// HALBORN

Biconomy - Cross Chain Messaging Protocol

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **September 26th, 2022 - October 21st, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY 5
CONTACTS 5
1 EXECUTIVE OVERVIEW 7
1.1 INTRODUCTION 8
1.2 AUDIT SUMMARY 8
1.3 TEST APPROACH & METHODOLOGY 8
RISK METHODOLOGY 9
1.4 SCOPE 11
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW 12
3 FINDINGS & TECH DETAILS 13
3.1 (HAL-01) FEE PAYMENT BYPASS - HIGH 15
Description 15
Code Location 15
Proof of Concept 16
Risk Level 17
Recommendation 17
Remediation Plan 17
3.2 (HAL-02) USING ARBITRARY TOKENS ALLOW FEE PAYMENT BYPASS - HIGH 18
Description 18
Code Location 18
Proof of Concept 19
Risk Level 20
Recommendation 20

Remediation Plan 21
3.3 (HAL-03) UN HANDLED SITUATION ALLOWS TWICE FEE PAYMENT - MEDIUM 22
Description 22
Code Location 22
Proof of Concept 23
Risk Level 24
Recommendation 24
Remediation Plan 24
3.4 (HAL-04) CONTRACT OWNER AND PAUSER CAN RENOUNCE HIMSELF - LOW 25
Description 25
Code Location 25
Proof of Concept 26
Risk Level 26
Recommendations 26
Remediation Plan 27
3.5 (HAL-05) MISSING ZERO ADDRESS CHECK - LOW 28
Description 28
Some code location examples 28
Risk Level 29
Recommendation 29
Remediation Plan 29
3.6 (HAL-06) DIFFERENT PRAGMA VERSIONS USED - INFORMATIONAL 30
Description 30

Code Location 30
Risk Level 31
Recommendation 31
Remediation Plan 31
3.7 (HAL-07) GAS OPTIMIZATIONS - INFORMATIONAL 32
Description 32
Risk Level 32
Recommendation 32
Remediation Plan 33
3.8 (HAL-08) OPEN TODOs - INFORMATIONAL 34
Description 34
Code Location 34
TO-DO 34
Risk Level 34
Recommendation 34
Remediation Plan 34
3.9 (HAL-09) INCREASE OPTIMIZER RUNS - INFORMATIONAL 35
Description 35
Risk Level 35
Code Location 35
Recommendation 36
Remediation Plan 36
4 AUTOMATED TESTING 37
4.1 STATIC ANALYSIS REPORT 38
Description 38
Slither results 38

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	10/14/2022	Luis Arroyo
0.2	Draft Review	10/24/2022	Kubilay Onur Gungor
0.3	Document Update	10/24/2022	Luis Arroyo
0.4	Draft Review	10/24/2022	Kubilay Onur Gungor
0.5	Draft Review	10/24/2022	Gabi Urrutia
1.0	Remediation Plan	11/09/2022	Luis Arroyo
1.1	Remediation Plan Review	11/10/2022	Kubilay Onur Gungor
1.2	Remediation Plan Review	11/10/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com

Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Kubilay Onur Gungor	Halborn	Kubilay.Gungor@halborn.com
Luis Arroyo	Halborn	Luis.Arroyo@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Biconomy engaged Halborn to conduct a security audit on their Cross Chain Messaging Protocol (CCMP) smart contracts beginning on September 26th, 2022 and ending on October 21st, 2022. The security assessment was scoped to the smart contracts provided in the GitHub repository bcnmy/ccmp-contracts with commit ID fb3fe86282f371086332843a55be3e5f11405c85.

1.2 AUDIT SUMMARY

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

The purpose of this audit is to:

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

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

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the code and can quickly identify

items that do not follow the security best practices. The following phases and associated tools were used during the audit:

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

RISK METHODOLOGY:

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

RISK SCALE - LIKELIHOOD

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

RISK SCALE - IMPACT

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

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

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

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

1.4 SCOPE

IN-SCOPE:
The security assessment was scoped to the following smart contracts:

-CCMPExecutor.sol
-AbacusAdapter.sol
-AxelarAdaptor.sol
-WormholeAdaptor.sol
-AbacusConnectionClient.sol
-CCMPAdaptorBase.sol
-Diamond.sol
-CCMPConfigurationFacet.sol
-CCMPReceiveMessageFacet.sol
-CCMPSendMessageFacet.sol
-DiamondCutFacet.sol
-DiamondInit.sol
-DiamondLoupeFacet.sol
-LibDiamond.sol
-CCMPReceiver.sol
-CCMPReceiverBase.sol
-CCMPReceiverUpgradeable.sol
-Pausable.sol
-Constants.sol
-CrossChainMessage.sol
-Wormhole.sol

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	2	1	2	4

LIKELIHOOD

IMPACT

			(HAL-02)	(HAL-01)
	(HAL-04)	(HAL-03)		
		(HAL-05)		
(HAL-07) (HAL-08) (HAL-09)	(HAL-06)			

EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - FEE PAYMENT BYPASS	High	RISK ACCEPTED
HAL02 - USING ARBITRARY TOKENS ALLOW FEE PAYMENT BYPASS	High	RISK ACCEPTED
HAL03 - UN HANDLED SITUATION ALLOWS TWICE FEE PAYMENT	Medium	SOLVED - 11/08/2022
HAL04 - CONTRACT OWNER AND PAUSER CAN RENOUNCE HIMSELF	Low	RISK ACCEPTED
HAL05 - MISSING ZERO ADDRESS CHECK	Low	SOLVED - 11/08/2022
HAL06 - DIFFERENT PRAGMA VERSIONS USED	Informational	SOLVED - 11/08/2022
HAL07 - GAS OPTIMIZATIONS	Informational	SOLVED - 11/08/2022
HAL08 - OPEN TODOs	Informational	ACKNOWLEDGED
HAL09 - INCREASE OPTIMIZER RUNS	Informational	SOLVED - 11/08/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) FEE PAYMENT BYPASS - HIGH

Description:

The CCMP protocol uses an internal struct sent inside the messages to determine the fee amount, the token used to pay it, and the relayer who will receive the fee. As this struct is filled by the user and sent in his message, it is possible to modify the relayer address by the users to pay fees to themselves.

Code Location:

Listing 1: CrossChainMessage.sol

```
14 struct GasFeePaymentArgs {
15     address feeTokenAddress;
16     uint256 feeAmount;
17     address relayer;
18 }
```

Listing 2: CCMPSendMessageFacet.sol (Line 94)

```
92 function _handleFee(CCMPMessage memory _message) internal {
93     uint256 feeAmount = _message.gasFeePaymentArgs.feeAmount;
94     address relayer = _message.gasFeePaymentArgs.relayer;
95     address tokenAddress = _message.gasFeePaymentArgs.
↳ feeTokenAddress;
96
97     if (feeAmount >= 0) {
98         if (_message.gasFeePaymentArgs.feeTokenAddress ==
↳ NATIVE_ADDRESS) {
99             if (msg.value != feeAmount) {
100                 revert NativeAmountMismatch();
101             }
102             (bool success, bytes memory returndata) = relayer.call
↳ {
103                 value: msg.value
104             }("");
```

```
105             if (!success) {
106                 revert NativeTransferFailed(relayer, returndata);
107             }
108         } else {
109             IERC20(_message.gasFeePaymentArgs.feeTokenAddress)
110                 .safeTransferFrom(
111                     _message.sender,
112                     relayer,
113                     _message.gasFeePaymentArgs.feeAmount
114                 );
115         }
116     }
117
118     emit FeePaid(tokenAddress, feeAmount, relayer);
119 }
```

Proof of Concept:

- Deploy a ICCMPGateway using the owner address.
- modify the gasFeePaymentArgs.relayer variable with the user address.
- Execute ICCMPGateway.sendMessage() function with the modified gasFeePaymentArgs.

