// HALBORN

deBridge

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **August 25th, 2021 - September 15th, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY 6
CONTACTS 6
1 EXECUTIVE OVERVIEW 7
1.1 Introduction 8
1.2 Audit Summary 8
1.3 Test Approach & Methodology 9
RISK METHODOLOGY 9
1.4 Scope 11
2 Assessment Summary & Findings Overview 12
3 FINDINGS & TECH DETAILS 13
3.1 (HAL-01) IMPROPER ROLE BASED ACCESS CONTROL POLICY - MEDIUM 15
Description 15
Code Location 15
Risk Level 16
Recommendation 16
Remediation plan 19
3.2 (HAL-02) IMPROPER HANDLING OF ARGUMENTS - MEDIUM 21
Description 21
Code Location 21
Risk Level 22
Recommendation 22
Remediation plan 22
3.3 (HAL-03) MISSING RE-ENTRANCY PROTECTION - MEDIUM 23
Description 23

Code Location 23
Risk Level 24
Recommendation 24
Remediation plan 24
3.4 (HAL-04) OUTDATED DEPENDENCIES - LOW 25
Description 25
Code Location 25
Risk Level 25
Recommendation 25
References 26
Remediation plan 26
3.5 (HAL-05) IMPROPER APPLICATION OF PRINCIPLE OF LEAST PRIVILEGE - LOW 27
Description 27
Code Location 27
Risk Level 27
Recommendation 27
Remediation plan 27
3.6 (HAL-06) FOR LOOP OVER DYNAMIC DATA STRUCTURE - LOW 28
Description 28
Code Location 28
Risk Level 30
Recommendation 30
Remediation plan 30
3.7 (HAL-07) USE OF BLOCK.TIMESTAMP - LOW 32
Description 32

Code Location 32
Risk Level 32
Recommendation 33
Remediation plan 33
3.8 (HAL-08) IGNORED RETURN VALUES - LOW 34
Description 34
Code Location 34
Risk Level 35
Recommendation 36
Remediation plan 36
3.9 (HAL-09) MISSING ZERO VALUE CHECK - LOW 37
Description 37
Code Location 37
Risk Level 38
Recommendation 38
Remediation plan 38
3.10 (HAL-10) MISSING ZERO ADDRESS CHECK - LOW 39
Description 39
Code Location 39
Risk Level 41
Recommendation 41
Remediation plan 41
3.11 (HAL-11) USE OF INLINE ASSEMBLY - INFORMATIONAL 42
Description 42
Code Location 42
Risk Level 42

Recommendation 42
Remediation plan 43
3.12 (HAL-12) MISSING EVENT EMITTING - INFORMATIONAL 44
Description 44
Code Location 44
Risk Level 45
Recommendation 45
Remediation plan 45
3.13 (HAL-13) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL 46
Description 46
Code Location 46
Risk Level 46
Recommendation 47
Remediation 47
3.14 (HAL-14) OUTDATED DOCUMENTATION - INFORMATIONAL 48
Description 48
Code Location 48
Risk Level 48
Recommendation 48
3.15 (HAL-15) INCONSISTENT CODE - INFORMATIONAL 49
Description 49
Code Location 49
Risk Level 49
Recommendation 49
Remediation 50

3.16 (HAL-16) TYPO IN CODE - INFORMATIONAL 51
Description 51
Code Location 51
Risk Level 52
Recommendation 52
Remediation 52
4 AUTOMATED TESTING 52
4.1 STATIC ANALYSIS REPORT 54
Description 54
Results 54
4.2 AUTOMATED SECURITY SCAN 58
Description 58
Results 58

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	08/25/2021	Timur Guvenkaya
0.9	Document Edits	09/15/2021	Timur Guvenkaya
1.0	Final Draft	09/15/2021	Gabi Urrutia
1.1	Remediation Plan	09/29/2021	Timur Guvenkaya
1.1	Remediation Plan Review	09/29/2021	Gabi Urrutia
1.1	Remediation Plan Edit	09/30/2021	Timur Guvenkaya
1.1	Final Remediation Plan	09/30/2021	Timur Guvenkaya

CONTACTS

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

EXECUTIVE OVERVIEW

1.1 Introduction

deBridge engaged Halborn to conduct a security assessment on their smart contracts beginning on August 25th, 2021 and ending September 15th, 2021. deBridge a cross-chain interoperability and liquidity transfer protocol that allows truly decentralized transfer of assets between various blockchains. The cross-chain intercommunication of deBridge smart contracts is powered by the network of independent oracles/validators which are elected by deBridge governance.

Though this security audit's outcome is satisfactory, only the most essential aspects were tested and verified to achieve objectives and deliverables set in the scope due to time and resource constraints. It is essential to note the use of the best practices for secure development.

1.2 Audit Summary

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

The purpose of this audit is to achieve the following:

- Identify potential security issues with the smart contracts.
- Ensure that smart contract deployment scripts function as intended.

In summary, Halborn identified few security risks that were either acknowledged or solved by 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 the smart contract audit. While manual testing is recommended to uncover flaws in logic, process,and implementation; automated testing techniques help enhance coverage of the smart contract code and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

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

RISK METHODOLOGY:

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

RISK SCALE - LIKELIHOOD

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

RISK SCALE - IMPACT

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

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

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 Scope

The review was scoped to contracts and deployment scripts in the audit branch in debridge-contracts-v1 repository.

Smart contracts:

- DeBridgeGate.sol
- AggregatorBase.sol
- ConfirmationAggregator.sol
- SignatureAggregator.sol
- SignatureVerifier.sol
- CallProxy.sol
- FeeProxy.sol
- WrappedAsset.sol
- SignatureUtil.sol

Deployment Scripts:

- 00_external.js
- 01_DBR.js
- 02_SignatureAggregator.js
- 03_SignatureVerifier.js
- 04_ConfirmationAggregator.js
- 05_CallProxy.js
- 06_FeeProxy.js
- 07_DefiController.js
- 08_DeBridgeGate.js

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	3	7	6

LIKELIHOOD

IMPACT

(HAL-01)				
(HAL-04)	(HAL-03)			
	(HAL-05) (HAL-06) (HAL-07)			
(HAL-11)	(HAL-10)	(HAL-08) (HAL-09)		(HAL-02)
(HAL-13) (HAL-14) (HAL-15) (HAL-16)	(HAL-12)			

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
IMPROPER ROLE BASED ACCESS CONTROL POLICY	Medium	SOLVED - 09/29/2021
IMPROPER HANDLING OF ARGUMENTS	Medium	SOLVED - 09/29/2021
MISSING RE-ENTRANCY PROTECTION	Medium	SOLVED - 09/29/2021
OUTDATED DEPENDENCIES	Low	SOLVED - 09/29/2021
IMPROPER APPLICATION OF PRINCIPLE OF LEAST PRIVILEGES	Low	SOLVED - 09/29/2021
FOR LOOP OVER DYNAMIC DATA STRUCTURE	Low	ACKNOWLEDGED
USE OF BLOCK.TIMESTAMP	Low	NOT APPLICABLE
IGNORED RETURN VALUES	Low	SOLVED - 09/29/2021
MISSING ZERO VALUE CHECK	Low	SOLVED - 09/29/2021
MISSING ZERO ADDRESS CHECK	Low	PARTIALLY SOLVED - 09/29/2021
USE OF INLINE ASSEMBLY	Informational	ACKNOWLEDGED
MISSING EVENT EMITTING	Informational	ACKNOWLEDGED
POSSIBLE MISUSE OF PUBLIC FUNCTIONS	Informational	SOLVED - 09/29/2021
OUTDATED DOCUMENTATION	Informational	ACKNOWLEDGED
INCONSISTENT CODE	Informational	SOLVED - 09/29/2021
TYPO IN CODE	Informational	SOLVED - 09/29/2021

FINDINGS & TECH DETAILS

3.1 (HAL-01) IMPROPER ROLE BASED ACCESS CONTROL POLICY - MEDIUM

Description:

In smart contracts, implementing a correct Access Control policy is an essential step to maintain security and decentralization for permissions on a token. All the features of the smart contract , such as mint/burn tokens and pause contracts are given by Access Control. For instance, Ownership is the most common form of Access Control. In other words, the owner of a contract (the account that deployed it by default) can do some administrative tasks on it. Nevertheless,other authorization levels are required to follow the principle of least privilege, also known as least authority. Briefly, any process, user or program only can access to the necessary resources or information. Otherwise, the ownership role is useful in a simple system, but more complex projects require the use of more roles by using Role-based access control.

Therefore, there could be multiple roles such as manager, minter, admin, or pauser in contracts which use a proxy contract. In the **deBridge** the role based access control is implemented. However, Admin user is able to set himself/herself as any of those roles. This can lead to accumulation of too many privileges in one user. So, if the private key of the admin account is compromised and multi-signature was not implemented, the attacker can perform many actions such as transferring ownership or destruct the contract by taking advantage of the lack of the principle of least privilege.

Code Location:

Functions with role setting functionality that should not allow admin to have those roles:

AggregatorBase.sol

- updateOracleAdminByOwner

- updateOracleAdmin
- addOracles

ConfirmationAggregator.sol

- setWrappedAssetAdmin

SignatureVerifier.sol

- setWrappedAssetAdmin

WrappedAsset.sol

- constructor

AccessControl.sol

- grantRole

AccessControlUpgradeable.sol

- grantRole

Risk Level:

Likelihood - 1
Impact - 5

Recommendation:

Add a conditional check that won’t let an admin to have any other roles apart from the admin role. Possible solution is to:

- Create a modifier in AggregatorBase.sol that will check whether address passed to function responsible for setting admin roles does not have DEFAULT_ADMIN_ROLE assigned to it.

```
51      modifier onlyNonAdminAddress(address _account⬆) {
52          if(hasRole(DEFAULT_ADMIN_ROLE, _account⬆)) revert AdminAddressDenied();
53          _;
54      }
55
```

- Apply modifier to relevant functions and in case of function accepting an array of addresses - add conditional check directly

AggregatorBase.sol

- updateOracleAdminByOwner

```
143     /// @dev Update oracle admin.
144     /// @param _oracle Oracle address.
145     /// @param _admin new admin address.
        ftrace | funcSig
146     function updateOracleAdminByOwner(address _oracle⬆, address _admin⬆) external onlyAdmin onlyNonAdminAddress(_admin) {
147
148         OracleInfo storage oracleInfo = getOracleInfo[_oracle⬆];
149         if (!oracleInfo.exist) revert OracleNotFound();
150         oracleInfo.admin = _admin⬆;
151         emit UpdateOracleAdminByOwner(_oracle⬆, _admin⬆);
152     }
```

- updateOracleAdmin

```
67      /* ========== ORACLES ========== */
68
69      /// @dev Updates oracle's admin address.
70      /// @param _oracle Oracle address.
71      /// @param _newOracleAdmin New oracle address.
        ftrace | funcSig
72      function updateOracleAdmin(address _oracle⬆, address _newOracleAdmin⬆) external onlyNonAdminAddress(_newOracleAdmin){
73          if (getOracleInfo[_oracle⬆].admin ≠ msg.sender) revert OraclesAdminAccessDenied();
74          getOracleInfo[_oracle⬆].admin = _newOracleAdmin⬆;
75          emit UpdateOracleAdmin(_oracle⬆, _newOracleAdmin⬆);
76      }
77
```

- addOracles

```
ftrace | funcSig
function addOracles(
    address[] memory _oracles,
    address[] memory _admins,
    bool[] memory _required
) external onlyAdmin {
    for (uint256 i = 0; i < _oracles.length; i++) {
        if(hasRole(DEFAULT_ADMIN_ROLE, _admins[i])) revert AdminAddressDenied();
        You, a minute ago • Uncommitted changes
        OracleInfo storage oracleInfo = getOracleInfo[_oracles[i]];
        if (oracleInfo.exist) revert OracleAlreadyExist();

        oracleAddresses.push(_oracles[i]);

        if (_required[i]) {
            requiredOraclesCount += 1;
        }
```

