdapter.initialize(address,address)._dlnDestination (contracts/adapters/DlnExternalCallAdapter.sol#88) is not in mixedCase
Parameter DlnExternalCallAdapter.initialize(address,address)._executor (contracts/adapters/DlnExternalCallAdapter.sol#88) is not in mixedCase
Parameter DlnExternalCallAdapter.receiveCall(bytes32,address,address,uint256,bytes,address)._orderId (contracts/adapters/DlnExternalCallAdapter.sol#110) is not in mixedCase
Parameter DlnExternalCallAdapter.receiveCall(bytes32,address,address,uint256,bytes,address)._callAuthority (contracts/adapters/DlnExternalCallAdapter.sol#111) is not in mixedCase
Parameter DlnExternalCallAdapter.receiveCall(bytes32,address,address,uint256,bytes,address)._tokenAddress (contracts/adapters/DlnExternalCallAdapter.sol#112) is not in mixedCase
Parameter DlnExternalCallAdapter.receiveCall(bytes32,address,address,uint256,bytes,address)._transferredAmount (contracts/adapters/DlnExternalCallAdapter.sol#113) is not in mixedCase
Parameter DlnExternalCallAdapter.receiveCall(bytes32,address,address,uint256,bytes,address)._externalCall (contracts/adapters/DlnExternalCallAdapter.sol#114) is not in mixedCase
Parameter DlnExternalCallAdapter.receiveCall(bytes32,address,address,uint256,bytes,address)._externalCallRewardBeneficiary (contracts/adapters/DlnExternalCallAdapter.sol#115) is not in mixedCase
Parameter DlnExternalCallAdapter.executeCall(bytes32,address,address,uint256,bytes,address)._orderId (contracts/adapters/DlnExternalCallAdapter.sol#162) is not in mixedCase
Parameter DlnExternalCallAdapter.executeCall(bytes32,address,address,uint256,bytes,address)._callAuthority (contracts/adapters/DlnExternalCallAdapter.sol#163) is not in mixedCase
Parameter DlnExternalCallAdapter.executeCall(bytes32,address,address,uint256,bytes,address)._tokenAddress (contracts/adapters/DlnExternalCallAdapter.sol#164) is not in mixedCase
Parameter DlnExternalCallAdapter.executeCall(bytes32,address,address,uint256,bytes,address)._tokenAmount (contracts/adapters/DlnExternalCallAdapter.sol#165) is not in mixedCase
Parameter DlnExternalCallAdapter.executeCall(bytes32,address,address,uint256,bytes,address)._externalCall (contracts/adapters/DlnExternalCallAdapter.sol#166) is not in mixedCase
Parameter DlnExternalCallAdapter.executeCall(bytes32,address,address,uint256,bytes,address)._rewardBeneficiary (contracts/adapters/DlnExternalCallAdapter.sol#167) is not in mixedCase
Parameter DlnExternalCallAdapter.cancelCall(bytes32,address,address,uint256,address,bytes)._orderId (contracts/adapters/DlnExternalCallAdapter.sol#190) is not in mixedCase
Parameter DlnExternalCallAdapter.cancelCall(bytes32,address,address,uint256,address,bytes)._callAuthority (contracts/adapters/DlnExternalCallAdapter.sol#191) is not in mixedCase
Parameter DlnExternalCallAdapter.cancelCall(bytes32,address,address,uint256,address,bytes)._tokenAddress (contracts/adapters/DlnExternalCallAdapter.sol#192) is not in mixedCase
Parameter DlnExternalCallAdapter.cancelCall(bytes32,address,address,uint256,address,bytes)._tokenAmount (contracts/adapters/DlnExternalCallAdapter.sol#193) is not in mixedCase
Parameter DlnExternalCallAdapter.cancelCall(bytes32,address,address,uint256,address,bytes)._recipient (contracts/adapters/DlnExternalCallAdapter.sol#194) is not in mixedCase
Parameter DlnExternalCallAdapter.cancelCall(bytes32,address,address,uint256,address,bytes)._externalCall (contracts/adapters/DlnExternalCallAdapter.sol#195) is not in mixedCase