Listing 3: feeBypass.js

```
gasFeePaymentArgs.relayer = relayer.address;

//initial balances
console.log("FEE -> " + ethers.utils.formatEther(gasFeePaymentArgs
↳ .feeAmount))
console.log("Owner ETH -> " + ethers.utils.formatEther(await owner
↳ .getBalance()).toString())

//paying fees correctly
await CCMPGateway.sendMessage(1, "wormhole", payloads,
↳ gasFeePaymentArgs, routeArgs,
    { value: gasFeePaymentArgs.feeAmount });

console.log("Owner ETH -> " + ethers.utils.formatEther(await owner
↳ .getBalance()).toString())

```

```
13 //change relayer to owner
14 gasFeePaymentArgs.relayer = owner.address;
15 await CCMPGateway.sendMessage(1, "wormhole", payloads,
↳ gasFeePaymentArgs, routeArgs,
16     { value: gasFeePaymentArgs.feeAmount })
17
18 console.log("Owner ETH -> " + ethers.utils.formatEther(await owner
↳ .getBalance()).toString())
```

Listing 4: Console Output

```
1 FEE -> 100.0
2 Owner ETH -> 9999.98221086197086311
3 Owner ETH -> 9899.982088081765816406
4 Owner ETH -> 9899.98198524440481391
```

Risk Level:

Likelihood - 5
Impact - 4

Recommendation:

Validating the relayer address before fee transfer is recommended.

Remediation Plan:

RISK ACCEPTED: The Biconomy team accepted the risk of this finding and implemented an off-chain service to correctly validate the payment to a whitelisted relayer address before sending the message.

3.2 (HAL-02) USING ARBITRARY TOKENS ALLOW FEE PAYMENT BYPASS - HIGH

Description:

The CCMP protocol uses an internal struct sent inside the messages to determine the fee amount, the token used to pay it, and the relayer who will receive the fee. As this struct is filled by the user and sent in his message, it is possible to modify the feeTokenAddress address and use an arbitrary token with a custom implementation of the transferFrom() function to avoid paying fees.

Code Location:

Listing 5: CrossChainMessage.sol

```
14 struct GasFeePaymentArgs {
15     address feeTokenAddress;
16     uint256 feeAmount;
17     address relayer;
18 }
```

Listing 6: CCMPSendMessageFacet.sol (Line 94)

```
92 function _handleFee(CCMPMessage memory _message) internal {
93     uint256 feeAmount = _message.gasFeePaymentArgs.feeAmount;
94     address relayer = _message.gasFeePaymentArgs.relayer;
95     address tokenAddress = _message.gasFeePaymentArgs.
↳ feeTokenAddress;
96
97     if (feeAmount >= 0) {
98         if (_message.gasFeePaymentArgs.feeTokenAddress ==
↳ NATIVE_ADDRESS) {
99             if (msg.value != feeAmount) {
100                 revert NativeAmountMismatch();
101             }
102             (bool success, bytes memory returndata) = relayer.call
↳ {
103                 value: msg.value
```

```
104             }("");
105             if (!success) {
106                 revert NativeTransferFailed(relayer, returndata);
107             }
108         } else {
109             IERC20(_message.gasFeePaymentArgs.feeTokenAddress)
110                 .safeTransferFrom(
111                     _message.sender,
112                     relayer,
113                     _message.gasFeePaymentArgs.feeAmount
114                 );
115         }
116     }
117
118     emit FeePaid(tokenAddress, feeAmount, relayer);
119 }
```

Proof of Concept:

- Deploy a ICCMPGateway using the owner address.
- Deploy an arbitrary ERC20 token.
- modify the gasFeePaymentArgs.feeTokenAddress variable with the token address.
- Execute ICCMPGateway.sendMessage() function with the modified gasFeePaymentArgs.

Listing 7: feeBypass.js

```
1 //deploying, minting and approving evil token
2 let evilToken = await (await (await ethers.getContractFactory("
↳ EvilToken")).deploy()).deployed();
3 await evilToken.mint(parseUnits("1000000", 18));
4 await evilToken.approve(CCMPGateway.address, gasFeePaymentArgs.
↳ feeAmount.mul(2));
5
6 //initial balances
7 console.log("FEE -> " + ethers.utils.formatEther(gasFeePaymentArgs
↳ .feeAmount))
8 console.log("Owner token -> " + ethers.utils.formatEther(await
↳ Token.balanceOf(owner.address)))
9 console.log("Owner evil token -> " + ethers.utils.formatEther(
```

```
↳ await evilToken.balanceOf(owner.address)))
10
11 //sending message with standard token
12 gasFeePaymentArgs.feeTokenAddress = Token.address;
13
14 await CCMPGateway.sendMessage(1, "wormhole", payloads,
↳ gasFeePaymentArgs, routeArgs);
15
16 console.log("Owner token -> " + ethers.utils.formatEther(await
↳ Token.balanceOf(owner.address)))
17
18 //sending message with evil token
19 gasFeePaymentArgs.feeTokenAddress = evilToken.address;
20
21 await CCMPGateway.sendMessage(1, "wormhole", payloads,
↳ gasFeePaymentArgs, routeArgs);
22
23 console.log("Owner evil token -> " + ethers.utils.formatEther(
↳ await evilToken.balanceOf(owner.address)))
```

Listing 8: Console Output

```
1 FEE -> 100.0
2 Owner token -> 1000000.0
3 Owner evil token -> 1000000.0
4 Owner token -> 999900.0
5 Owner evil token -> 1000000.0
```

Risk Level:

Likelihood - 4
Impact - 4

Recommendation:

Accepting native or well-known ERC20 tokens as fee payment tokens is recommended.

Remediation Plan:

RISK ACCEPTED: The Biconomy team will accept any token. Still, they have implemented an off-chain service to validate that the fee payment has been correctly received before sending the message.

3.3 (HAL-03) UN HANDLED SITUATION ALLOWS TWICE FEE PAYMENT - MEDIUM

Description:

When the CCMP protocol sends a message, an auxiliary function handles the fee payment. This function determines whether the payment is made via native ETH or ERC20 token. Suppose a user tries to send a message with native ETH, but the address added in the gasFeePaymentArgs struct is a valid ERC20 token address. In that case, the protocol will take the ETH sent as value and the token amount marked on gasFeePaymentArgs.feeAmount.