ConfirmationAggregator.sol

- setWrappedAssetAdmin

```
/// @dev Set admin for any deployed wrapped asset.
/// @param _wrappedAssetAdmin Admin address.
ftrace | funcSig
function setWrappedAssetAdmin(address _wrappedAssetAdmin) public onlyAdmin onlyNonAdminAddress(_wrappedAssetAdmin) {

    wrappedAssetAdmin = _wrappedAssetAdmin;
}
```

SignatureVerifier.sol

- setWrappedAssetAdmin

```
/// @dev Set admin for any deployed wrapped asset.
/// @param _wrappedAssetAdmin Admin address.
ftrace | funcSig
function setWrappedAssetAdmin(address _wrappedAssetAdmin) public onlyAdmin onlyNonAdminAddress(_wrappedAssetAdmin){
    wrappedAssetAdmin = _wrappedAssetAdmin;
}
```

WrappedAsset.sol

- constructor

```
ftrace
37  constructor(
38      string memory _name⇑,
39      string memory _symbol⇑,
40      uint8 _tokenDecimals⇑,
41      address _admin⇑,
42      address[] memory _minters⇑
43  ) ERC20(_name, _symbol) {
44      _decimals = _tokenDecimals⇑;
45      _setupRole(DEFAULT_ADMIN_ROLE, _admin⇑);
46      for (uint256 i = 0; i < _minters⇑.length; i++) {
47          if(_admin⇑ == _minters⇑[i]) revert AdminAddressDenied();
48      You, 4 minutes ago • Uncommitted changes
49          _setupRole(MINTER_ROLE, _minters⇑[i]);
50
51
```

Also, to avoid Admin to grant any roles to himself through grantRole public function in node_modules/@openzeppelin/contracts/access/AccessControl.sol and node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol:

- Create additional modifier that will check that the address passed to grantRole does not belong to the role's admin address.

```
72
73      modifier onlyNonAdminAddress(bytes32 role⇑, address account⇑) {
74          require(!hasRole(getRoleAdmin(role⇑), account⇑), "Cannot set admin address");
75          _;
76      }
```

- Add modifier to grantRole function.

```
ftrace | funcSig
140 function grantRole(bytes32 role⇑, address account⇑) public virtual override onlyRole(getRoleAdmin(role)) onlyNonAdminAddress(role, account) {
141     _grantRole(role⇑, account⇑);
142
143 }
```

Please note that in order for changes to persist in case of library updates, it is advised to override grantRole and use it within contracts when needed.

Remediation plan:

SOLVED: deBridge team will use a multi-signature wallet for the deployment. Additional changes include:

- In the `addOracles` function, `_admins` argument was removed.
- `updateOracleAdminByOwner` function was removed.
- `updateOracleAdmin` function was removed.
- `setWrappedAssetAdmin` function was removed. Token deployments are now happening through new `DeTokenDeployer.sol` contract.
- `WrappedAsset.sol` was renamed to `DeBridgeToken.sol`.

3.2 (HAL-02) IMPROPER HANDLING OF ARGUMENTS - MEDIUM

Description:

Incorrect arguments passed from public to internal function in good case scenario can just revert the transaction but in worst case scenario can silently lead to unexpected behaviour of the system.

Code Location:

In DeBridgeGate.sol contract, public function autoBurn is passing the _reservedFlag as a last argument to an internal _burn function. Unfortunately, the _burn function expects the _referralCode to be the last argument in the function.

DeBridgeGate.sol: Line#414

```
400     function autoBurn(
401         bytes32 _debridgeId,
402         bytes memory _receiver,
403         uint256 _amount,
404         uint256 _chainIdTo,
405         bytes memory _fallbackAddress,
406         uint256 _executionFee,
407         bytes memory _data,
408         bytes memory _permit,
409         bool _useAssetFee,
410         uint8 _reservedFlag,
411         uint32 _referralCode
412     ) external payable override nonReentrant whenNotPaused {
413         // the amount will be reduced by the protocol fee
414         _amount = _burn(_debridgeId, _amount, _chainIdTo, _permit, _useAssetFee, _reservedFlag);
```

DeBridgeGate.sol /debridge-contracts-v1/contracts/transfers - Definitions (

ftrace | funcSig

```
867     function _burn(
868         bytes32 _debridgeId,
869         uint256 _amount,
870         uint256 _chainIdTo,
871         bytes memory _permit,
872         bool _useAssetFee,
873         uint32 _referralCode
874     ) internal returns (uint256) {
875         DebridgeInfo storage debridge = getDebridge[_debridgeId];
```

Risk Level:

Likelihood - 5
Impact - 2

Recommendation:

Replace the _reservedFlag argument with _referralCode as a last argument to the _burn function in the autoBurn public function.

Remediation plan:

SOLVED: deBridge team solved the issue by removing autoBurn completely by merging it with burn. The changes include:
- send function was merged with autoSend function.
- burn function was merged with autoBurn function.
- claim function was merged with autoClaim function.
- Then, claim function was merged with mint function.
- Then, send function was merged with burn function.

At the end, the code was refactored to only two functions:
- claim
- send

3.3 (HAL-03) MISSING RE-ENTRANCY PROTECTION - MEDIUM

Description:

To protect against cross-function reentrancy attacks, it may be necessary to use a mutex. By using this lock, an attacker can no longer exploit the function with a recursive call. OpenZeppelin has it's own mutex implementation called ReentrancyGuard which provides a modifier to any function called "nonReentrant" that guards the function with a mutex against the Reentrancy attacks.

Code Location:

In DeBridgeGate.sol contract, functions returnReserves and requestReserves are missing nonReentrant guard. Also, in both of these functions, external calls are called before all state changes are resolved, making it vulnerable to a Reentrancy attack.

DeBridgeGate.sol: Line#646

```
ftrace | funcSig
646     function requestReserves(address _tokenAddress, uint256 _amount)
647         external      artyukh, 3 weeks ago • prettier formatting code
648         override
649         onlyDefiController
650     {
651         bytes32 debridgeId = getDebridgeId(getChainId(), _tokenAddress);
652         DebridgeInfo storage debridge = getDebridge[debridgeId];
653         if (!debridge.exist) revert DebridgeNotFound();
654         uint256 minReserves = (debridge.balance * debridge.minReservesBps) / BPS_DENOMINATOR;
655
656         if (minReserves + _amount > IERC20(_tokenAddress).balanceOf(address(this)))
657             revert NotEnoughReserves();
658
659         IERC20(_tokenAddress).safeTransfer(defiController, _amount);
660         debridge.lockedInStrategies += _amount;      idealatom, a month ago • Fix typo in var
661     }
```

DeBridgeGate.sol: Line#666

```
666     function returnReserves(address _tokenAddress, uint256 _amount)
667         external      idealatom, a month ago • trim trailing whitespaces
668         override
669         onlyDefiController
670     {
671         bytes32 debridgeId = getDebridgeId(getChainId(), _tokenAddress);
672         DebridgeInfo storage debridge = getDebridge[debridgeId];
673         if (!debridge.exist) revert DebridgeNotFound();
674         IERC20(debridge.tokenAddress).safeTransferFrom(
675             defiController,      idealatom, 2 weeks ago • remove DefiController interface fro
676             address(this),
677             _amount
678         );
679         debridge.lockedInStrategies -= _amount;
680     }
```

Risk Level:

Likelihood - 2
Impact - 4

Recommendation:

Change the code to follow the checks-effects-interactions pattern and use ReentrancyGuard through the nonReentrant modifier.

Remediation plan:

SOLVED: deBridge team solved the issue by adding nonReentrant modifier and implementing the checks-effects-interactions pattern.

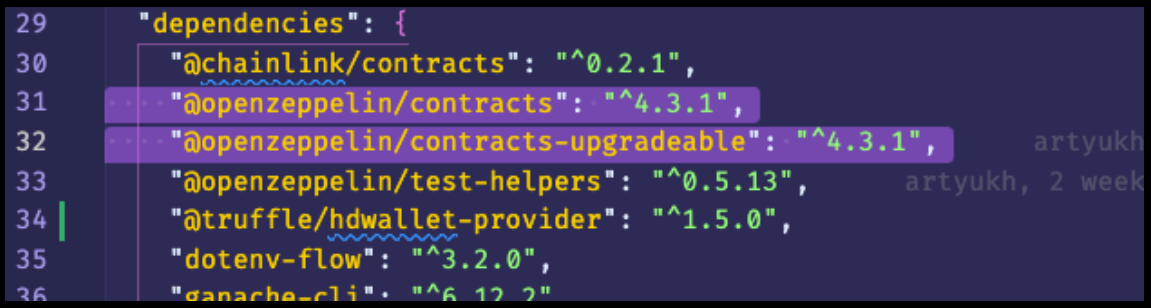
3.4 (HAL-04) OUTDATED DEPENDENCIES - LOW

Description:

It was noticed that the 4.3.1 version of openzepplin-contracts and openzepplin-contracts-upgradable is used in smart contracts. However, the latest version of those libraries is 4.3.2, which fixes a vulnerability in UUPSUpgradeable.

Code Location:

package.json: Line#31, Line#32



Risk Level:

Likelihood - 1
Impact - 4

Recommendation:

Even though UUPSUpgradeable is not used directly within contracts, it is always important to keep all libraries up-to-date.

References:

Open Zeppelin Advisory
UUPS Implementation Workaround

Remediation plan:

SOLVED: deBridge team solved the issue by updating dependencies to the latest version.

3.5 (HAL-05) IMPROPER APPLICATION OF PRINCIPLE OF LEAST PRIVILEGE - LOW

Description:

In 01_DBR.js deployment script admin is also set as minter. Upon deployment, admin will also have a minter role which violates the principle of least privilege.

Code Location:

```
17  if (deployInitParams.deploy.DBR) {
18    await deploy("WrappedAsset", {
19      from: deployer,
20      args: [
21        "Debridge token",
22        "DBR",
23        18,
24        deployer,
25        [deployer],
26      ],      idealatom, 2 weeks ago • add hardhat-deploy migrations
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

Add another address as minter to ensure that the system follows the principle of least privilege.

Remediation plan:

SOLVED: deBridge team solved by removing admin as minter in the deployment script.

3.6 (HAL-06) FOR LOOP OVER DYNAMIC DATA STRUCTURE - LOW

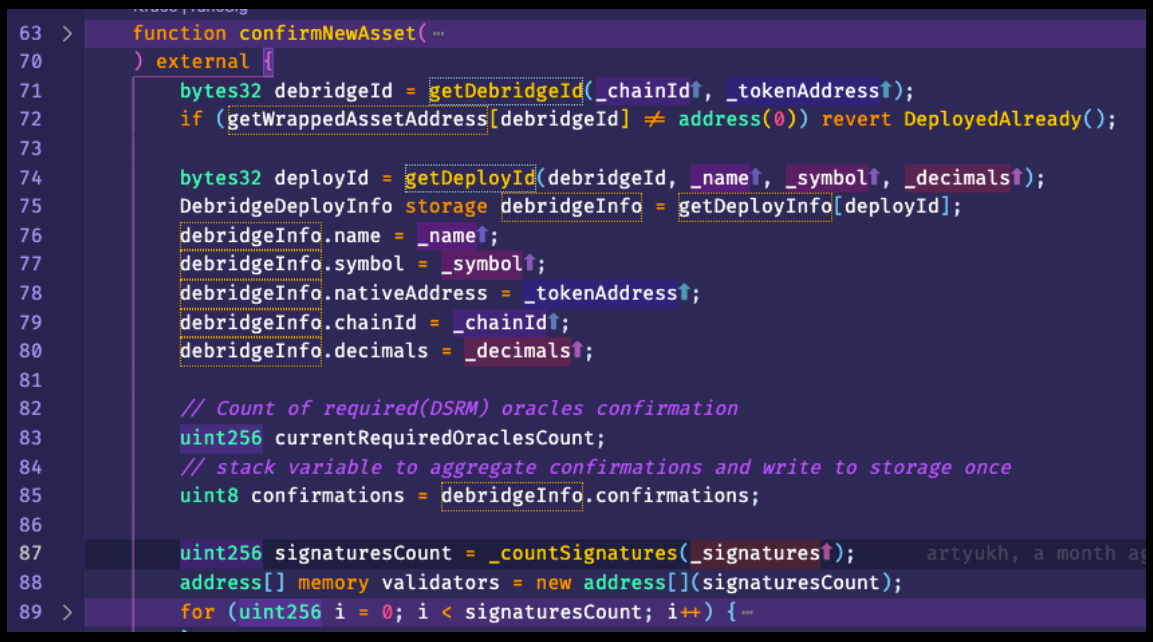
Description:

Calls inside a loop might lead to a denial-of-service attack. There are instances in SignatureVerifier.sol, AggregatorBase.sol, WrappedAsset.sol , and DebridgeGate.sol, where actions are performed within loop over a data structure with dynamic size.

Code Location:

SignatureVerifier.sol: Line#63, Line#116

- confirmNewAsset



