Competitive Security Assessment

Mitosis

Mar 17th, 2024

Secure3

secure3.io

Summary

This report is prepared for the project to identify vulnerabilities and issues in the smart contract source code. A group of NDA covered experienced security experts have participated in the Secure3′s Audit Contest to find vulnerabilities and optimizations. Secure3 team has participated in the contest process as well to provide extra auditing coverage and scrutiny of the finding submissions.

The comprehensive examination and auditing scope includes:

• Cross checking contract implementation against functionalities described in the documents and white paper disclosed by the project owner.

• Contract Privilege Role Review to provide more clarity on smart contract roles and privilege.

• Using static analysis tools to analyze smart contracts against common known vulnerabilities patterns.

• Verify the code base is compliant with the most up-to-date industry standards and security best practices.

• Comprehensive line-by-line manual code review of the entire codebase by industry experts.

The security assessment resulted in findings that are categorized in four severity levels: Critical, Medium, Low, Informational. For each of the findings, the report has included recommendations of fix or mitigation for security and best practices.

Overview

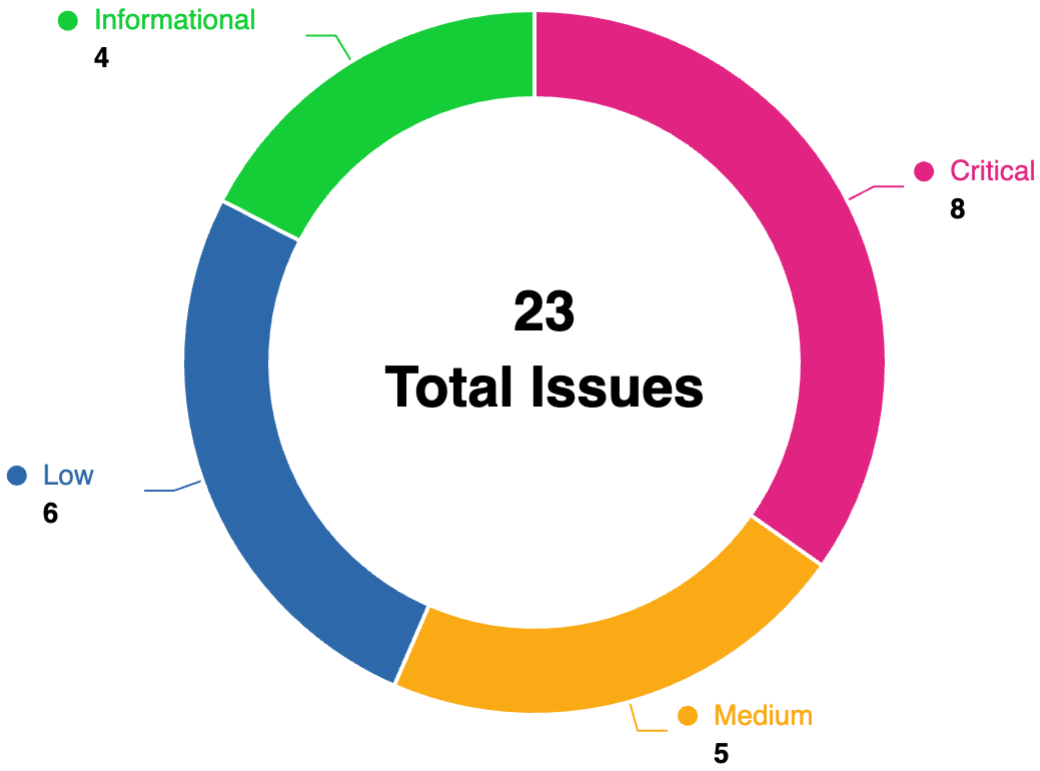
Project Name	Mitosis
Language	Solidity
Codebase	• https://github.com/mitosis-org/evm • audit version - 7b06d163b7ee2ec41ca0dec704d5411b9b16cc41 • final version - 4ac3e65cadcbb4a0fee9b1a6ee6a016a2d73e432
Audit Methodology	• Audit Contest • Business Logic and Code Review • Privileged Roles Review • Static Analysis