Code Location:

Listing 9: CCMPSendMessageFacet.sol (Line 94)

```
92 function _handleFee(CCMPMessage memory _message) internal {
93     uint256 feeAmount = _message.gasFeePaymentArgs.feeAmount;
94     address relayer = _message.gasFeePaymentArgs.relayer;
95     address tokenAddress = _message.gasFeePaymentArgs.
↳ feeTokenAddress;
96
97     if (feeAmount >= 0) {
98         if (_message.gasFeePaymentArgs.feeTokenAddress ==
↳ NATIVE_ADDRESS) {
99             if (msg.value != feeAmount) {
100                 revert NativeAmountMismatch();
101             }
102             (bool success, bytes memory returndata) = relayer.call
↳ {
103                 value: msg.value
104             }("");
105             if (!success) {
106                 revert NativeTransferFailed(relayer, returndata);
107             }
108         } else {
109             IERC20(_message.gasFeePaymentArgs.feeTokenAddress)
110                 .safeTransferFrom(
111                     _message.sender,
112                     relayer,
```

```
113                     _message.gasFeePaymentArgs.feeAmount
114                 );
115             }
116         }
117
118         emit FeePaid(tokenAddress, feeAmount, relayer);
119 }
```

Proof of Concept:

- Deploy a ICCMPGateway using the owner address.
- modify the gasFeePaymentArgs.feeTokenAddress variable with a token address.
- Execute ICCMPGateway.sendMessage() function with the modified gasFeePaymentArgs.

Listing 10: feeBypass.js

```
1 //initial balances
2 console.log("FEE -> " + ethers.utils.formatEther(gasFeePaymentArgs
↳ .feeAmount))
3 console.log("Owner ETH -> " + ethers.utils.formatEther(await owner
↳ .getBalance()).toString())
4 console.log("Owner token -> " + ethers.utils.formatEther(await
↳ Token.balanceOf(owner.address)))
5
6 //sending message with ERC20token
7 gasFeePaymentArgs.feeTokenAddress = Token.address;
8 await CCMPGateway.sendMessage(1, "wormhole", payloads,
↳ gasFeePaymentArgs, routeArgs);
9
10 console.log("Owner ETH -> " + ethers.utils.formatEther(await owner
↳ .getBalance()).toString())
11 console.log("Owner token -> " + ethers.utils.formatEther(await
↳ Token.balanceOf(owner.address)))
12
13 //sending message with ERC20 token and ETH
14 await CCMPGateway.sendMessage(1, "wormhole", payloads,
↳ gasFeePaymentArgs, routeArgs,
15 { value: gasFeePaymentArgs.feeAmount });
16
```

```
17 console.log("Owner ETH -> " + ethers.utils.formatEther(await owner
↳ .getBalance()).toString())
18 console.log("Owner token -> " + ethers.utils.formatEther(await
↳ Token.balanceOf(owner.address)))
```

Listing 11: Console Output

```
1 FEE -> 100.0
2 Owner ETH -> 9999.984234153287750867
3 Owner token -> 1000000.0
4 Owner ETH -> 9999.984087518824239997
5 Owner token -> 999900.0
6 Owner ETH -> 9899.983963765455794609
7 Owner token -> 999800.0
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

It is recommended to check the msg.value is 0 when ERC20 tokens are used as payment.

Remediation Plan:

SOLVED: The Biconomy team now checks that the native balance is 0 in case the user is paying with another token.

3.4 (HAL-04) CONTRACT OWNER AND PAUSER CAN RENOUNCE HIMSELF - LOW

Description:

The Owner of the contract is usually the account that deploys the contract. The ICCMPGateway smart contract, Only Owner and Pauser can perform some privileged actions such as setCCMPGateway, setCCMPExecutor, pause etc., the transferOwnership() and setPauser() functions are used to change the Owner and Pauser. Suppose an Owner or Pauser is mistakenly changed to the contract address. In that case, administrative access will result in the contract having no Owner or Pauser, eliminating the ability to call privileged functions. In such a case, contracts would have to be redeployed.

Code Location:

Listing 12: ownable.sol (Lines 62,71,80)

```
61    function renounceAdmin() public virtual
62     function renounceOwnership() public virtual onlyOwner {
63         _transferOwnership(address(0));
64     }
65
66     /**
67      * @dev Transfers ownership of the contract to a new account
↳ (`newOwner`).
68      * Can only be called by the current owner.
69      */
70     function transferOwnership(address newOwner) public virtual
↳ onlyOwner {
71         require(newOwner != address(0), "Ownable: new owner is the
↳  zero address");
72         _transferOwnership(newOwner);
73     }
74
75     /**
76      * @dev Transfers ownership of the contract to a new account
↳ (`newOwner`).
```

```
77      * Internal function without access restriction.
78      */
79     function _transferOwnership(address newOwner) internal virtual
↳ {
80         address oldOwner = _owner;
81         _owner = newOwner;
82         emit OwnershipTransferred(oldOwner, newOwner);
83     }
```

Listing 13: CCMPConfigurationFacet.sol (Line 77)

```
75     function setPauser(address _pauser) external {
76         LibDiamond._enforceIsContractOwner();
77         LibDiamond._diamondStorage().pauser = _pauser;
78         emit PauserUpdated(_pauser);
79     }
```

Proof of Concept:

- Deploy a ICCMPGateway using the owner address.
- Execute ICCMPGateway.setPauser() function with the contract address as parameter.
- Execute ICCMPGateway.transferOwnership() function with the contract address as parameter.

Risk Level:

Likelihood - 2
Impact - 3

Recommendations:

It is recommended that the contract Owner cannot set new Pausers or Owners without checking the address before. In addition, if a multi-signature wallet is used, calling the transferOwnership() and setPauser() functions should be confirmed for two or more users.

Remediation Plan:

RISK ACCEPTED: The Biconomy team accepted the risk of this finding.

3.5 (HAL-05) MISSING ZERO ADDRESS CHECK - LOW

Description:

There is no validation of the addresses in the `contractor()` of the Adaptors like `WormholeAdaptor`, `AxelarAdaptor` or `AbacusAdapter`, and other smart contracts. Addresses should be validated and checked that are different from zero when necessary. This issue is present in all the smart contracts, in the constructors and functions that use addresses as parameters. These examples show how the factory could be set up with a wrong address and how mint could burn NFTs if `_owner` is `address(0)`

Some code location examples:

Listing 14: AxelarAdaptor.sol

```
55 constructor(
56         address _axelarGateway,
57         address _ccmpGateway,
58         address _pauser
59     ) CCMPAdaptorBase(_ccmpGateway, _pauser) {
60         axelarGateway = IAxelarGateway(_axelarGateway);
61 
62         // Setup Mainnet Chain ID to Names
63         chainIdToName[1313161554] = "aurora";
64         chainIdToName[43114] = "Avalanche";
65         chainIdToName[56] = "binance";
66         chainIdToName[1] = "Ethereum";
67         chainIdToName[250] = "Fantom";
68         chainIdToName[1284] = "Moonbeam";
69         chainIdToName[137] = "Polygon";
70 
71         // Setup Testnet Chain ID to Names
72         chainIdToName[1313161555] = "aurora";
73         chainIdToName[43113] = "Avalanche";
74         chainIdToName[97] = "binance";
75         chainIdToName[3] = "Ethereum";
76         chainIdToName[4002] = "Fantom";
77         chainIdToName[1287] = "Moonbeam";
```