```
63  function confirmNewAsset(…
70  ) external {
71      bytes32 debridgeId = getDebridgeId(_chainId, _tokenAddress);
72      if (getWrappedAssetAddress[debridgeId] != address(0)) revert DeployedAlready();
73
74      bytes32 deployId = getDeployId(debridgeId, _name, _symbol, _decimals);
75      DebridgeDeployInfo storage debridgeInfo = getDeployInfo[deployId];
76      debridgeInfo.name = _name;
77      debridgeInfo.symbol = _symbol;
78      debridgeInfo.nativeAddress = _tokenAddress;
79      debridgeInfo.chainId = _chainId;
80      debridgeInfo.decimals = _decimals;
81
82      // Count of required(DSRM) oracles confirmation
83      uint256 currentRequiredOraclesCount;
84      // stack variable to aggregate confirmations and write to storage once
85      uint8 confirmations = debridgeInfo.confirmations;
86
87      uint256 signaturesCount = _countSignatures(_signatures);      artyukh, a month ag
88      address[] memory validators = new address[](signaturesCount);
89      for (uint256 i = 0; i < signaturesCount; i++) {…
```

- submit

```
ftrace | funcSig
116 >   function submit( ⋯
120     ) external override onlyDeBridgeGate {
121         //Need confirmation to confirm submission
122 >       uint8 needConfirmations = _excessConfirmations↑ > minConfirmations ⋯
125         // Count of required(DSRM) oracles confirmation
126         uint256 currentRequiredOraclesCount;
127         // stack variable to aggregate confirmations and write to storage once
128         uint8 confirmations;
129         uint256 signaturesCount = _countSignatures(_signatures↑);
130         address[] memory validators = new address[](signaturesCount);
131 >       for (uint256 i = 0; i < signaturesCount; i++) {⋯
```

AggregatorBase.sol: Line#80

- addOracles

```
80      function addOracles(
81          address[] memory _oracles↑,
82          address[] memory _admins↑,
83          bool[] memory _required↑
84      ) external onlyAdmin {
85          for (uint256 i = 0; i < _oracles↑.length; i++) {
86              OracleInfo storage oracleInfo = getOracleInfo[_oracles↑[i]];      artyukh, a
87              if (oracleInfo.exist) revert OracleAlreadyExist();
```

WrappedAsset.sol: Line#35

- constructor

```
26      constructor(
27          string memory _name↑,
28          string memory _symbol↑,
29          uint8 _tokenDecimals↑,
30          address _admin↑,
31          address[] memory _minters↑
32      ) ERC20(_name, _symbol) {
33          _decimals = _tokenDecimals↑;
34          _setupRole(DEFAULT_ADMIN_ROLE, _admin↑);
35          for (uint256 i = 0; i < _minters↑.length; i++) {
36              _setupRole(MINTER_ROLE, _minters↑[i]);      kstasi, 4 mo
37          }
```

DebridgeGate.sol: Line#537, Line#553

- updateChainSupport

```
533     function updateChainSupport(
534         uint256[] memory _supportedChainIds,
535         ChainSupportInfo[] memory _chainSupportInfo
536     ) external onlyAdmin {
537         for (uint256 i = 0; i < _supportedChainIds.length; i++) {
538             getChainSupport[_supportedChainIds[i]] = _chainSupportInfo[i];
```

- updateAssetFixedFees

```
547     function updateAssetFixedFees(
548         bytes32 _debridgeId,
549         uint256[] memory _supportedChainIds,
550         uint256[] memory _assetFeesInfo
551     ) external onlyAdmin {
552         DebridgeFeeInfo storage debridgeFee = getDebridgeFeeInfo[_debridgeId];
553         for (uint256 i = 0; i < _supportedChainIds.length; i++) {
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

Consider limiting number of iterations by specifying max allowed length of supplied data. Also, whenever possible, use pull over push strategy for external calls.

Remediation plan:

ACKNOWLEDGED: deBridge team decided to acknowledge the issue because they claim that likelihood is low. Additional changes include:

- In the `addOracles` function `_admins` argument was removed.
- `WrappedAsset.sol` was renamed to `DeBridgeToken.sol`.

3.7 (HAL-07) USE OF BLOCK.TIMESTAMP - LOW

Description:

During a manual static review, the tester noticed the use of `block.timestamp` in `WrappedAsset.sol` contract. `block.timestamp` can be influenced by miners to a certain degree, so developers should be aware that this may have some risk if miners collude on time manipulation to influence the price oracles.

Code Location:

WrappedAsset.sol: Line#88

```
ftrace | funcSig
function permit(
    address _owner,
    address _spender,
    uint256 _value,
    uint256 _deadline,
    uint8 _v,
    bytes32 _r,
    bytes32 _s
) external override {
    require(_deadline >= block.timestamp, "permit: EXPIRED");
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

Use block.number instead of block.timestamp or now to reduce the risk of Maximal Extractable Value (MEV) attacks. Check if the timescale of the project occurs across years, days and months rather than seconds. If possible, it is recommended to use Oracles.

Remediation plan:

NOT APPLICABLE: deBridge team considers the usage of block.timestamp safe because it is directly derived from draft-ERC20Permit.sol. Also WrappedAsset.sol was renamed to DeBridgeToken.sol.

3.8 (HAL-08) IGNORED RETURN VALUES - LOW

Description:

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

Code Location:

DeBridgeGate.sol: Line#635, Line#892

- withdrawFee

```
624     /// @dev Withdraw fees.
625     /// @param _debridgeId Asset identifier.
        ftrace | funcSig
626     function withdrawFee(bytes32 _debridgeId) external override nonReentrant onlyFeeProxy {
627         DebridgeFeeInfo storage debridgeFee = getDebridgeFeeInfo[_debridgeId];
628         // Amount for transfer to treasure
629         uint256 amount = debridgeFee.collectedFees - debridgeFee.withdrawnFees;
630         debridgeFee.withdrawnFees += amount;
631
632         if (amount == 0) revert NotEnoughReserves();
633
634         if (_debridgeId == getDebridgeId(getChainId(), address(0))) {
635             payable(feeProxy).transfer(amount);
```

- _burn

```
872     function _burn(
873         bytes32 _debridgeId,
874         uint256 _amount,
875         uint256 _chainIdTo,
876         bytes memory _permit,
877         bool _useAssetFee,
878         uint32 _referralCode
879     ) internal returns (uint256) {
880         DebridgeInfo storage debridge = getDebridge[_debridgeId];
881         if (!debridge.exist) revert DebridgeNotFound();
882         if (debridge.chainId == getChainId()) revert WrongChain();
883         if (!getChainSupport[_chainIdTo].isSupported) revert WrongTargedChain();
884         if (_amount > debridge.maxAmount) revert TransferAmountTooHigh();
885         IWrappedAsset wrappedAsset = IWrappedAsset(debridge.tokenAddress);
886         if (_permit.length > 0) {
887             //First dealine, next is signature
888             uint256 deadline = _permit.toUint256(0);
889             (bytes32 r, bytes32 s, uint8 v) = _permit.parseSignature(32);
890             wrappedAsset.permit(msg.sender, address(this), _amount, deadline, v, r, s);
891         }
892         wrappedAsset.transferFrom(msg.sender, address(this), _amount);
```

CallProxy.sol: Line#49

- call

```
35     function call(
36         address _reserveAddress,
37         address _receiver,
38         bytes memory _data,
39         uint8 _reservedFlag,
40         bytes memory _nativeSender
41     ) external payable override onlyGateRole  returns (bool _result) {
42         // Add last argument is sender from original network
43         if (_reservedFlag == PROXY_WITH_SENDER_FLAG) {
44             _data = abi.encodePacked(_data, _nativeSender);
45         }
46
47         _result = externalCall(_receiver, msg.value, _data.length, _data);
48         if (!_result) {
49             payable(_reserveAddress).transfer(msg.value);
50         }       kstasi, 4 months ago • autoclaim with reserve address
51     }
52
```

Risk Level:

Likelihood - 3

Impact - 2

Recommendation:

Add return value check to avoid unexpected crash of the contract. Return value check will help in handling the exceptions better way.

Remediation plan:

SOLVED: deBridge team solved the issue by implementing _safeTransferETH function.

3.9 (HAL-09) MISSING ZERO VALUE CHECK - LOW

Description:

There are several functions in ConfirmationAggregator.sol and DeBridge.sol that should have a zero value check in place.

Code Location:

ConfirmationAggregator.sol: Line#168, Line#174

- setExcessConfirmations
- setThreshold

```
163
164     /* ========== ADMIN ========== */
165
166     /// @dev Sets minimal required confirmations.
167     /// @param _excessConfirmations Confirmation info.
        ftrace | funcSig
168     function setExcessConfirmations(uint8 _excessConfirmations) public onlyAdmin {
169         excessConfirmations = _excessConfirmations;
170     }
171
172     /// @dev Sets minimal required confirmations.      kstasi, 3 months ago • new oracle conf
173     /// @param _confirmationThreshold Confirmation info.
        ftrace | funcSig
174     function setThreshold(uint8 _confirmationThreshold) public onlyAdmin {
175         confirmationThreshold = _confirmationThreshold;
176     }
```

DeBridgeGate.sol: Line#558

- updateExcessConfirmations

```
        ftrace | funcSig
558     function updateExcessConfirmations(uint8 _excessConfirmations) external onlyAdmin {
559         excessConfirmations = _excessConfirmations;
560     }      Yaroslav, 2 months ago • Add test ExcessConfirmations
```

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

It is recommended to add a zero value check to ensure that those state variables won't be set to 0 by accident.

Remediation plan:

SOLVED: deBridge team solved the issue by adding additional conditional check to functions:
- setExcessConfirmations: `if (_excessConfirmations < minConfirmations) revert LowMinConfirmations();`
- setThreshold: `if (_confirmationThreshold == 0)revert WrongArgument();`
- updateExcessConfirmations: `if (_excessConfirmations == 0)revert WrongArgument();`

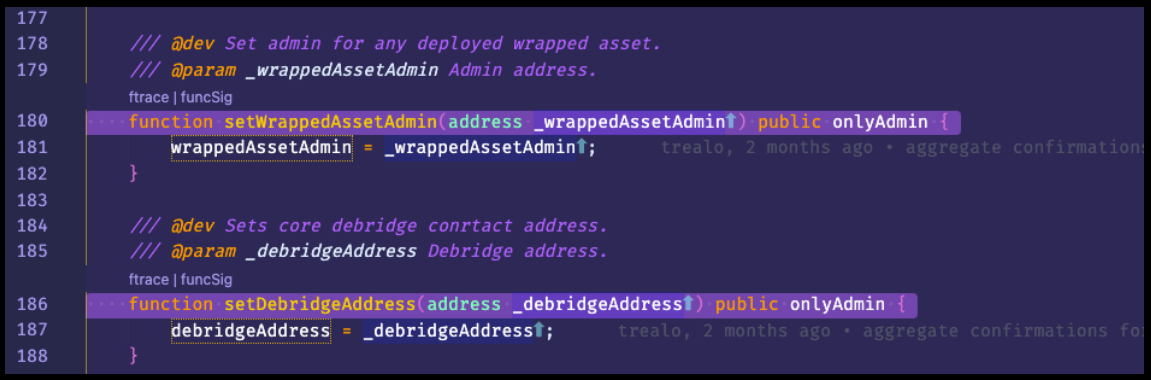
3.10 (HAL-10) MISSING ZERO ADDRESS CHECK - LOW

Description:

Lack of zero address validation has been found at many instances in ConfirmationAggregator.sol, FeeProxy.sol, DeBridgeGate.sol, and SignatureVerifier.sol when assigning user supplied address values to state variables directly.

Code Location:

ConfirmationAggregator.sol: Line#180, Line#186