Audit Scope

File	SHA256 Hash
src/vault/BasicVault.sol	91dad4a0783b5bdee19b7b928af12ebd8621acfeede1c718871913622a4fa7c8
src/vault/Cap.sol	2cae110325f2f47ac4d730e1f1013406ceee5ae23ebecf40ac3cd18bafc4578f
src/helpers/ccdm/CCDMClient.sol	966592c5fc7b7cdb86a916b6c5627bb38b9d263d2aec1f947cc9bdb159ec8f64
src/vault/BasicVaultFactory.sol	9a58451f37d740802265d3e9193883bb966ac90fccdc39a49653aa07d6674ce2
src/helpers/ccdm/CCDMHost.sol	15a9909ba92460a7be6c856d15df93e8641e89632d246ca75da31508317d36c3
src/helpers/ccdm/Message.sol	f5253db06015172e246981cafe017f186a3c6436f18e5da7ec27e7777e083b4b
src/interfaces/bridge/IPolygonZkEvmBridge.sol	9e1b0a5abea72d4b887fc46ac2aa79140bda2ab0542e311d0fc63142bdc47e91
src/helpers/EETHDepositHelper.sol	ecedef2b16e86b5e70c9f6875e15fbd20f02d86657ae583d07b4e7f78314c8b8
src/vault/VaultHub.sol	f85d000130c25153eb1280c32d0601b1fba7a9e492a5a81e787b0157cd16d9d2
src/interfaces/EtherFi.sol	b61551446c5703d2d18ca2a34d8e7342856da5210ecd361ae8df1e57c5f39037
src/interfaces/IVault.sol	2ad64ee70b1505073c80dc58e57c28705a6950fca0dee130bc7eb877b7a07239
src/helpers/adapter/ArbitrumBridgeAdapter.sol	32be4f440d744b266af63eb7c17d2f0cfefca5da751afd78a059c60ab7f1dc36
src/lib/Error.sol	08df42edef8fd9185b6adf5e373ca02d0cdbf37b8ef710c869ed59345695cea1
src/helpers/adapter/PolygonZkEvmBridgeAdapter.sol	0374f58a4c93f4e5820a1dec11a2eb31dd99b66e2116c819aaf5674a00de883a
src/helpers/adapter/OptimismBridgeAdapter.sol	82fa7239d0c04b98e2cce1df3e80a9fc2c5975e184899ab50b26c03932575a2d
src/interfaces/ICap.sol	489e7ddd3e164442afb51eff097724cd22b7c0252c292f82eb2275c94e5f265b
src/interfaces/bridge/IArbitrumGateway.sol	7c09459b5c76ab227bc81ba09683b2acb98a21a76ba7d9b5b9ece418ccca457b
src/interfaces/bridge/IOptimismGateway.sol	712d183cb6c89d452557114b00d3ed20fbec3e8aba7cdc9d85a5aecc7db5d344
src/interfaces/KelpDAO.sol	9f5f8dbdb673c7bb243295402051b9d71e670582b4f522972b557a1734b522d5

src/lib/Conv.sol	7cd6a1b80465254582671924189674 8d2f1cdd2aed4a50079c7ef253ffaa087c
src/entrypoint/Entrypoint.sol	c71f132abd7ba14d58f7556237f30b6322c49c495715f673e2d46e46554b53d8
src/interfaces/Renzo.sol	bf48eef29bfc622b50091cf8a6d2333c0ed519470e36b444dacc95aea9755c9d
src/interfaces/IVaultFactory.sol	5c0e27db5bc0b31de26c7e392d36494a5fcef49dd809b88e3896752aaf3d3622
src/interfaces/bridge/IBridgeAdapter.sol	0697ebefe40bee1d7032e329bf66705d229013767d2ee9f1ea85a0c6b869701c