```
78          chainIdToName[80001] = "Polygon";
79      }
```

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

Validate that necessary address inputs are different from zero.

Remediation Plan:

SOLVED: The Biconomy team implemented verifications to check if the parameters in the constructor are equal to address(0).

3.6 (HAL-06) DIFFERENT PRAGMA VERSIONS USED - INFORMATIONAL

Description:

The CCMP smart contracts use floating pragma and different pragma versions, like >=0.6.11. Latest pragma version 0.8.16 released on August 8th, 2022, is used in some contracts, but ^0.8.0 is also used. Avoid using floating pragma is recommended as best practices.

Reference: Solidity Releases

Code Location:

- >=0.6.11 (contracts/adaptors/base/AbacusConnectionClient.sol#2)
- ^0.8.0 (contracts/gateway/facets/DiamondInit.sol#2)
- ^0.8.0 (contracts/gateway/facets/DiamondLoupeFacet.sol#2)
- ^0.8.0 (contracts/interfaces/IAxelarGateway.sol#3)
- ^0.8.0 (contracts/interfaces/ICCMPGateway.sol#2)
- 0.8.16 (contracts/interfaces/ICCMPRouterAdaptor.sol#2)
- ^0.8.0 (contracts/interfaces/IDiamond.sol#2)
- ^0.8.0 (contracts/interfaces/IDiamondCut.sol#2)
- ^0.8.0 (contracts/interfaces/IDiamondLoupe.sol#2)
- ^0.8.0 (contracts/interfaces/IERC165.sol#2)
- ^0.8.0 (contracts/interfaces/IERC173.sol#2)
- ^0.8.0 (contracts/interfaces/IWormhole.sol#4)
- ^0.8.16 (contracts/libraries/LibDiamond.sol#2)
- ^0.8.0 (contracts/mock/MockWormhole.sol#4)
- ^0.8.0 (contracts/structures/CrossChainMessage.sol#2)
- ^0.8.0 (contracts/structures/Wormhole.sol#4)

Note: The contracts with version 0.8.16 are not shown in the list.

Risk Level:

Likelihood - 2
Impact - 1

Recommendation:

It is recommended to update the pragma versions used in the CCMP smart contracts and lock them on the used version 0.8.16.

Remediation Plan:

SOLVED: The Biconomy team now uses Solidity version 0.8.16 in their contracts.

3.7 (HAL-07) GAS OPTIMIZATIONS - INFORMATIONAL

Description:

The Pausable.sol contract contains several functions that could be marked as external instead of public in order to save gas.

Listing 15: Pausable.sol

```
58     function changePauser(address newPauser) public onlyPauser
↳ whenNotPaused {
59         _changePauser(newPauser);
60     }
```

Listing 16: Pausable.sol

```
77     function pause() public onlyPauser {
78         _pause();
79     }
80
81     function unpause() public onlyPauser {
82         _unpause();
83     }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to change the public visibility of the mentioned functions to external to save gas.

Remediation Plan:

SOLVED: The Biconomy team implemented the mentioned functions as external instead of public.

3.8 (HAL-08) OPEN TODOs - INFORMATIONAL

Description:

Open To-dos can point to architecture or programming issues that still need to be resolved. Often these kinds of comments indicate areas of complexity or confusion for developers. This provides value and insight to an attacker who aims to cause damage to the protocol.

Code Location:

TO-DO:

Listing 17: Open ToDOs

```
1 CCMPReceiverMessageFacet.sol#L1
2 CrossChainMessage.sol#L7
3 CrossChainMessage.sol#L53
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider resolving the To-dos before deploying code to a production context. Use an independent issue tracker or other project management software to track development tasks.

Remediation Plan:

ACKNOWLEDGED: The Biconomy team acknowledged this finding.

3.9 (HAL-09) INCREASE OPTIMIZER RUNS - INFORMATIONAL

Description:

Solidity 0.8.2 and 0.8.16 have good optimizers that save a gas when compiling to bytecode. The team can use it by increasing the number of runs to something like **2000** at least in the config.

Risk Level:

Likelihood - 1
Impact - 1

Code Location:

Listing 18: hsrdhat.config.ts (Lines 27,41)

```
15 solidity: {
16     compilers: [
17       {
18         version: "0.8.2",
19         settings: {
20           outputSelection: {
21             "*": {
22               "*": ["storageLayout"],
23             },
24           },
25           optimizer: {
26             enabled: true,
27             runs: 200,
28           },
29         },
30       },
31       {
32         version: "0.8.16",
33         settings: {
34           outputSelection: {
35             "*": {
```

```
36                "*": ["storageLayout"],
37              },
38            },
39            optimizer: {
40              enabled: true,
41              runs: 200,
42            },
43            viaIR: true,
44          },
45        },
46      ],
47    },
```

Recommendation:

Consider increasing optimizer runs.

Remediation Plan:

SOLVED: The Biconomy team has set the compiler optimizer to 2000 runs.

AUTOMATED TESTING

4.1 STATIC ANALYSIS REPORT

Description:

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

Slither results:

Slither results for the Smart Contracts:

```
Contract locking ether found:
        Contract MockWormhole (contracts/mock/MockWormhole.sol#8-43) has payable functions:
         - MockWormhole.publishMessage(uint32,bytes,uint8) (contracts/mock/MockWormhole.sol#14-21)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether

Reentrancy in WormholeAdaptor.routePayload(CCMPMessage,bytes) (contracts/adaptors/WormholeAdaptor.sol#109-126):
        External calls:
        - sequenceId = wormhole.publishMessage(wormholeMessageNonce,abi.encode(_message.hash()),consistencyLevel) (contracts/adaptors/WormholeAdaptor.sol#
114-118)
        State variables written after the call(s):
        - ++ wormholeMessageNonce (contracts/adaptors/WormholeAdaptor.sol#125)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

CCMPSendMessageFacet._handleFee(CCMPMessage) (contracts/gateway/facets/CCMPSendMessageFacet.sol#92-119) contains a tautology or contradiction:
        - feeAmount >= 0 (contracts/gateway/facets/CCMPSendMessageFacet.sol#97)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-contradiction

LibDiamond._addFunctions(address,bytes4[]).selectorIndex (contracts/libraries/LibDiamond.sol#190) is a local variable never initialized
LibDiamond._diamondCut(IDiamond.FacetCut[],address,bytes).facetIndex (contracts/libraries/LibDiamond.sol#151) is a local variable never initialized
LibDiamond._removeFunctions(address,bytes4[]).selectorIndex (contracts/libraries/LibDiamond.sol#264) is a local variable never initialized
LibDiamond._replaceFunctions(address,bytes4[]).selectorIndex (contracts/libraries/LibDiamond.sol#227) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

AbacusAdapter.constructor(address,address,address,address)._pauser (contracts/adaptors/AbacusAdapter.sol#51) shadows:
        - PausableBase._pauser (contracts/security/Pausable.sol#17) (state variable)
AxelarAdaptor.constructor(address,address,address)._pauser (contracts/adaptors/AxelarAdaptor.sol#48) shadows:
        - PausableBase._pauser (contracts/security/Pausable.sol#17) (state variable)
WormholeAdaptor.constructor(address,address,address,WormholeAdaptor.DeploymentConfiguartion)._pauser (contracts/adaptors/WormholeAdaptor.sol#41) shadows:
        - PausableBase._pauser (contracts/security/Pausable.sol#17) (state variable)
CCMPAdaptorBase.constructor(address,address)._pauser (contracts/adaptors/base/CCMPAdaptorBase.sol#35) shadows:
        - PausableBase._pauser (contracts/security/Pausable.sol#17) (state variable)
CCMPReceiver.constructor(address)._ccmpExecutor (contracts/receiver/CCMPReceiver.sol#7) shadows:
        - CCMPReceiverBase._ccmpExecutor (contracts/receiver/CCMPReceiverBase.sol#9) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
```