```
177
178     /// @dev Set admin for any deployed wrapped asset.
179     /// @param _wrappedAssetAdmin Admin address.
        ftrace | funcSig
180     function setWrappedAssetAdmin(address _wrappedAssetAdmin) public onlyAdmin {
181         wrappedAssetAdmin = _wrappedAssetAdmin;      trealo, 2 months ago • aggregate confirmation
182     }
183
184     /// @dev Sets core debridge conrtact address.
185     /// @param _debridgeAddress Debridge address.
        ftrace | funcSig
186     function setDebridgeAddress(address _debridgeAddress) public onlyAdmin {
187         debridgeAddress = _debridgeAddress;      trealo, 2 months ago • aggregate confirmations fo
188     }
```

FeeProxy.sol: Line#85, Line#89, Line#93

```
        ftrace | funcSig
85      function setTreasury(uint256 _chainId, bytes memory _treasuryAddress) external onlyAdmin {
86          treasuryAddresses[_chainId] = _treasuryAddress;      artyukh, 3 weeks ago • receiver/token
87      }
88
        ftrace | funcSig
89      function setDeEthToken(address _deEthToken) external onlyAdmin {
90          deEthToken = _deEthToken;      artyukh, a month ago • Support withdraw collected fees
91      }
92
        ftrace | funcSig
93      function setFeeProxyAddress(uint256 _chainId, bytes memory _address) external onlyAdmin {
94          feeProxyAddresses[_chainId] = _address;      artyukh, 2 weeks ago • WIP. Upgrade withdraw f
95      }
```

DeBridgeGate.sol: Line#597, Line#603, Line#609, Line#689

- setAggregator
- setSignatureVerifier
- setDefiController

```
595     /// @dev Set aggregator address.
596     /// @param _aggregator Submission aggregator address.
        ftrace | funcSig
597     function setAggregator(address _aggregator) external onlyAdmin {
598         confirmationAggregator = _aggregator;      artyukh, a month ago • removed old aggregators methods
599     }
600
601     /// @dev Set signature verifier address.
602     /// @param _verifier Signature verifier address.
        ftrace | funcSig
603     function setSignatureVerifier(address _verifier) external onlyAdmin {
604         signatureVerifier = _verifier;      artyukh, a month ago • removed old aggregators methods
605     }
606
607     /// @dev Set defi controoler.
608     /// @param _defiController Defi controller address address.
        ftrace | funcSig
609     function setDefiController(address _defiController) external onlyAdmin {
610         // TODO: claim all the reserves before      kstasi, 6 months ago • support defi interface, fee swap, better nonces
611         defiController = _defiController;
612     }
```

- setFeeProxy

```
687     /// @dev Set fee converter proxy.
688     /// @param _feeProxy Fee proxy address.
        ftrace | funcSig
689     function setFeeProxy(address _feeProxy) external onlyAdmin {
690         feeProxy = _feeProxy;
691     }
```

SignatureVerifier.sol: Line#211, Line#217

- setWrappedAssetAdmin
- setDebridgeAddress

```
        ftrace | funcSig
211     function setWrappedAssetAdmin(address _wrappedAssetAdmin) public onlyAdmin {
212         wrappedAssetAdmin = _wrappedAssetAdmin;      trealo, 2 months ago • update migrations
213     }
214
215     /// @dev Sets core debridge conrtact address.
216     /// @param _debridgeAddress Debridge address.
        ftrace | funcSig
217     function setDebridgeAddress(address _debridgeAddress) public onlyAdmin {
218         debridgeAddress = _debridgeAddress;      trealo, 2 months ago • update migrations
219     }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

It is advised to have a proper address validation for every state variable assignment done from user supplied input.

Remediation plan:

PARTIALLY SOLVED: deBridge team partially solved the issue because the zero address check was only added to some functions. Also setWrappedAssetAdmin function was removed. Token deployments are now happening through new DeTokenDeployer.sol contract.

3.11 (HAL-11) USE OF INLINE ASSEMBLY - INFORMATIONAL

Description:

Inline assembly is a way to access the Ethereum Virtual Machine at a low level. Low level access discards several important safety features of Solidity, and the static compiler. Since the EVM is a stack machine, it is often hard to address the correct stack slot and provide arguments to opcodes at the correct point on the stack. Solidity's inline assembly tries to facilitate that and other issues arising when writing manual assembly. Assembly is much more difficult to write because the compiler does not perform checks, so the contract developer should be aware of this warning.

Code Location:

```
→ grep --include=\*.sol --exclude=mock -Rnw 'contracts' -e "assembly"
contracts/mock/MockLinkToken.sol:56:        assembly {
contracts/libraries/SignatureUtil.sol:41:       assembly {
contracts/libraries/SignatureUtil.sol:58:       assembly {
contracts/transfers/DeBridgeGate.sol:1194:       assembly {
contracts/periphery/CallProxy.sol:86:       assembly {
contracts/periphery/FeeProxy.sol:271:       assembly {
contracts/periphery/FeeProxy.sol:277:       assembly {
contracts/periphery/WrappedAsset.sol:42:       assembly {
```

Risk Level:

Likelihood - 1
Impact - 2

Recommendation:

When possible, do not use inline assembly because it is a way to access the EVM (Ethereum Virtual Machine) at a low level. An attacker could

bypass many important safety features of Solidity.

Remediation plan:

ACKNOWLEDGED: deBridge team acknowledged the issue.

3.12 (HAL-12) MISSING EVENT EMITTING - INFORMATIONAL

Description:

Functions that have important functionality in the DebridgeGate.sol, SignatureVerifier.sol, ConfirmationAggregator.sol, and FeeProxy.sol contracts are missing event emitting. These functions should emit events.

Code Location:

DebridgeGate.sol: Line#547, Line#558, Line#582, Line#597, Line#603, Line#609, Line#647, Line#669, Line#689, Line#706, Line#715

- updateAssetFixedFees
- updateExcessConfirmations
- updateAsset
- setAggregator
- setSignatureVerifier
- setDefiController
- requestReserves
- returnReserves
- setFeeProxy
- updateFlashFee
- updateFeeDiscount

SignatureVerifier.sol: Line#211, Line#217

- setWrappedAssetAdmin
- setDebridgeAddress

ConfirmationAggregator.sol: Line#168, Line#174, Line#180, Line#187

- setExcessConfirmations

- setThreshold
- setWrappedAssetAdmin
- setDebridgeAddress

FeeProxy.sol: Line#77, Line#81, Line#85, Line#89, Line#93, Line#99, Line#149

- setUniswapFactory
- setDebridgeGate
- setTreasury
- setDeEthToken
- setFeeProxyAddress
- withdrawFee
- withdrawNativeFee

Risk Level:

Likelihood - 2
Impact - 1

Recommendation:

Please add event emitting to mentioned functions

Remediation plan:

ACKNOWLEDGED: deBridge team acknowledged the issue and relevant events will be added later. Also setWrappedAssetAdmin function was removed. Also setWrappedAssetAdmin function was removed. Token deployments are now happening through new DeTokenDeployer.sol contract.

3.13 (HAL-13) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL

Description:

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

Code Location:

ConfirmationAggregator.sol

- setExcessConfirmations
- setThreshold
- setWrappedAssetAdmin
- setDebridgeAddress

SignatureVerifier.sol

- setWrappedAssetAdmin
- setDebridgeAddress

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider as much as possible declaring functions external instead of public. As for best practice, you should use external if you expect that the function will only be called externally and use public if you need to call the function internally. To sum up, all can access to public functions, external functions only can be accessed externally and internal functions can only be called within the contract.

Remediation:

SOLVED: deBridge team solved the issue by marking relevant functions as external. Also setWrappedAssetAdmin function was removed. Also setWrappedAssetAdmin function was removed. Token deployments are now happening through new DeTokenDeployer.sol contract.

3.14 (HAL-14) OUTDATED DOCUMENTATION - INFORMATIONAL

Description:

Documentation on docs.debridge.finance is outdated. Many functions have outdated arguments.

Code Location:

docs.debridge.finance

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider updating the documentation to reflect the latest changes in the code.

ACKNOWLEDGED: deBridge team acknowledged and the documentation will be updated later.

3.15 (HAL-15) INCONSISTENT CODE - INFORMATIONAL

Description:

There is an inconsistent code in one of the modifiers. In WrappedAsset contract require is used to validate condition while in other contracts if statement is used.

Code Location:

WrappedAsset.sol: Line#18

```
17      modifier onlyMinter() {
18          require(hasRole(MINTER_ROLE, msg.sender), "onlyMinter: bad role");
19          _;
20      }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Use if statement in WrappedAsset contract's modifier to ensure consistency across entire codebase.

```
18      /* ========== ERRORS ========== */
19       error MinterBadRole();
20
21      /* ========== MODIFIERS ========== */
22
23      modifier onlyMinter() {
24          if(!hasRole(MINTER_ROLE, msg.sender)) revert MinterBadRole();
25          _;
26      }       kstasi, 6 months ago • fix contracts
```

Remediation:

SOLVED: deBridge team solved the issue by turning `require` statement into `if` statement. Also `WrappedAsset.sol` contract was renamed to `DeBridgeToken.sol`

3.16 (HAL-16) TYPO IN CODE - INFORMATIONAL

Description:

There is a typo in a word Treasury in FeeProxy.sol

Code Location:

FeeProxy.sol: Line#109, Line#127, Line#130, Line#160, Line#168, Line #170

- withdrawFee

```
99    function withdrawFee(address _tokenAddress) external payable override onlyWorker whenNotPaused {
100       (uint256 nativeChain, bytes memory nativeAddress) = debridgeGate.getNativeTokenInfo(
101           _tokenAddress
102       );
103       bytes32 debridgeId = getbDebridgeId(nativeChain, nativeAddress);
104
105       uint256 chainId = getChainId();
106       if (feeProxyAddresses[nativeChain].length == 0) revert EmptyFeeProxyAddress(nativeChain);
107       if (treasuryAddresses[chainId].length == 0) revert EmptyTreasuryAddress(chainId);
108
109       address currentTreaseryAddress = toAddress(treasuryAddresses[chainId]);      artyukh, a month ag
110
111       debridgeGate.withdrawFee(debridgeId);
112       uint256 amount = IERC20(_tokenAddress).balanceOf(address(this));
113       // original token chain is the same as contract chain
114       if (chainId == nativeChain) {
115           //Reward is token (DBR, LINK, WETH, deDBT, deLINK, deETH)
116           //If token is deETH
117           if (_tokenAddress == deEthToken) {…
120           }
121           //For others tokens
122           else {
123               // create swap to weth
124               if (_tokenAddress != address(weth)) {…
126               }
127               //If we are in Ethereum chain transfer to Treasery      artyukh, a month ago • FeeProxy:
128               if (chainId == ETH_CHAINID) {
129                   IERC20(address(weth)).safeTransfer(
130                       address(currentTreaseryAddress),      artyukh, a month ago • FeeProxy:  add with
131                       weth.balanceOf(address(this))
```

- withdrawNativeFee

```
149     function withdrawNativeFee() external payable override onlyWorker whenNotPaused {
150         uint256 chainId = getChainId();
151         //DebridgeId of weth in ethereum network
152         //TODO: can be set as contstant
153         (, bytes memory nativeAddress) = debridgeGate.getNativeTokenInfo(deEthToken);
154         bytes32 wethEthNetworkDebridgeId = getbDebridgeId(ETH_CHAINID, nativeAddress);
155         if (feeProxyAddresses[chainId].length == 0) revert EmptyFeeProxyAddress(chainId);
156
157         // TODO: treasuryAddresses can keep only for ETH network
158         // if (treasuryAddresses[chainId].length == 0) revert EmptyTreasuryAddress(chainId);
159
160         // address currentTreaseryAddress = toAddress(treasuryAddresses[chainId]);
161         debridgeGate.withdrawFee(getDebridgeId(chainId, address(0)));
162         uint256 amount = address(this).balance - msg.value;
163
164         //reward is native token (ETH/BNB/HT)
165         //If we are in Ethereum chain
166         if (chainId == ETH_CHAINID) {
167             if (treasuryAddresses[chainId].length == 0) revert EmptyTreasuryAddress(chainId);
168             address currentTreaseryAddress = toAddress(treasuryAddresses[chainId]);
169             //TODO: send 50% reward to slashing contract
170             payable(currentTreaseryAddress).transfer(amount);
171         }      artyukh, a month ago • FeeProxy:  add withdrawNativeFee, Fix test 05_Deb…
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Please fix the typo by changing existing word to Treasury

Remediation:

SOLVED: deBridge team solved the issue by fixing the typo.