Code Assessment Findings



ID	Name	Category	Severity	Client Response	Contributor
MTS-1	`encodeRefund()` and `encodeBridge()` function in `Message` contract have wrong logic leads to wrong behavior	Logical	Critical	Fixed	0xhuy0512, ravikiran_web3
MTS-2	`ArbitrumBridgeAdapter::outboundTransfer()` and `PolygonZkEvmBridgeAdapter::bridgeAsset()` are `payable` function but has no mechanism to forward `msg.value` resulting in Denial of Service(DoS)	DOS	Critical	Fixed	0xzoobi
MTS-3	OpenZeppelin's `SafeERC20.safePermit()` is not follow EIP-712	Signature Forgery or Replay	Critical	Fixed	0xhuy0512, ravikiran_web3
MTS-4	Missing control in bridgeAsset function	Privilege Related	Critical	Fixed	ravikiran_web3

MTS-5	EETHDepositHelper contract deposit wrong amount, causing can't deposit forever	Logical	Critical	Fixed	0xhuy0512, 0xzoobi
MTS-6	Cap.sol contract can't send crosschain using Hyperlane because it doesn't hold any native token as crosschain fee	Logical	Critical	Fixed	0xhuy0512
MTS-7	CCDM Host's token approval to wrong receiver	Logical	Critical	Fixed	0xhuy0512
MTS-8	Bridging token using native bridge send to wrong receiver	Logical	Critical	Fixed	0xhuy0512
MTS-9	User who hold vault token by trading can't redeem token to earn back themself, make the vault token useless	Logical	Medium	Mitigated	0xhuy0512
MTS-10	Unsupported Opcode in Multi-Chain Deployment	Language Specific	Medium	Fixed	ret2basic
MTS-11	Underpayaing the L2 gas may lead to loss of funds	DOS	Medium	Fixed	0xzoobi
MTS-12	The epoch assertion in `_checkRemoteStateAndAdvance()` function is not behave as expected	Logical	Medium	Fixed	0xhuy0512
MTS-13	Not pulling token to Adapter contracts lead to to can't bridge token crosschain using native bridge	Logical	Medium	Fixed	0xhuy0512
MTS-14	zkEVM bridge need an extra step in L2 to successfully bridged	Logical	Low	Acknowledged	0xhuy0512
MTS-15	Use `Ownable2StepUpgradeable` instead of `OwnableUpgradeable` contract	Privilege Related	Low	Fixed	Meliclit, 0xzoobi
MTS-16	The `bridgeGateway` addresses are hardcoded	DOS	Low	Fixed	0xzoobi

MTS-17	No whitelist token in CCDM Host lead to user can accidentally losing token	Logical	Low	Fixed	0xhuy0512
MTS-18	Missing 0 amount check	Logical	Low	Fixed	ravikiran_web3
MTS-19	Ensure no native asset value is sent in `payable` method that can handle ERC20 transfers as well	Logical	Low	Fixed	0xzoobi
MTS-20	Typo error	Code Style	Informational	Fixed	ravikiran_web3
MTS-21	Open TODOs	Code Style	Informational	Fixed	Meliclit, ravikiran_web3
MTS-22	Gas Optimization	Gas Optimization	Informational	Fixed	0xhuy0512
MTS-23	ArbitrumBridgeAdapter::bridgeAsset() ignores the seqNumber returned by underlying bridge on outbound transfer	Code Style	Informational	Fixed	ravikiran_web3

MTS-1: `encodeRefund()` and `encodeBridge()` function in `Message` contract have wrong logic leads to wrong behavior

Category	Severity	Client Response	Contributor
Logical	Critical	Fixed	0xhuy0512, ravikiran_web3

Code Reference

- code/src/helpers/ccdm/Message.sol#L86
- code/src/helpers/ccdm/Message.sol#L107-L109
- code/src/helpers/ccdm/Message.sol#L108

```
86: return abi.encodePacked(MsgType.Deposit, msg_.receiver, msg_.token, msg_.amount);
```

```
107: function encodeBridge(MsgBridge memory msg_) internal pure returns (bytes memory) {
108:         return abi.encodePacked(MsgType.Deposit, msg_.receiver, msg_.token, msg_.amount);
109:     }
```

```
108: return abi.encodePacked(MsgType.Deposit, msg_.receiver, msg_.token, msg_.amount);
```