```
CCMPExecutor.constructor(address)._ccmpGateway (contracts/CCMPExecutor.sol#24) lacks a zero-check on :
                - ccmpGateway = _ccmpGateway (contracts/CCMPExecutor.sol#25)
CCMPExecutor.execute(address,bytes)._to (contracts/CCMPExecutor.sol#29) lacks a zero-check on :
                - (success,returndata) = _to.call{gas: gasleft()()}(_calldata) (contracts/CCMPExecutor.sol#34)
CCMPExecutor.updateCCMPGateway(address)._newCCMPGateway (contracts/CCMPExecutor.sol#37) lacks a zero-check on :
                - ccmpGateway = _newCCMPGateway (contracts/CCMPExecutor.sol#38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in AxelarAdaptor.execute(bytes32,string,string,bytes) (contracts/adaptors/AxelarAdaptor.sol#126-169):
        External calls:
        - ! axelarGateway.validateContractCall(_commandId,_sourceChain,_sourceAddress,payloadHash) (contracts/adaptors/AxelarAdaptor.sol#136-141)
        State variables written after the call(s):
        - messageHashVerified[ccmpMessageHash] = true (contracts/adaptors/AxelarAdaptor.sol#160)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in CCMPSendMessageFacet._handleFee(CCMPMessage) (contracts/gateway/facets/CCMPSendMessageFacet.sol#92-119):
        External calls:
        - (success,returndata) = relayer.call{value: msg.value}() (contracts/gateway/facets/CCMPSendMessageFacet.sol#102-104)
        - IERC20(_message.gasFeePaymentArgs.feeTokenAddress).safeTransferFrom(_message.sender,relayer,_message.gasFeePaymentArgs.feeAmount) (contracts/gat
eway/facets/CCMPSendMessageFacet.sol#109-114)
        External calls sending eth:
        - (success,returndata) = relayer.call{value: msg.value}() (contracts/gateway/facets/CCMPSendMessageFacet.sol#102-104)
        Event emitted after the call(s):
        - FeePaid(tokenAddress,feeAmount,relayer) (contracts/gateway/facets/CCMPSendMessageFacet.sol#118)
Reentrancy in AxelarAdaptor.execute(bytes32,string,string,bytes) (contracts/adaptors/AxelarAdaptor.sol#126-169):
        External calls:
        - ! axelarGateway.validateContractCall(_commandId,_sourceChain,_sourceAddress,payloadHash) (contracts/adaptors/AxelarAdaptor.sol#136-141)
        Event emitted after the call(s):
        - AxelarMessageVerified(_commandId,_sourceChain,_sourceAddress,_payload,ccmpMessageHash) (contracts/adaptors/AxelarAdaptor.sol#162-168)
Reentrancy in AbacusAdapter.routePayload(CCMPMessage,bytes) (contracts/adaptors/AbacusAdapter.sol#116-149):
        External calls:
        - messageId = _outbox().dispatch(destinationChainDomainId,destinationRouterAddressEncoded,abi.encode(_message.hash())) (contracts/adaptors/AbacusA
dapter.sol#142-146)
        Event emitted after the call(s):
        - AbacusMessageRouted(messageId) (contracts/adaptors/AbacusAdapter.sol#148)
Reentrancy in AxelarAdaptor.routePayload(CCMPMessage,bytes) (contracts/adaptors/AxelarAdaptor.sol#73-103):
        External calls:
        - axelarGateway.callContract(destinationChainName,destinationRouterAddress,abi.encode(_message.hash())) (contracts/adaptors/AxelarAdaptor.sol#96-1
00)
        Event emitted after the call(s):
        - AxelarMessageRouted() (contracts/adaptors/AxelarAdaptor.sol#102)
Reentrancy in WormholeAdaptor.routePayload(CCMPMessage,bytes) (contracts/adaptors/WormholeAdaptor.sol#109-126):
        External calls:
        - sequenceId = wormhole.publishMessage(wormholeMessageNonce,abi.encode(_message.hash()),consistencyLevel) (contracts/adaptors/WormholeAdaptor.sol#
114-118)
        Event emitted after the call(s):
        - CCMPMessageRoutedViaWormhole(wormholeMessageNonce,sequenceId,consistencyLevel) (contracts/adaptors/WormholeAdaptor.sol#119-123)
Reentrancy in CCMPSendMessageFacet.sendMessage(uint256,string,CCMPMessagePayload[],GasFeePaymentArgs,bytes) (contracts/gateway/facets/CCMPSendMessageFacet
.sol#25-89):
        External calls:
        - _handleFee(message) (contracts/gateway/facets/CCMPSendMessageFacet.sol#70)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/
SafeERC20.sol#110)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
                - (success,returndata) = relayer.call{value: msg.value}() (contracts/gateway/facets/CCMPSendMessageFacet.sol#102-104)
                - IERC20(_message.gasFeePaymentArgs.feeTokenAddress).safeTransferFrom(_message.sender,relayer,_message.gasFeePaymentArgs.feeAmount) (contr
acts/gateway/facets/CCMPSendMessageFacet.sol#109-114)
        - adaptor.routePayload(message,_routerArgs) (contracts/gateway/facets/CCMPSendMessageFacet.sol#72)
        External calls sending eth:
        - _handleFee(message) (contracts/gateway/facets/CCMPSendMessageFacet.sol#70)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
                - (success,returndata) = relayer.call{value: msg.value}() (contracts/gateway/facets/CCMPSendMessageFacet.sol#102-104)
        Event emitted after the call(s):
        - CCMPMessageRouted(message.hash(),message.sender,message.sourceGateway,message.sourceAdaptor,message.sourceChainId,message.destinationGateway,mes
sage.destinationChainId,message.nonce,message.routerAdaptor,message.gasFeePaymentArgs,message.payload) (contracts/gateway/facets/CCMPSendMessageFacet.sol#
74-86)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