AUTOMATED TESTING

4.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance coverage of certain areas of the scoped contract. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their abi and binary formats. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Results:

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

```
Reentrancy in DeBridgeGate._send(address,uint256,uint256,bool,uint32) (contracts/transfers/DeBridgeGate.sol#886-940):
        External calls:
        - weth.deposit{value: msg.value}() (contracts/transfers/DeBridgeGate.sol#920)
        - token.safeTransferFrom(msg.sender,address(this),_amount) (contracts/transfers/DeBridgeGate.sol#925)
        External calls sending eth:
        - weth.deposit{value: msg.value}() (contracts/transfers/DeBridgeGate.sol#920)
        State variables written after the call(s):
        - debridge.balance += _amount (contracts/transfers/DeBridgeGate.sol#938)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

AccessControlUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#259) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#38)
PausableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#96) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

MockProxyConsumer.transferToken(address,address,address,bytes) (contracts/mock/MockProxyConsumer.sol#19-46) ignores return value by IERC20(_token).transfer(callProxy,msg.value) (contracts/mock/MockProxyConsumer.sol#35)
DeBridgeGate._burn(bytes32,uint256,uint256,bytes,bool,uint32) (contracts/transfers/DeBridgeGate.sol#947-986) ignores return value by wrappedAsset.transferFrom(msg.sender,address(this),_amount) (contracts/transfers/DeBridgeGate.sol#973)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

FeeProxy.getAmountOut(uint256,uint256,uint256) (contracts/periphery/FeeProxy.sol#269-282) uses a dangerous strict equality:
        - amountIn == 0 (contracts/periphery/FeeProxy.sol#275)
SignatureVerifier.submit(bytes32,bytes,uint8) (contracts/transfers/SignatureVerifier.sol#123-184) uses a dangerous strict equality:
        - currentBlock == uint40(block.number) (contracts/transfers/SignatureVerifier.sol#168)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in DeBridgeGate._burn(bytes32,uint256,uint256,bytes,bool,uint32) (contracts/transfers/DeBridgeGate.sol#947-986):
        External calls:
        - wrappedAsset.permit(msg.sender,address(this),_amount,deadline,v,r,s) (contracts/transfers/DeBridgeGate.sol#969)
        - wrappedAsset.transferFrom(msg.sender,address(this),_amount) (contracts/transfers/DeBridgeGate.sol#973)
        - wrappedAsset.burn(_amount) (contracts/transfers/DeBridgeGate.sol#983)
        State variables written after the call(s):
        - debridge.balance -= _amount (contracts/transfers/DeBridgeGate.sol#984)
Reentrancy in DeBridgeGate._checkAndDeployAsset(bytes32,address) (contracts/transfers/DeBridgeGate.sol#796-808):
        External calls:
        - (wrappedAssetAddress,nativeAddress,nativeChainId) = IConfirmationAggregator(_aggregatorAddress).deployAsset(_debridgeId) (contracts/transfers/DeBridgeGate.sol#798-802)
        State variables written after the call(s):
        - _addAsset(_debridgeId,wrappedAssetAddress,nativeAddress,nativeChainId) (contracts/transfers/DeBridgeGate.sol#805)
                - debridge.exist = true (contracts/transfers/DeBridgeGate.sol#856)
                - debridge.tokenAddress = _tokenAddress (contracts/transfers/DeBridgeGate.sol#857)
                - debridge.chainId = _chainId (contracts/transfers/DeBridgeGate.sol#858)
                - debridge.maxAmount = 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff (contracts/transfers/DeBridgeGate.sol#861)
Reentrancy in DeBridgeGate._mint(bytes32,bytes32,address,uint256,address,uint256,bytes,uint8,bytes) (contracts/transfers/DeBridgeGate.sol#1038-1080):
        External calls:
        - IWrappedAsset(debridge.tokenAddress).mint(msg.sender,_executionFee) (contracts/transfers/DeBridgeGate.sol#1058)
        - IWrappedAsset(debridge.tokenAddress).mint(callProxyAddress,_amount) (contracts/transfers/DeBridgeGate.sol#1064)
        - status = ICallProxy(callProxyAddress).callERC20(debridge.tokenAddress,_fallbackAddress,_receiver,_data,_reservedFlag,_nativeSender) (contracts/transfers/DeBridgeGate.sol#1066-1073)
        - IWrappedAsset(debridge.tokenAddress).mint(_receiver,_amount) (contracts/transfers/DeBridgeGate.sol#1076)
        State variables written after the call(s):
        - debridge.balance += _amount + _executionFee (contracts/transfers/DeBridgeGate.sol#1079)
```

```
Reentrancy in DeBridgeGate.autoMint(bytes32,uint256,address,uint256,uint256,bytes,address,uint256,bytes,uint8,bytes) (contracts/transfers/DeBridgeGate.sol#362-424):
        External calls:
        - _checkConfirmations(submissionId,_debridgeId,_amount,_signatures) (contracts/transfers/DeBridgeGate.sol#397)
              - ISignatureVerifier(signatureVerifier).submit(_submissionId,_signatures,excessConfirmations) (contracts/transfers/DeBridgeGate.sol#820-824)
              - ISignatureVerifier(signatureVerifier).submit(_submissionId,_signatures,0) (contracts/transfers/DeBridgeGate.sol#820-824)
        - _mint(submissionId,_debridgeId,_receiver,_amount,_fallbackAddress,_executionFee,_data,_reservedFlag,_nativeSender) (contracts/transfers/DeBridgeGate.sol#399-409)
              - IWrappedAsset(debridge.tokenAddress).mint(msg.sender,_executionFee) (contracts/transfers/DeBridgeGate.sol#1058)
              - IWrappedAsset(debridge.tokenAddress).mint(callProxyAddress,_amount) (contracts/transfers/DeBridgeGate.sol#1064)
              - status = ICallProxy(callProxyAddress).callERC20(debridge.tokenAddress,_fallbackAddress,_receiver,_data,_reservedFlag,_nativeSender) (contracts/transfers/DeBridgeGate.sol#1066-1073)
              - IWrappedAsset(debridge.tokenAddress).mint(_receiver,_amount) (contracts/transfers/DeBridgeGate.sol#1076)
        - _checkAndDeployAsset(_debridgeId,signatureVerifier) (contracts/transfers/DeBridgeGate.sol#392-395)
              - (wrappedAssetAddress,nativeAddress,nativeChainId) = IConfirmationAggregator(_aggregatorAddress).deployAsset(_debridgeId) (contracts/transfers/DeBridgeGate.sol#798-802)
        - _checkAndDeployAsset(_debridgeId,confirmationAggregator) (contracts/transfers/DeBridgeGate.sol#392-395)
              - (wrappedAssetAddress,nativeAddress,nativeChainId) = IConfirmationAggregator(_aggregatorAddress).deployAsset(_debridgeId) (contracts/transfers/DeBridgeGate.sol#798-802)
        State variables written after the call(s):
        - _mint(submissionId,_debridgeId,_receiver,_amount,_fallbackAddress,_executionFee,_data,_reservedFlag,_nativeSender) (contracts/transfers/DeBridgeGate.sol#399-409)
              - debridge.balance += _amount + _executionFee (contracts/transfers/DeBridgeGate.sol#1079)
Reentrancy in DeBridgeGate.mint(bytes32,uint256,address,uint256,uint256,bytes) (contracts/transfers/DeBridgeGate.sol#210-238):
        External calls:
        - _checkConfirmations(submissionId,_debridgeId,_amount,_signatures) (contracts/transfers/DeBridgeGate.sol#233)
              - ISignatureVerifier(signatureVerifier).submit(_submissionId,_signatures,excessConfirmations) (contracts/transfers/DeBridgeGate.sol#820-824)
              - ISignatureVerifier(signatureVerifier).submit(_submissionId,_signatures,0) (contracts/transfers/DeBridgeGate.sol#820-824)
        - _mint(submissionId,_debridgeId,_receiver,_amount,address(0),0,,0,) (contracts/transfers/DeBridgeGate.sol#235)
              - IWrappedAsset(debridge.tokenAddress).mint(msg.sender,_executionFee) (contracts/transfers/DeBridgeGate.sol#1058)
              - IWrappedAsset(debridge.tokenAddress).mint(callProxyAddress,_amount) (contracts/transfers/DeBridgeGate.sol#1064)
              - status = ICallProxy(callProxyAddress).callERC20(debridge.tokenAddress,_fallbackAddress,_receiver,_data,_reservedFlag,_nativeSender) (contracts/transfers/DeBridgeGate.sol#1066-1073)
              - IWrappedAsset(debridge.tokenAddress).mint(_receiver,_amount) (contracts/transfers/DeBridgeGate.sol#1076)
        - _checkAndDeployAsset(_debridgeId,signatureVerifier) (contracts/transfers/DeBridgeGate.sol#228-231)
              - (wrappedAssetAddress,nativeAddress,nativeChainId) = IConfirmationAggregator(_aggregatorAddress).deployAsset(_debridgeId) (contracts/transfers/DeBridgeGate.sol#798-802)
        - _checkAndDeployAsset(_debridgeId,confirmationAggregator) (contracts/transfers/DeBridgeGate.sol#228-231)
              - (wrappedAssetAddress,nativeAddress,nativeChainId) = IConfirmationAggregator(_aggregatorAddress).deployAsset(_debridgeId) (contracts/transfers/DeBridgeGate.sol#798-802)
        State variables written after the call(s):
        - _mint(submissionId,_debridgeId,_receiver,_amount,address(0),0,,0,) (contracts/transfers/DeBridgeGate.sol#235)
              - debridge.balance += _amount + _executionFee (contracts/transfers/DeBridgeGate.sol#1079)
Reentrancy in DeBridgeGate.requestReserves(address,uint256) (contracts/transfers/DeBridgeGate.sol#693-712):
        External calls:
        - IERC20(_tokenAddress).safeTransfer(address(defiController),_amount) (contracts/transfers/DeBridgeGate.sol#710)
        State variables written after the call(s):
        - debridge.lockedInStrategies += _amount (contracts/transfers/DeBridgeGate.sol#711)
Reentrancy in DeBridgeGate.returnReserves(address,uint256) (contracts/transfers/DeBridgeGate.sol#717-734):
        External calls:
        - IERC20(debridge.tokenAddress).safeTransferFrom(address(defiController),address(this),_amount) (contracts/transfers/DeBridgeGate.sol#726-730)
        State variables written after the call(s):
        - debridge.lockedInStrategies -= _amount (contracts/transfers/DeBridgeGate.sol#733)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

SignatureVerifier.confirmNewAsset(bytes,uint256,string,string,uint8,bytes).currentRequiredOraclesCount (contracts/transfers/SignatureVerifier.sol#87) is a local variable never initialized
SignatureVerifier.submit(bytes32,bytes,uint8).currentRequiredOraclesCount (contracts/transfers/SignatureVerifier.sol#133) is a local variable never initialized
ConfirmationAggregator.submitMany(bytes32[]).i (contracts/transfers/ConfirmationAggregator.sol#59) is a local variable never initialized
SignatureAggregator.submitMany(bytes32[],bytes[]).i (contracts/transfers/SignatureAggregator.sol#75) is a local variable never initialized
SignatureVerifier.submit(bytes32,bytes,uint8).confirmations (contracts/transfers/SignatureVerifier.sol#135) is a local variable never initialized
DeBridgeGate._processFeeForTransfer(bytes32,uint256,uint256,bool,uint32).fixedFee (contracts/transfers/DeBridgeGate.sol#998) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
```

```
MockLinkToken.constructor(string,string,uint8)._name (contracts/mock/MockLinkToken.sol#13) shadows:
        - ERC20._name (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#40) (state variable)
MockLinkToken.constructor(string,string,uint8)._symbol (contracts/mock/MockLinkToken.sol#14) shadows:
        - ERC20._symbol (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#41) (state variable)
MockToken.constructor(string,string,uint8)._name (contracts/mock/MockToken.sol#12) shadows:
        - ERC20._name (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#40) (state variable)
MockToken.constructor(string,string,uint8)._symbol (contracts/mock/MockToken.sol#13) shadows:
        - ERC20._symbol (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#41) (state variable)
WrappedAsset.constructor(string,string,uint8,address,address[])._name (contracts/periphery/WrappedAsset.sol#27) shadows:
        - ERC20._name (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#40) (state variable)
WrappedAsset.constructor(string,string,uint8,address,address[])._symbol (contracts/periphery/WrappedAsset.sol#28) shadows:
        - ERC20._symbol (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#41) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

MockAggregator.getRoundData(uint80).answer (contracts/mock/MockAggregator.sol#40) shadows:
        - MockAggregator.answer (contracts/mock/MockAggregator.sol#7) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

DeBridgeGate.initialize(uint8,address,address,address,uint256[],IDeBridgeGate.ChainSupportInfo[],IWETH,address,IDefiController,address) (contracts/transfers/DeBridgeGate.sol#131-171) should emit an event for:
        - feeProxy = _feeProxy (contracts/transfers/DeBridgeGate.sol#165)
DeBridgeGate.setFeeProxy(address) (contracts/transfers/DeBridgeGate.sol#738-743) should emit an event for:
        - feeProxy = _feeProxy (contracts/transfers/DeBridgeGate.sol#739)
MockDeBridgeGate.initializeMock(uint8,address,address,address,uint256[],IDeBridgeGate.ChainSupportInfo[],IWETH,address,IDefiController,address,uint256) (contracts/mock/MockDeBridgeGate.sol#16-64) should emit an event for:
        - feeProxy = _feeProxy (contracts/mock/MockDeBridgeGate.sol#60)
ConfirmationAggregator.initialize(uint8,uint8,uint8,address,address) (contracts/transfers/ConfirmationAggregator.sol#38-52) should emit an event for:
        - debridgeAddress = _debridgeAddress (contracts/transfers/ConfirmationAggregator.sol#49)