Parameter DlnExternalCallAdapter.updateExecutor(address)._newExecutor (contracts/adapters/DlnExternalCallAdapter.sol#220) is not in mixedCase
Parameter DlnExternalCallAdapter.getCallId(bytes32,address,address,uint256,bytes)._orderId (contracts/adapters/DlnExternalCallAdapter.sol#344) is not in mixedCase
Parameter DlnExternalCallAdapter.getCallId(bytes32,address,address,uint256,bytes)._callAuthority (contracts/adapters/DlnExternalCallAdapter.sol#345) is not in mixedCase
Parameter DlnExternalCallAdapter.getCallId(bytes32,address,address,uint256,bytes)._tokenAddress (contracts/adapters/DlnExternalCallAdapter.sol#346) is not in mixedCase
Parameter DlnExternalCallAdapter.getCallId(bytes32,address,address,uint256,bytes)._transferredAmount (contracts/adapters/DlnExternalCallAdapter.sol#347) is not in mixedCase
Parameter DlnExternalCallAdapter.getCallId(bytes32,address,address,uint256,bytes)._externalCall (contracts/adapters/DlnExternalCallAdapter.sol#348) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

AAVECallExecutor.sol



```
INFO:Detectors:
Function AAVECallExecutor.constructor() (contracts/adapters/AAVECallExecutor.sol#39-41) 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 AAVECallExecutor.onlyAdapter() (contracts/adapters/AAVECallExecutor.sol#32-35) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initialize.md
INFO:Detectors:
Pragma version^0.8.17 (contracts/adapters/AAVECallExecutor.sol#2) 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:
Parameter AAVECallExecutor.call(bytes32,address,uint256,address,address,uint256,bytes)._token (contracts/adapters/AAVECallExecutor.sol#47) is not in mixedCase
Parameter AAVECallExecutor.call(bytes32,address,uint256,address,address,uint256,bytes)._tokenAmount (contracts/adapters/AAVECallExecutor.sol#48) is not in mixedCase
Parameter AAVECallExecutor.call(bytes32,address,uint256,address,address,uint256,bytes)._fallbackAddress (contracts/adapters/AAVECallExecutor.sol#49) is not in mixedCase
Parameter AAVECallExecutor.call(bytes32,address,uint256,address,address,uint256,bytes)._to (contracts/adapters/AAVECallExecutor.sol#50) is not in mixedCase
Parameter AAVECallExecutor.call(bytes32,address,uint256,address,address,uint256,bytes)._data (contracts/adapters/AAVECallExecutor.sol#52) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

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

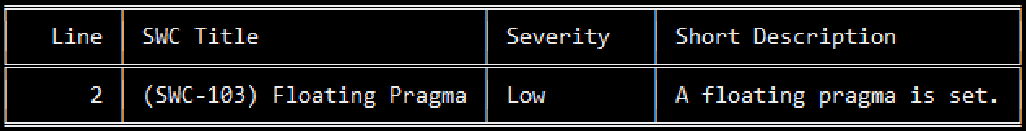
7.2 AUTOMATED SECURITY SCAN

Description:

Halborn used automated security scanners to assist with detection of well-known security issues and to identify low-hanging fruits on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on the smart contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

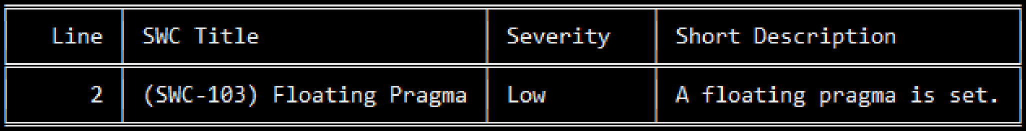
MythX results:

DlnDestination.sol



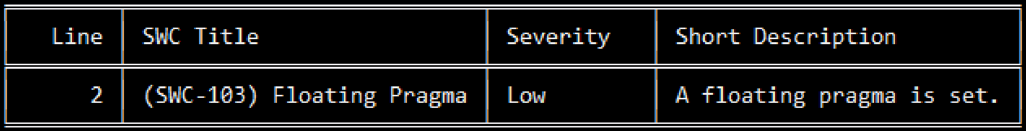
Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

DlnSource.sol



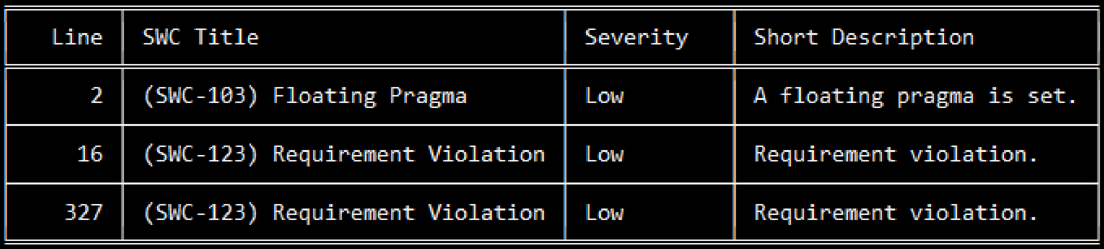
Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

ExternalCallExecutor.sol



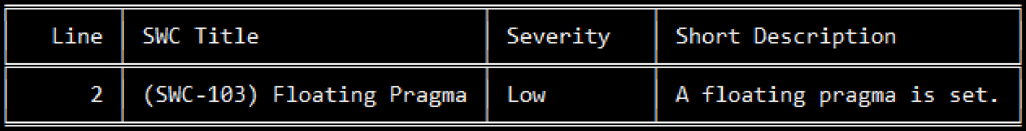
Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

DlnExternalCallAdapter.sol



Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
16	(SWC-123) Requirement Violation	Low	Requirement violation.
327	(SWC-123) Requirement Violation	Low	Requirement violation.

AAVECallExecutor.sol



Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

- No major issues were found by MythX.

THANK YOU FOR CHOOSING

HALBORN