TypeCasts.coerceString(bytes32) (node_modules/@abacus-network/core/contracts/libs/TypeCasts.sol#6-23) uses assembly
        - INLINE ASM (node_modules/@abacus-network/core/contracts/libs/TypeCasts.sol#17-22)
AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#174-194) uses assemb
ly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#186-189)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#201-221) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#213-216)
Diamond.fallback() (contracts/gateway/Diamond.sol#42-73) uses assembly
        - INLINE ASM (contracts/gateway/Diamond.sol#46-48)
        - INLINE ASM (contracts/gateway/Diamond.sol#57-72)
DiamondLoupeFacet.facets() (contracts/gateway/facets/DiamondLoupeFacet.sol#26-81) uses assembly
        - INLINE ASM (contracts/gateway/facets/DiamondLoupeFacet.sol#73-75)
        - INLINE ASM (contracts/gateway/facets/DiamondLoupeFacet.sol#78-80)
DiamondLoupeFacet.facetFunctionSelectors(address) (contracts/gateway/facets/DiamondLoupeFacet.sol#86-115) uses assembly
        - INLINE ASM (contracts/gateway/facets/DiamondLoupeFacet.sol#112-114)
DiamondLoupeFacet.facetAddresses() (contracts/gateway/facets/DiamondLoupeFacet.sol#119-161) uses assembly
        - INLINE ASM (contracts/gateway/facets/DiamondLoupeFacet.sol#158-160)
LibDiamond._diamondStorage() (contracts/libraries/LibDiamond.sol#83-92) uses assembly
        - INLINE ASM (contracts/libraries/LibDiamond.sol#89-91)
LibDiamond._initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#303-326) uses assembly
        - INLINE ASM (contracts/libraries/LibDiamond.sol#318-321)
LibDiamond._enforceHasContractCode(address,string) (contracts/libraries/LibDiamond.sol#328-339) uses assembly
        - INLINE ASM (contracts/libraries/LibDiamond.sol#333-335)
CCMPReceiverBase._ccmpMsgOrigin() (contracts/receiver/CCMPReceiverBase.sol#16-35) uses assembly
        - INLINE ASM (contracts/receiver/CCMPReceiverBase.sol#31-34)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Different versions of Solidity are used:
        - Version used: ['0.8.16', '>=0.6.11', '^0.8.0', '^0.8.1', '^0.8.16', '^0.8.2']
        - >=0.6.11 (node_modules/@abacus-network/core/contracts/libs/TypeCasts.sol#2)
        - >=0.6.11 (node_modules/@abacus-network/core/interfaces/IAbacusConnectionManager.sol#2)
        - >=0.6.11 (node_modules/@abacus-network/core/interfaces/IInterchainGasPaymaster.sol#2)
        - >=0.6.11 (node_modules/@abacus-network/core/interfaces/IMailbox.sol#2)
        - >=0.6.11 (node_modules/@abacus-network/core/interfaces/IMessageRecipient.sol#2)
        - >=0.6.11 (node_modules/@abacus-network/core/interfaces/IOutbox.sol#2)
        - ^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/security/Pausable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4)
        - ^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
        - 0.8.16 (contracts/CCMPExecutor.sol#2)
        - 0.8.16 (contracts/adaptors/AbacusAdapter.sol#2)
        - 0.8.16 (contracts/adaptors/AxelarAdaptor.sol#2)
        - 0.8.16 (contracts/adaptors/WormholeAdaptor.sol#2)
        - >=0.6.11 (contracts/adaptors/base/AbacusConnectionClient.sol#2)
        - 0.8.16 (contracts/adaptors/base/CCMPAdaptorBase.sol#2)
        - 0.8.16 (contracts/gateway/Diamond.sol#2)
        - 0.8.16 (contracts/gateway/facets/CCMPConfigurationFacet.sol#2)
        - 0.8.16 (contracts/gateway/facets/CCMPReceiveMessageFacet.sol#4)
        - 0.8.16 (contracts/gateway/facets/CCMPSendMessageFacet.sol#2)
        - 0.8.16 (contracts/gateway/facets/DiamondCutFacet.sol#2)
        - ^0.8.0 (contracts/gateway/facets/DiamondInit.sol#2)
        - ^0.8.0 (contracts/gateway/facets/DiamondLoupeFacet.sol#2)
        - ^0.8.0 (contracts/interfaces/IAxelarGateway.sol#3)
        - 0.8.16 (contracts/interfaces/ICCMPExecutor.sol#2)
        - ^0.8.0 (contracts/interfaces/ICCMPGateway.sol#2)
        - 0.8.16 (contracts/interfaces/ICCMPRouterAdaptor.sol#2)
        - ^0.8.0 (contracts/interfaces/IDiamond.sol#2)
        - ^0.8.0 (contracts/interfaces/IDiamondCut.sol#2)
        - ^0.8.0 (contracts/interfaces/IDiamondLoupe.sol#2)
        - ^0.8.0 (contracts/interfaces/IERC165.sol#2)
        - ^0.8.0 (contracts/interfaces/IERC173.sol#2)
        - ^0.8.0 (contracts/interfaces/IWormhole.sol#4)
        - ^0.8.16 (contracts/libraries/LibDiamond.sol#2)
        - ^0.8.0 (contracts/mock/MockWormhole.sol#4)
        - 0.8.16 (contracts/receiver/CCMPReceiver.sol#2)
        - 0.8.16 (contracts/receiver/CCMPReceiverBase.sol#2)
        - 0.8.16 (contracts/receiver/CCMPReceiverUpgradeable.sol#2)
        - 0.8.16 (contracts/security/Pausable.sol#3)
        - 0.8.16 (contracts/structures/Constants.sol#2)
        - ^0.8.0 (contracts/structures/CrossChainMessage.sol#2)
        - ^0.8.0 (contracts/structures/Wormhole.sol#4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

AbacusConnectionClient._localDomain() (contracts/adaptors/base/AbacusConnectionClient.sol#135-137) is never used and should be removed
CCMPReceiverBase._ccmpMsgOrigin() (contracts/receiver/CCMPReceiverBase.sol#16-35) is never used and should be removed
CCMPReceiverBase._setCCMPExecutor(address) (contracts/receiver/CCMPReceiverBase.sol#11-14) is never used and should be removed
CCMPReceiverUpgradeable.__CCMPReceiver_init(address) (contracts/receiver/CCMPReceiverUpgradeable.sol#13-18) is never used and should be removed
LibDiamond._contractPauser() (contracts/libraries/LibDiamond.sol#120-122) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/Pausable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version0.8.16 (contracts/CCMPExecutor.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.16 (contracts/adaptors/AbacusAdapter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.16 (contracts/adaptors/AxelarAdaptor.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.16 (contracts/adaptors/WormholeAdaptor.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.
7
Pragma version0.8.16 (contracts/adaptors/base/CCMPAdaptorBase.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
/0.8.7
Pragma version0.8.16 (contracts/gateway/Diamond.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.16 (contracts/gateway/facets/CCMPConfigurationFacet.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.
12/0.7.6/0.8.7
Pragma version0.8.16 (contracts/gateway/facets/CCMPReceiveMessageFacet.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6
.12/0.7.6/0.8.7
Pragma version0.8.16 (contracts/gateway/facets/CCMPSendMessageFacet.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12
/0.7.6/0.8.7
Pragma version0.8.16 (contracts/gateway/facets/DiamondCutFacet.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.
6/0.8.7
Pragma version^0.8.0 (contracts/gateway/facets/DiamondInit.sol#2) allows old versions
Pragma version^0.8.0 (contracts/gateway/facets/DiamondLoupeFacet.sol#2) allows old versions
Pragma version^0.8.0 (contracts/interfaces/IAxelarGateway.sol#3) allows old versions
Pragma version0.8.16 (contracts/interfaces/ICCMPExecutor.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.
7
Pragma version^0.8.0 (contracts/interfaces/ICCMPGateway.sol#2) allows old versions
Pragma version0.8.16 (contracts/interfaces/ICCMPRouterAdaptor.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
/0.8.7
Pragma version^0.8.0 (contracts/interfaces/IDiamond.sol#2) allows old versions
Pragma version^0.8.0 (contracts/interfaces/IDiamondCut.sol#2) allows old versions
Pragma version^0.8.0 (contracts/interfaces/IDiamondLoupe.sol#2) allows old versions
Pragma version^0.8.0 (contracts/interfaces/IERC165.sol#2) allows old versions
Pragma version^0.8.0 (contracts/interfaces/IERC173.sol#2) allows old versions
Pragma version^0.8.0 (contracts/interfaces/IWormhole.sol#4) allows old versions
Pragma version^0.8.16 (contracts/libraries/LibDiamond.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.0 (contracts/mock/MockWormhole.sol#4) allows old versions
Pragma version0.8.16 (contracts/receiver/CCMPReceiver.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.16 (contracts/receiver/CCMPReceiverBase.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8
.7
Pragma version0.8.16 (contracts/receiver/CCMPReceiverUpgradeable.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.
7.6/0.8.7
Pragma version0.8.16 (contracts/security/Pausable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.16 (contracts/structures/Constants.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.0 (contracts/structures/CrossChainMessage.sol#2) allows old versions
Pragma version^0.8.0 (contracts/structures/Wormhole.sol#4) allows old versions
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#63)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUp
gradeable.sol#128-139):
```