ConfirmationAggregator.setDebridgeAddress(address) (contracts/transfers/ConfirmationAggregator.sol#202-207) should emit an event for:
        - debridgeAddress = _debridgeAddress (contracts/transfers/ConfirmationAggregator.sol#205)
SignatureVerifier.initialize(uint8,uint8,uint8,address,address) (contracts/transfers/SignatureVerifier.sol#49-63) should emit an event for:
        - debridgeAddress = _debridgeAddress (contracts/transfers/SignatureVerifier.sol#60)
SignatureVerifier.setDebridgeAddress(address) (contracts/transfers/SignatureVerifier.sol#235-240) should emit an event for:
        - debridgeAddress = _debridgeAddress (contracts/transfers/SignatureVerifier.sol#236)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

MockDeBridgeGate.initializeMock(uint8,address,address,address,uint256[],IDeBridgeGate.ChainSupportInfo[],IWETH,address,IDefiController,address,uint256) (contracts/mock/MockDeBridgeGate.sol#16-64) should emit an event for:
        - chainId = overrideChainId (contracts/mock/MockDeBridgeGate.sol#48)
MockFeeProxy.overrideChainId(uint256) (contracts/mock/MockFeeProxy.sol#14-16) should emit an event for:
        - chainId = _chainId (contracts/mock/MockFeeProxy.sol#15)
AggregatorBase.setMinConfirmations(uint8) (contracts/transfers/AggregatorBase.sol#75-80) should emit an event for:
        - minConfirmations = _minConfirmations (contracts/transfers/AggregatorBase.sol#78)
ConfirmationAggregator.initialize(uint8,uint8,uint8,address,address) (contracts/transfers/ConfirmationAggregator.sol#38-52) should emit an event for:
        - confirmationThreshold = _confirmationThreshold (contracts/transfers/ConfirmationAggregator.sol#46)
        - excessConfirmations = _excessConfirmations (contracts/transfers/ConfirmationAggregator.sol#47)
ConfirmationAggregator.setExcessConfirmations(uint8) (contracts/transfers/ConfirmationAggregator.sol#177-181) should emit an event for:
        - excessConfirmations = _excessConfirmations (contracts/transfers/ConfirmationAggregator.sol#178)
ConfirmationAggregator.setThreshold(uint8) (contracts/transfers/ConfirmationAggregator.sol#185-189) should emit an event for:
        - confirmationThreshold = _confirmationThreshold (contracts/transfers/ConfirmationAggregator.sol#186)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

DeBridgeGate.initialize(uint8,address,address,address,uint256[],IDeBridgeGate.ChainSupportInfo[],IWETH,address,IDefiController,address)._signatureVerifier (contracts/transfers/DeBridgeGate.sol#133) lacks a zero-check on :
        - signatureVerifier = _signatureVerifier (contracts/transfers/DeBridgeGate.sol#156)
DeBridgeGate.initialize(uint8,address,address,address,uint256[],IDeBridgeGate.ChainSupportInfo[],IWETH,address,IDefiController,address)._confirmationAggregator (contracts/transfers/DeBridgeGate.sol#134) lacks a zero-check on :
        - confirmationAggregator = _confirmationAggregator (contracts/transfers/DeBridgeGate.sol#157)
```

```
DeBridgeGate.initialize(uint8,address,address,address,uint256[],IDeBridgeGate.ChainSupportInfo[],IWETH,address,IDefiController,address)._confirmationAggregator (contracts/transfers/DeBridgeGate.sol#134) lacks a zero-check on :
        - confirmationAggregator = _confirmationAggregator (contracts/transfers/DeBridgeGate.sol#157)
DeBridgeGate.initialize(uint8,address,address,address,uint256[],IDeBridgeGate.ChainSupportInfo[],IWETH,address,IDefiController,address)._feeProxy (contracts/transfers/DeBridgeGate.sol#139) lacks a zero-check on :
        - feeProxy = _feeProxy (contracts/transfers/DeBridgeGate.sol#165)
DeBridgeGate.initialize(uint8,address,address,address,uint256[],IDeBridgeGate.ChainSupportInfo[],IWETH,address,IDefiController,address)._treasury (contracts/transfers/DeBridgeGate.sol#141) lacks a zero-check on :
        - treasury = _treasury (contracts/transfers/DeBridgeGate.sol#166)
DeBridgeGate.setAggregator(address)._aggregator (contracts/transfers/DeBridgeGate.sol#632) lacks a zero-check on :
        - confirmationAggregator = _aggregator (contracts/transfers/DeBridgeGate.sol#633)
DeBridgeGate.setSignatureVerifier(address)._verifier (contracts/transfers/DeBridgeGate.sol#640) lacks a zero-check on :
        - signatureVerifier = _verifier (contracts/transfers/DeBridgeGate.sol#641)
DeBridgeGate.setFeeProxy(address)._feeProxy (contracts/transfers/DeBridgeGate.sol#738) lacks a zero-check on :
        - feeProxy = _feeProxy (contracts/transfers/DeBridgeGate.sol#739)
DeBridgeGate.updateTreasury(address)._address (contracts/transfers/DeBridgeGate.sol#783) lacks a zero-check on :
        - treasury = _address (contracts/transfers/DeBridgeGate.sol#784)
MockDeBridgeGate.initializeMock(uint8,address,address,address,address,uint256[],IDeBridgeGate.ChainSupportInfo[],IWETH,address,IDefiController,address,uint256)._signatureVerifier (contracts/mock/MockDeBridgeGate.sol#18) lacks a zero-check on :
        - signatureVerifier = _signatureVerifier (contracts/mock/MockDeBridgeGate.sol#51)
MockDeBridgeGate.initializeMock(uint8,address,address,address,address,uint256[],IDeBridgeGate.ChainSupportInfo[],IWETH,address,IDefiController,address,uint256)._confirmationAggregator (contracts/mock/MockDeBridgeGate.sol#19) lacks a zero-check on
        - confirmationAggregator = _confirmationAggregator (contracts/mock/MockDeBridgeGate.sol#52)
MockDeBridgeGate.initializeMock(uint8,address,address,address,address,uint256[],IDeBridgeGate.ChainSupportInfo[],IWETH,address,IDefiController,address,uint256)._feeProxy (contracts/mock/MockDeBridgeGate.sol#24) lacks a zero-check on :
        - feeProxy = _feeProxy (contracts/mock/MockDeBridgeGate.sol#60)
MockDeBridgeGate.initializeMock(uint8,address,address,address,address,uint256[],IDeBridgeGate.ChainSupportInfo[],IWETH,address,IDefiController,address,uint256)._treasury (contracts/mock/MockDeBridgeGate.sol#26) lacks a zero-check on :
        - treasury = _treasury (contracts/mock/MockDeBridgeGate.sol#61)
FeeProxy.setDeEthToken(address)._deEthToken (contracts/periphery/FeeProxy.sol#96) lacks a zero-check on :
        - deEthToken = _deEthToken (contracts/periphery/FeeProxy.sol#97)
MockFlashCallback.flash(address,address,address,uint256,bool)._tokenAddress (contracts/mock/MockFlashCallBack.sol#27) lacks a zero-check on :
        - lastTokenAddress = _tokenAddress (contracts/mock/MockFlashCallBack.sol#34)
MockFlashCallback.flash(address,address,address,uint256,bool)._flashReceiver (contracts/mock/MockFlashCallBack.sol#26) lacks a zero-check on :
        - lastFlashReceiver = _flashReceiver (contracts/mock/MockFlashCallBack.sol#35)
MockProxyConsumer.constructor(address,address)._callProxy (contracts/mock/MockProxyConsumer.sol#14) lacks a zero-check on :
        - callProxy = _callProxy (contracts/mock/MockProxyConsumer.sol#15)
MockProxyConsumer.constructor(address,address)._token (contracts/mock/MockProxyConsumer.sol#14) lacks a zero-check on :
        - token = _token (contracts/mock/MockProxyConsumer.sol#16)
CallProxy.call(address,address,bytes,uint8,bytes)._reserveAddress (contracts/periphery/CallProxy.sol#36) lacks a zero-check on :
        - address(_reserveAddress).transfer(msg.value) (contracts/periphery/CallProxy.sol#49)
ConfirmationAggregator.initialize(uint8,uint8,uint8,address,address)._wrappedAssetAdmin (contracts/transfers/ConfirmationAggregator.sol#42) lacks a zero-check on :
        - wrappedAssetAdmin = _wrappedAssetAdmin (contracts/transfers/ConfirmationAggregator.sol#48)
ConfirmationAggregator.initialize(uint8,uint8,uint8,address,address)._debridgeAddress (contracts/transfers/ConfirmationAggregator.sol#43) lacks a zero-check on :
        - debridgeAddress = _debridgeAddress (contracts/transfers/ConfirmationAggregator.sol#49)
ConfirmationAggregator.setWrappedAssetAdmin(address)._wrappedAssetAdmin (contracts/transfers/ConfirmationAggregator.sol#193) lacks a zero-check on :
        - wrappedAssetAdmin = _wrappedAssetAdmin (contracts/transfers/ConfirmationAggregator.sol#194)
ConfirmationAggregator.setDebridgeAddress(address)._debridgeAddress (contracts/transfers/ConfirmationAggregator.sol#202) lacks a zero-check on :
        - debridgeAddress = _debridgeAddress (contracts/transfers/ConfirmationAggregator.sol#203)
SignatureVerifier.initialize(uint8,uint8,uint8,address,address)._wrappedAssetAdmin (contracts/transfers/SignatureVerifier.sol#53) lacks a zero-check on :
        - wrappedAssetAdmin = _wrappedAssetAdmin (contracts/transfers/SignatureVerifier.sol#59)
SignatureVerifier.initialize(uint8,uint8,uint8,address,address)._debridgeAddress (contracts/transfers/SignatureVerifier.sol#54) lacks a zero-check on :
        - debridgeAddress = _debridgeAddress (contracts/transfers/SignatureVerifier.sol#60)
SignatureVerifier.setWrappedAssetAdmin(address)._wrappedAssetAdmin (contracts/transfers/SignatureVerifier.sol#226) lacks a zero-check on :
        - wrappedAssetAdmin = _wrappedAssetAdmin (contracts/transfers/SignatureVerifier.sol#227)
SignatureVerifier.setDebridgeAddress(address)._debridgeAddress (contracts/transfers/SignatureVerifier.sol#235) lacks a zero-check on :
        - debridgeAddress = _debridgeAddress (contracts/transfers/SignatureVerifier.sol#236)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in DeBridgeGate._burn(bytes32,uint256,uint256,bytes,bool,uint32) (contracts/transfers/DeBridgeGate.sol#947-986):
```

```
Reentrancy in DeBridgeGate._burn(bytes32,uint256,uint256,bytes,bool,uint32) (contracts/transfers/DeBridgeGate.sol#947-986):
        External calls:
        - wrappedAsset.permit(msg.sender,address(this),_amount,deadline,v,r,s) (contracts/transfers/DeBridgeGate.sol#969)
        - wrappedAsset.transferFrom(msg.sender,address(this),_amount) (contracts/transfers/DeBridgeGate.sol#973)
        State variables written after the call(s):
        - _amount = _processFeeForTransfer(_debridgeId,_amount,_chainIdTo,_useAssetFee,_referralCode) (contracts/transfers/DeBridgeGate.sol#974-980)
                - getDebridgeFeeInfo[nativeDebridgeId].collectedFees += msg.value (contracts/transfers/DeBridgeGate.sol#1013)
                - debridgeFee.collectedFees += transferFee (contracts/transfers/DeBridgeGate.sol#1025)
Reentrancy in DeBridgeGate._checkAndDeployAsset(bytes32,address) (contracts/transfers/DeBridgeGate.sol#796-808):
        External calls:
        - (wrappedAssetAddress,nativeAddress,nativeChainId) = IConfirmationAggregator(_aggregatorAddress).deployAsset(_debridgeId) (contracts/transfers/DeBridgeGate.sol#798-802)
        State variables written after the call(s):
        - _addAsset(_debridgeId,wrappedAssetAddress,nativeAddress,nativeChainId) (contracts/transfers/DeBridgeGate.sol#805)
                - getAmountThreshold[_debridgeId] = 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff (contracts/transfers/DeBridgeGate.sol#865-867)
        - _addAsset(_debridgeId,wrappedAssetAddress,nativeAddress,nativeChainId) (contracts/transfers/DeBridgeGate.sol#805)
                - tokenInfo.chainId = _chainId (contracts/transfers/DeBridgeGate.sol#871)
                - tokenInfo.nativeAddress = _nativeAddress (contracts/transfers/DeBridgeGate.sol#872)
Reentrancy in DeBridgeGate._send(address,uint256,uint256,bool,uint32) (contracts/transfers/DeBridgeGate.sol#886-940):
        External calls:
        - weth.deposit{value: msg.value}() (contracts/transfers/DeBridgeGate.sol#920)
        - token.safeTransferFrom(msg.sender,address(this),_amount) (contracts/transfers/DeBridgeGate.sol#925)
        External calls sending eth:
        - weth.deposit{value: msg.value}() (contracts/transfers/DeBridgeGate.sol#920)
        State variables written after the call(s):
        - _amount = _processFeeForTransfer(debridgeId,_amount,_chainIdTo,_useAssetFee,_referralCode) (contracts/transfers/DeBridgeGate.sol#930-936)
                - getDebridgeFeeInfo[nativeDebridgeId].collectedFees += msg.value (contracts/transfers/DeBridgeGate.sol#1013)
                - debridgeFee.collectedFees += transferFee (contracts/transfers/DeBridgeGate.sol#1025)
Reentrancy in DeBridgeGate.autoBurn(bytes32,bytes,uint256,uint256,bytes,bytes,bytes,bool,uint8,uint32) (contracts/transfers/DeBridgeGate.sol#435-478):
        External calls:
        - _amount = _burn(_debridgeId,_amount,_chainIdTo,_permit,_useAssetFee,_reservedFlag) (contracts/transfers/DeBridgeGate.sol#449)
                - wrappedAsset.permit(msg.sender,address(this),_amount,deadline,v,r,s) (contracts/transfers/DeBridgeGate.sol#969)
                - wrappedAsset.transferFrom(msg.sender,address(this),_amount) (contracts/transfers/DeBridgeGate.sol#973)
                - wrappedAsset.burn(_amount) (contracts/transfers/DeBridgeGate.sol#983)
        State variables written after the call(s):
        - nonce ++ (contracts/transfers/DeBridgeGate.sol#477)
Reentrancy in DeBridgeGate.autoSend(address,bytes,uint256,uint256,bytes,uint256,bytes,bool,uint8,uint32) (contracts/transfers/DeBridgeGate.sol#302-352):
        External calls:
        - (_amount,debridgeId) = _send(_tokenAddress,_amount,_chainIdTo,_useAssetFee,_referralCode) (contracts/transfers/DeBridgeGate.sol#316-322)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
                - weth.deposit{value: msg.value}() (contracts/transfers/DeBridgeGate.sol#920)
                - token.safeTransferFrom(msg.sender,address(this),_amount) (contracts/transfers/DeBridgeGate.sol#925)
        External calls sending eth:
        - (_amount,debridgeId) = _send(_tokenAddress,_amount,_chainIdTo,_useAssetFee,_referralCode) (contracts/transfers/DeBridgeGate.sol#316-322)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
                - weth.deposit{value: msg.value}() (contracts/transfers/DeBridgeGate.sol#920)
        State variables written after the call(s):
        - nonce ++ (contracts/transfers/DeBridgeGate.sol#351)
Reentrancy in DeBridgeGate.burn(bytes32,bytes,uint256,uint256,bytes,bool,uint32) (contracts/transfers/DeBridgeGate.sol#246-261):
        External calls:
        - _amount = _burn(_debridgeId,_amount,_chainIdTo,_permit,_useAssetFee,_referralCode) (contracts/transfers/DeBridgeGate.sol#256)
                - wrappedAsset.permit(msg.sender,address(this),_amount,deadline,v,r,s) (contracts/transfers/DeBridgeGate.sol#969)
```

```
Reentrancy in DeBridgeGate.flash(address,address,uint256,bytes) (contracts/transfers/DeBridgeGate.sol#540-563):
        External calls:
        - IERC20(_tokenAddress).safeTransfer(_receiver,_amount) (contracts/transfers/DeBridgeGate.sol#553)
        - IFlashCallback(msg.sender).flashCallback(currentFlashFee,_data) (contracts/transfers/DeBridgeGate.sol#554)
        State variables written after the call(s):
        - getDebridgeFeeInfo[debridgeId].collectedFees += paid (contracts/transfers/DeBridgeGate.sol#561)
Reentrancy in DeBridgeGate.send(address,bytes,uint256,uint256,bool,uint32) (contracts/transfers/DeBridgeGate.sol#180-203):
        External calls:
        - (_amount,debridgeId) = _send(_tokenAddress,_amount,_chainIdTo,_useAssetFee,_referralCode) (contracts/transfers/DeBridgeGate.sol#192-198)
              - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#92)
              - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
              - weth.deposit{value: msg.value}() (contracts/transfers/DeBridgeGate.sol#920)
              - token.safeTransferFrom(msg.sender,address(this),_amount) (contracts/transfers/DeBridgeGate.sol#925)
        External calls sending eth:
        - (_amount,debridgeId) = _send(_tokenAddress,_amount,_chainIdTo,_useAssetFee,_referralCode) (contracts/transfers/DeBridgeGate.sol#192-198)
              - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#131)
              - weth.deposit{value: msg.value}() (contracts/transfers/DeBridgeGate.sol#920)
        State variables written after the call(s):
        - nonce += (contracts/transfers/DeBridgeGate.sol#202)
Reentrancy in MockProxyReceiver.setArrayAndPullToken(address,uint256,uint256[]) (contracts/mock/MockProxyReceiver.sol#29-40):
        External calls:
        - IERC20(_token).safeTransferFrom(address(msg.sender),address(this),_amount) (contracts/mock/MockProxyReceiver.sol#37)
        State variables written after the call(s):
        - tokensReceived = balanceAfter - balanceBefore (contracts/mock/MockProxyReceiver.sol#39)
Reentrancy in MockProxyReceiver.setUint256AndPullToken(address,uint256,uint256) (contracts/mock/MockProxyReceiver.sol#42-53):
        External calls:
        - IERC20(_token).safeTransferFrom(address(msg.sender),address(this),_amount) (contracts/mock/MockProxyReceiver.sol#50)
        State variables written after the call(s):
        - tokensReceived = balanceAfter - balanceBefore (contracts/mock/MockProxyReceiver.sol#52)
Reentrancy in MockProxyConsumer.transferToken(address,address,address,bytes) (contracts/mock/MockProxyConsumer.sol#19-46):
        External calls:
        - status = ICallProxy(callProxy).call{value: msg.value}(_fallbackAddress,_receiver,_data,0,) (contracts/mock/MockProxyConsumer.sol#27-33)
        - IERC20(_token).transfer(callProxy,msg.value) (contracts/mock/MockProxyConsumer.sol#35)
        - status = ICallProxy(callProxy).callERC20(_token,_fallbackAddress,_receiver,_data,0,) (contracts/mock/MockProxyConsumer.sol#36-43)
        External calls sending eth:
        - status = ICallProxy(callProxy).call{value: msg.value}(_fallbackAddress,_receiver,_data,0,) (contracts/mock/MockProxyConsumer.sol#27-33)
        State variables written after the call(s):
        - lastOperationStatus = status (contracts/mock/MockProxyConsumer.sol#45)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in DeBridgeGate._burn(bytes32,uint256,uint256,bytes,bool,uint32) (contracts/transfers/DeBridgeGate.sol#947-986):
        External calls:
        - wrappedAsset.permit(msg.sender,address(this),_amount,deadline,v,r,s) (contracts/transfers/DeBridgeGate.sol#969)
        - wrappedAsset.transferFrom(msg.sender,address(this),_amount) (contracts/transfers/DeBridgeGate.sol#973)
        Event emitted after the call(s):
        - CollectedFee(nativeDebridgeId,_referralCode,msg.value) (contracts/transfers/DeBridgeGate.sol#1014)
              - _amount = _processFeeForTransfer(_debridgeId,_amount,_chainIdTo,_useAssetFee,_referralCode) (contracts/transfers/DeBridgeGate.sol#974-980)
        - CollectedFee(_debridgeId,_referralCode,transferFee) (contracts/transfers/DeBridgeGate.sol#1026)
              - _amount = _processFeeForTransfer(_debridgeId,_amount,_chainIdTo,_useAssetFee,_referralCode) (contracts/transfers/DeBridgeGate.sol#974-980)
Reentrancy in DeBridgeGate._checkAndDeployAsset(bytes32,address) (contracts/transfers/DeBridgeGate.sol#796-808):
        External calls:
        - (wrappedAssetAddress,nativeAddress,nativeChainId) = IConfirmationAggregator(_aggregatorAddress).deployAsset(_debridgeId) (contracts/transfers/DeBridgeGate.sol#798-802)
        Event emitted after the call(s):
```

```
grantRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.grantRole(bytes32,address) (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#179-181)
revokeRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.revokeRole(bytes32,address) (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#192-194)
renounceRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.renounceRole(bytes32,address) (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#210-214)
grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#170-172)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#183-185)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#201-205)
name() should be declared external:
        - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#60-62)
symbol() should be declared external:
        - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#68-70)
decimals() should be declared external:
        - ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#85-87)
        - MockLinkToken.decimals() (contracts/mock/MockLinkToken.sol#28-30)
        - MockToken.decimals() (contracts/mock/MockToken.sol#27-29)
        - WrappedAsset.decimals() (contracts/periphery/WrappedAsset.sol#116-118)
totalSupply() should be declared external:
        - ERC20.totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#92-94)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#99-101)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#119-121)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#130-133)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#148-162)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#176-179)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#195-203)
initializeMock(uint8,address,address,address,uint256[],IDeBridgeGate.ChainSupportInfo[],IWETH,address,IDefiController,address,uint256) should be declared external:
        - MockDeBridgeGate.initializeMock(uint8,address,address,address,uint256[],IDeBridgeGate.ChainSupportInfo[],IWETH,address,IDefiController,address,uint256) (contracts/mock/MockDeBridgeGate.sol#16-64)
getSubmissionId(bytes32,uint256,uint256,address,uint256) should be declared external:
        - MockDeBridgeGate.getSubmissionId(bytes32,uint256,uint256,uint256,address,uint256) (contracts/mock/MockDeBridgeGate.sol#78-90)
addDebridge(address,uint256,uint256,uint256,uint256,uint16,uint256,bool) should be declared external:
        - MockDeBridgeGateForDefiController.addDebridge(address,uint256,uint256,uint256,uint256,uint16,uint256,bool) (contracts/mock/MockDeBridgeGateForDefiController.sol#13-34)
initializeMock(IUniswapV2Factory,IWETH) should be declared external:
        - MockFeeProxy.initializeMock(IUniswapV2Factory,IWETH) (contracts/mock/MockFeeProxy.sol#9-11)
transferAndCall(address,uint256,bytes) should be declared external:
        - MockLinkToken.transferAndCall(address,uint256,bytes) (contracts/mock/MockLinkToken.sol#32-43)
initialize(uint8,uint8,uint8,address,address) should be declared external:
        - ConfirmationAggregator.initialize(uint8,uint8,uint8,address,address) (contracts/transfers/ConfirmationAggregator.sol#38-52)
setExcessConfirmations(uint8) should be declared external:
        - ConfirmationAggregator.setExcessConfirmations(uint8) (contracts/transfers/ConfirmationAggregator.sol#177-181)
setThreshold(uint8) should be declared external:
        - ConfirmationAggregator.setThreshold(uint8) (contracts/transfers/ConfirmationAggregator.sol#185-189)
setWrappedAssetAdmin(address) should be declared external:
```

```
        - MockToken.decimals() (contracts/mock/MockToken.sol#27-29)
        - WrappedAsset.decimals() (contracts/periphery/WrappedAsset.sol#116-118)
totalSupply() should be declared external:
        - ERC20.totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#92-94)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#99-101)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#119-121)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#130-133)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#148-162)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#176-179)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#195-203)
initializeMock(uint8,address,address,address,uint256[],IDeBridgeGate.ChainSupportInfo[],IWETH,address,IDefiController,address,uint256) should be declared external:
        - MockDeBridgeGate.initializeMock(uint8,address,address,address,uint256[],IDeBridgeGate.ChainSupportInfo[],IWETH,address,IDefiController,address,uint256) (contracts/mock/MockDeBridgeGate.sol#16-64)
getSubmissionId(bytes32,uint256,uint256,uint256,address,uint256) should be declared external:
        - MockDeBridgeGate.getSubmissionId(bytes32,uint256,uint256,uint256,address,uint256) (contracts/mock/MockDeBridgeGate.sol#78-90)
addDebridge(address,uint256,uint256,uint256,uint256,uint16,uint256,bool) should be declared external:
        - MockDeBridgeGateForDefiController.addDebridge(address,uint256,uint256,uint256,uint256,uint256,uint16,uint256,bool) (contracts/mock/MockDeBridgeGateForDefiController.sol#13-34)
initializeMock(IUniswapV2Factory,IWETH) should be declared external:
        - MockFeeProxy.initializeMock(IUniswapV2Factory,IWETH) (contracts/mock/MockFeeProxy.sol#9-11)
transferAndCall(address,uint256,bytes) should be declared external:
        - MockLinkToken.transferAndCall(address,uint256,bytes) (contracts/mock/MockLinkToken.sol#32-43)
initialize(uint8,uint8,uint8,address,address) should be declared external:
        - ConfirmationAggregator.initialize(uint8,uint8,uint8,address,address) (contracts/transfers/ConfirmationAggregator.sol#38-52)
setExcessConfirmations(uint8) should be declared external:
        - ConfirmationAggregator.setExcessConfirmations(uint8) (contracts/transfers/ConfirmationAggregator.sol#177-181)
setThreshold(uint8) should be declared external:
        - ConfirmationAggregator.setThreshold(uint8) (contracts/transfers/ConfirmationAggregator.sol#185-189)
setWrappedAssetAdmin(address) should be declared external:
        - ConfirmationAggregator.setWrappedAssetAdmin(address) (contracts/transfers/ConfirmationAggregator.sol#193-198)
setDebridgeAddress(address) should be declared external:
        - ConfirmationAggregator.setDebridgeAddress(address) (contracts/transfers/ConfirmationAggregator.sol#202-207)
initialize(uint8,address,address,address,uint256[],IDeBridgeGate.ChainSupportInfo[],IWETH,address,IDefiController,address) should be declared external:
        - DeBridgeGate.initialize(uint8,address,address,address,uint256[],IDeBridgeGate.ChainSupportInfo[],IWETH,address,IDefiController,address) (contracts/transfers/DeBridgeGate.sol#131-171)
initialize(uint8) should be declared external:
        - SignatureAggregator.initialize(uint8) (contracts/transfers/SignatureAggregator.sol#26-28)
initialize(uint8,uint8,uint8,address,address) should be declared external:
        - SignatureVerifier.initialize(uint8,uint8,uint8,address,address) (contracts/transfers/SignatureVerifier.sol#49-63)
setWrappedAssetAdmin(address) should be declared external:
        - SignatureVerifier.setWrappedAssetAdmin(address) (contracts/transfers/SignatureVerifier.sol#226-231)
setDebridgeAddress(address) should be declared external:
        - SignatureVerifier.setDebridgeAddress(address) (contracts/transfers/SignatureVerifier.sol#235-240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

setAnswer(int256) should be declared external:
        - MockAggregator.setAnswer(int256) (contracts/mock/MockAggregator.sol#9-11)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
. analyzed (63 contracts with 75 detectors), 536 result(s) found
```