```
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#137)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.s
ol#157-166):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#164)
Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#60-65):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#63)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#128-139):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#157-166):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#164)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#184-193):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
Low level call in CCMPExecutor.execute(address,bytes) (contracts/CCMPExecutor.sol#29-35):
        - (success,returndata) = _to.call{gas: gasleft()()}(_calldata) (contracts/CCMPExecutor.sol#34)
Low level call in CCMPSendMessageFacet._handleFee(CCMPMessage) (contracts/gateway/facets/CCMPSendMessageFacet.sol#92-119):
        - (success,returndata) = relayer.call{value: msg.value}() (contracts/gateway/facets/CCMPSendMessageFacet.sol#102-104)
Low level call in LibDiamond._initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#303-326):
        - (success,error) = _init.delegatecall(_calldata) (contracts/libraries/LibDiamond.sol#313)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter TypeCasts.coerceString(bytes32)._buf (node_modules/@abacus-network/core/contracts/libs/TypeCasts.sol#6) is not in mixedCase
Parameter TypeCasts.addressToBytes32(address)._addr (node_modules/@abacus-network/core/contracts/libs/TypeCasts.sol#26) is not in mixedCase
Parameter TypeCasts.bytes32ToAddress(bytes32)._buf (node_modules/@abacus-network/core/contracts/libs/TypeCasts.sol#31) is not in mixedCase
Function IERC20Permit.DOMAIN_SEPARATOR() (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol#59) is not in mixedCase
Parameter CCMPExecutor.execute(address,bytes)._to (contracts/CCMPExecutor.sol#29) is not in mixedCase
Parameter CCMPExecutor.execute(address,bytes)._calldata (contracts/CCMPExecutor.sol#29) is not in mixedCase
Parameter CCMPExecutor.updateCCMPGateway(address)._newCCMPGateway (contracts/CCMPExecutor.sol#37) is not in mixedCase
Parameter AbacusAdapter.handle(uint32,bytes32,bytes)._origin (contracts/adaptors/AbacusAdapter.sol#87) is not in mixedCase
Parameter AbacusAdapter.handle(uint32,bytes32,bytes)._sender (contracts/adaptors/AbacusAdapter.sol#88) is not in mixedCase
Parameter AbacusAdapter.handle(uint32,bytes32,bytes)._message (contracts/adaptors/AbacusAdapter.sol#89) is not in mixedCase
Parameter AbacusAdapter.routePayload(CCMPMessage,bytes)._message (contracts/adaptors/AbacusAdapter.sol#116) is not in mixedCase
Parameter AbacusAdapter.verifyPayload(CCMPMessage,bytes)._ccmpMessage (contracts/adaptors/AbacusAdapter.sol#155) is not in mixedCase
Parameter AbacusAdapter.updateDomainId(uint256,uint32)._chainId (contracts/adaptors/AbacusAdapter.sol#173) is not in mixedCase
Parameter AbacusAdapter.updateDomainId(uint256,uint32)._domainId (contracts/adaptors/AbacusAdapter.sol#173) is not in mixedCase
Parameter AbacusAdapter.setAbacusAdaptor(uint256,address)._chainId (contracts/adaptors/AbacusAdapter.sol#180) is not in mixedCase
Parameter AbacusAdapter.setAbacusAdaptor(uint256,address)._abacusAdaptor (contracts/adaptors/AbacusAdapter.sol#180) is not in mixedCase
Parameter AxelarAdaptor.routePayload(CCMPMessage,bytes)._message (contracts/adaptors/AxelarAdaptor.sol#73) is not in mixedCase
Parameter AxelarAdaptor.verifyPayload(CCMPMessage,bytes)._ccmpMessage (contracts/adaptors/AxelarAdaptor.sol#109) is not in mixedCase
Parameter AxelarAdaptor.execute(bytes32,string,string,bytes)._commandId (contracts/adaptors/AxelarAdaptor.sol#127) is not in mixedCase
Parameter AxelarAdaptor.execute(bytes32,string,string,bytes)._sourceChain (contracts/adaptors/AxelarAdaptor.sol#128) is not in mixedCase
Parameter AxelarAdaptor.execute(bytes32,string,string,bytes)._sourceAddress (contracts/adaptors/AxelarAdaptor.sol#129) is not in mixedCase
Parameter AxelarAdaptor.execute(bytes32,string,string,bytes)._payload (contracts/adaptors/AxelarAdaptor.sol#130) is not in mixedCase
Parameter AxelarAdaptor.setChainIdToName(uint256,string)._chainId (contracts/adaptors/AxelarAdaptor.sol#171) is not in mixedCase
Parameter AxelarAdaptor.setChainIdToName(uint256,string)._chainName (contracts/adaptors/AxelarAdaptor.sol#171) is not in mixedCase
Parameter AxelarAdaptor.setAxelarAdaptorAddressChecksummed(string,string)._chainName (contracts/adaptors/AxelarAdaptor.sol#180) is not in mixedCase
Parameter AxelarAdaptor.setAxelarAdaptorAddressChecksummed(string,string)._adaptorAddressChecksummed (contracts/adaptors/AxelarAdaptor.sol#181) is not in
mixedCase
Parameter AxelarAdaptor.updateAxelarGateway(IAxelarGateway)._axelarGateway (contracts/adaptors/AxelarAdaptor.sol#192) is not in mixedCase
Parameter WormholeAdaptor.setWormholeAdaptorAddress(uint256,address)._chainId (contracts/adaptors/WormholeAdaptor.sol#79) is not in mixedCase
Parameter WormholeAdaptor.setWormholeAdaptorAddress(uint256,address)._adaptor (contracts/adaptors/WormholeAdaptor.sol#79) is not in mixedCase
Parameter WormholeAdaptor.updateWormhole(IWormhole)._wormhole (contracts/adaptors/WormholeAdaptor.sol#87) is not in mixedCase
Parameter WormholeAdaptor.updateWormholeChainId(uint16,uint256)._wormholeChainId (contracts/adaptors/WormholeAdaptor.sol#99) is not in mixedCase
Parameter WormholeAdaptor.updateWormholeChainId(uint16,uint256)._chainId (contracts/adaptors/WormholeAdaptor.sol#99) is not in mixedCase
Parameter WormholeAdaptor.routePayload(CCMPMessage,bytes)._message (contracts/adaptors/WormholeAdaptor.sol#110) is not in mixedCase
Parameter WormholeAdaptor.routePayload(CCMPMessage,bytes)._routerArgs (contracts/adaptors/WormholeAdaptor.sol#111) is not in mixedCase
Parameter WormholeAdaptor.verifyPayload(CCMPMessage,bytes)._ccmpMessage (contracts/adaptors/WormholeAdaptor.sol#132) is not in mixedCase