4.2 AUTOMATED SECURITY SCAN

Description:

Halborn used automated security scanners to assist with detection of well-known security issues, and to identify low-hanging fruit 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 testers machine and sent the compiled results to the analyzers to locate any vulnerabilities. In addition, only security findings are considered as in-scope.

Results:

All of the findings were either detected during manual code review or false positive. Only results with findings are shown below.

- CallProxy.sol

AUTOMATED TESTING

Issues

These were the issues detected in the scan. You can use the toggles to filter by severity and display or hide ignored issues.

Severity: Low (2) Medium (1) High (0)

Ignored Issues: Hidden (0)

ID	Severity	Name	File	Location
SWC-107	Medium	Write to persistent state following external call	callproxy.sol	L: 63 C: 19
SWC-107	Low	A call to a user-supplied address is executed.	callproxy.sol	L: 97 C: 67
SWC-113	Low	Multiple calls are executed in the same transaction.	callproxy.sol	L: 63 C: 19

- ConfirmationAggregator.sol

Issues

These were the issues detected in the scan. You can use the toggles to filter by severity and display or hide ignored issues.

Severity: Low (2) Medium (0) High (0)

Ignored Issues: Hidden (0)

ID	Severity	Name	File	Location
SWC-120	Low	Potential use of "block.number" as source of randonmness.	confirmationaggregator.sol	L: 112 C: 39
SWC-120	Low	Potential use of "block.number" as source of randonmness.	confirmationaggregator.sol	L: 113 C: 38

- DeBridgeGate.sol

Issues

These were the issues detected in the scan. You can use the toggles to filter by severity and display or hide ignored issues.

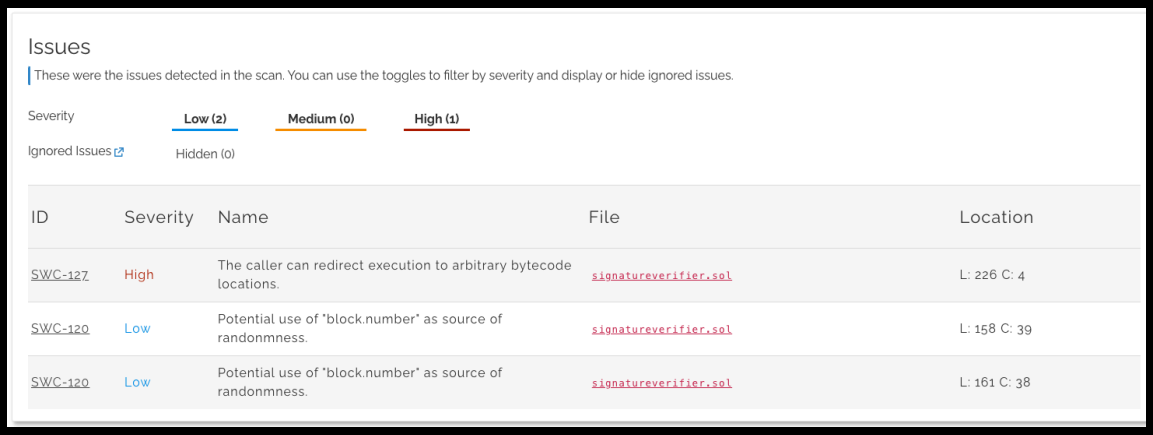
Severity: Low (2) Medium (0) High (0)

Ignored Issues: Hidden (0)

ID	Severity	Name	File	Location
SWC-107	Low	Read of persistent state following external call.	debridgegate.sol	L: 1032 C: 2
SWC-123	Low	Requirement violation.	debridgegate.sol	L: 39 C: 81

- SignatureVerifier.sol

AUTOMATED TESTING



Issues

These were the issues detected in the scan. You can use the toggles to filter by severity and display or hide ignored issues.

Severity: **Low (2)** **Medium (0)** **High (1)**

Ignored Issues: Hidden (0)

ID	Severity	Name	File	Location
SWC-127	High	The caller can redirect execution to arbitrary bytecode locations.	signatureverifier.sol	L: 226 C: 4
SWC-120	Low	Potential use of "block.number" as source of randonmness.	signatureverifier.sol	L: 158 C: 39
SWC-120	Low	Potential use of "block.number" as source of randonmness.	signatureverifier.sol	L: 161 C: 38

THANK YOU FOR CHOOSING

// HALBORN