Parameter WormholeAdaptor.verifyPayload(CCMPMessage,bytes)._verificationData (contracts/adaptors/WormholeAdaptor.sol#133) is not in mixedCase
Function AbacusConnectionClient.__AbacusConnectionClient_initialize(address) (contracts/adaptors/base/AbacusConnectionClient.sol#47-51) is not in mixedCas
e
Parameter AbacusConnectionClient.__AbacusConnectionClient_initialize(address)._abacusConnectionManager (contracts/adaptors/base/AbacusConnectionClient.sol
#48) is not in mixedCase
Parameter AbacusConnectionClient.setAbacusConnectionManager(address)._abacusConnectionManager (contracts/adaptors/base/AbacusConnectionClient.sol#67) is n
ot in mixedCase
Parameter AbacusConnectionClient.setInterchainGasPaymaster(address)._interchainGasPaymaster (contracts/adaptors/base/AbacusConnectionClient.sol#79) is not
 in mixedCase
Parameter CCMPAdaptorBase.setCCMPGateway(ICCMPGateway)._ccmpGateway (contracts/adaptors/base/CCMPAdaptorBase.sol#39) is not in mixedCase
Parameter CCMPConfigurationFacet.transferOwnership(address)._newOwner (contracts/gateway/facets/CCMPConfigurationFacet.sol#14) is not in mixedCase
Parameter CCMPConfigurationFacet.setGateway(uint256,ICCMPGateway)._chainId (contracts/gateway/facets/CCMPConfigurationFacet.sol#31) is not in mixedCase
Parameter CCMPConfigurationFacet.setGateway(uint256,ICCMPGateway)._gateway (contracts/gateway/facets/CCMPConfigurationFacet.sol#31) is not in mixedCase
Parameter CCMPConfigurationFacet.gateway(uint256)._chainId (contracts/gateway/facets/CCMPConfigurationFacet.sol#37) is not in mixedCase
Parameter CCMPConfigurationFacet.setCCMPExecutor(ICCMPExecutor)._ccmpExecutor (contracts/gateway/facets/CCMPConfigurationFacet.sol#61) is not in mixedCase
Parameter CCMPConfigurationFacet.setPauser(address)._pauser (contracts/gateway/facets/CCMPConfigurationFacet.sol#75) is not in mixedCase
Parameter CCMPReceiverMessageFacet.receiveMessage(CCMPMessage,bytes,bool)._message (contracts/gateway/facets/CCMPReceiveMessageFacet.sol#25) is not in mix
edCase
Parameter CCMPSendMessageFacet.sendMessage(uint256,string,CCMPMessagePayload[],GasFeePaymentArgs,bytes)._destinationChainId (contracts/gateway/facets/CCMP
SendMessageFacet.sol#26) is not in mixedCase
Parameter CCMPSendMessageFacet.sendMessage(uint256,string,CCMPMessagePayload[],GasFeePaymentArgs,bytes)._adaptorName (contracts/gateway/facets/CCMPSendMes
sageFacet.sol#27) is not in mixedCase
Parameter CCMPSendMessageFacet.sendMessage(uint256,string,CCMPMessagePayload[],GasFeePaymentArgs,bytes)._payloads (contracts/gateway/facets/CCMPSendMessag
eFacet.sol#28) is not in mixedCase
Parameter CCMPSendMessageFacet.sendMessage(uint256,string,CCMPMessagePayload[],GasFeePaymentArgs,bytes)._gasFeePaymentArgs (contracts/gateway/facets/CCMPS
endMessageFacet.sol#29) is not in mixedCase
Parameter CCMPSendMessageFacet.sendMessage(uint256,string,CCMPMessagePayload[],GasFeePaymentArgs,bytes)._routerArgs (contracts/gateway/facets/CCMPSendMess
ageFacet.sol#30) is not in mixedCase
Parameter DiamondCutFacet.diamondCut(IDiamond.FacetCut[],address,bytes)._diamondCut (contracts/gateway/facets/DiamondCutFacet.sol#23) is not in mixedCase
Parameter DiamondCutFacet.diamondCut(IDiamond.FacetCut[],address,bytes)._init (contracts/gateway/facets/DiamondCutFacet.sol#24) is not in mixedCase
Parameter DiamondCutFacet.diamondCut(IDiamond.FacetCut[],address,bytes)._calldata (contracts/gateway/facets/DiamondCutFacet.sol#25) is not in mixedCase
Parameter DiamondLoupeFacet.facetFunctionSelectors(address)._facet (contracts/gateway/facets/DiamondLoupeFacet.sol#86) is not in mixedCase
Parameter DiamondLoupeFacet.facetAddress(bytes4)._functionSelector (contracts/gateway/facets/DiamondLoupeFacet.sol#167) is not in mixedCase
Parameter DiamondLoupeFacet.supportsInterface(bytes4)._interfaceId (contracts/gateway/facets/DiamondLoupeFacet.sol#180) is not in mixedCase
Variable MockWormhole._validationState (contracts/mock/MockWormhole.sol#11) is not in mixedCase
Variable MockWormhole._payload (contracts/mock/MockWormhole.sol#12) is not in mixedCase
Variable CCMPReceiverBase._ccmpExecutor (contracts/receiver/CCMPReceiverBase.sol#9) is not in mixedCase
Function CCMPReceiverUpgradeable.__CCMPReceiver_init(address) (contracts/receiver/CCMPReceiverUpgradeable.sol#13-18) is not in mixedCase
Parameter CCMPReceiverUpgradeable.__CCMPReceiver_init(address).__ccmpExecutor (contracts/receiver/CCMPReceiverUpgradeable.sol#13) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable DiamondLoupeFacet.facetFunctionSelectors(address)._facetFunctionSelectors (contracts/gateway/facets/DiamondLoupeFacet.sol#90) is too similar to I
DiamondLoupe.facetFunctionSelectors(address).facetFunctionSelectors_ (contracts/interfaces/IDiamondLoupe.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

CCMPReceiverMessageFacet (contracts/gateway/facets/CCMPReceiveMessageFacet.sol#16-130) does not implement functions:
        - CCMPReceiverMessageFacet._executeCCMPMessage(CCMPMessage,bool) (contracts/gateway/facets/CCMPReceiveMessageFacet.sol#96-129)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

Constants.NATIVE_ADDRESS (contracts/structures/Constants.sol#5-6) is never used in CCMPReceiverMessageFacet (contracts/gateway/facets/CCMPReceiveMessageFa
cet.sol#16-130)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#61-63)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#69-72)
setValidationState(bool) should be declared external:
```

```
setPayload(bytes) should be declared external:
        - MockWormhole.setPayload(bytes) (contracts/mock/MockWormhole.sol#27-29)
changePauser(address) should be declared external:
        - PausableBase.changePauser(address) (contracts/security/Pausable.sol#58-60)
pause() should be declared external:
        - PausableBase.pause() (contracts/security/Pausable.sol#77-79)
unpause() should be declared external:
        - PausableBase.unpause() (contracts/security/Pausable.sol#81-83)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

- No major issues found by Slither.

THANK YOU FOR CHOOSING

// HALBORN