Description

0xhuy0512: 2 functions **`encodeRefund()`** and **`encodeBridge()`** both use **`MsgType.Deposit`** to encode the message which is wrong. And because **`MsgDeposit`**, **`MsgRefund`**, **`MsgBridge`** structs is compatible to each other (if not exactly the same), the protocol will mistaken refund and bridge requests to deposit requests without reverting the transaction, breaking core functionality

```
    function encodeRefund(MsgRefund memory msg_) internal pure returns (bytes memory) {
        return abi.encodePacked(MsgType.Deposit, msg_.receiver, msg_.token, msg_.amount);
    }

    function encodeBridge(MsgBridge memory msg_) internal pure returns (bytes memory) {
        return abi.encodePacked(MsgType.Deposit, msg_.receiver, msg_.token, msg_.amount);
    }
```

ravikiran_web3: The **encodeBridge()** helper function in ccmd::message library uses an incorrect message type for encoding the bridge message packet.
The functions for encoding and decoding are segregated for message types.
The message type is critical on how it will be handled and hence important to use the correct message type.

```
enum MsgType {
    Deposit,
    Refund,
    Bridge
}
```

Refer to the code below where the first parameter to abi.encodePacked should be **MsgType.Bridge** instead of ~~MsgType.Deposit~~

```
  function encodeBridge(MsgBridge memory msg_) internal pure returns (bytes memory) {
        return abi.encodePacked(MsgType.Deposit, msg_.receiver, msg_.token, msg_.amount);
    }
```

Recommendation

0xhuy0512:

```
    function encodeRefund(MsgRefund memory msg_) internal pure returns (bytes memory) {
-       return abi.encodePacked(MsgType.Deposit, msg_.receiver, msg_.token, msg_.amount);
+       return abi.encodePacked(MsgType.Refund, msg_.receiver, msg_.token, msg_.amount);
    }

    function encodeBridge(MsgBridge memory msg_) internal pure returns (bytes memory) {
-      return abi.encodePacked(MsgType.Deposit, msg_.receiver, msg_.token, msg_.amount);
+       return abi.encodePacked(MsgType.Bridge, msg_.receiver, msg_.token, msg_.amount);

    }
```

ravikiran_web3: Revise the encodeBridge function as below.

```
  function encodeBridge(MsgBridge memory msg_) internal pure returns (bytes memory) {
        return abi.encodePacked(MsgType.Bridge, msg_.receiver, msg_.token, msg_.amount);
    }


function encodeRefund(MsgRefund memory msg_) internal pure returns (bytes memory) {
        return abi.encodePacked(MsgType.Refund, msg_.receiver, msg_.token, msg_.amount);
    }
```

Client Response

0xhuy0512: Fixed.
https://github.com/mitosis-org/evm/blob/9dedfb6b742b71af5460789ea8735695a454feb4/src/helpers/ccdm/Message.sol#L92
https://github.com/mitosis-org/evm/blob/9dedfb6b742b71af5460789ea8735695a454feb4/src/helpers/ccdm/Message.sol#L114
ravikiran_web3: Fixed.
https://github.com/mitosis-org/evm/blob/9dedfb6b742b71af5460789ea8735695a454feb4/src/helpers/ccdm/Message.sol#L92
https://github.com/mitosis-org/evm/blob/9dedfb6b742b71af5460789ea8735695a454feb4/src/helpers/ccdm/Message.sol#L114

MTS-2: `ArbitrumBridgeAdapter::outboundTransfer()` and `PolygonZkEvmBridgeAdapter::bridgeAsset()` are `payable` function but has no mechanism to forward `msg.value` resulting in Denial of Service(DoS)

Category	Severity	Client Response	Contributor
DOS	Critical	Fixed	0xzoobi

Code Reference

- code/src/helpers/adapter/ArbitrumBridgeAdapter.sol#L33-L35

```
33: function bridgeAsset(address destAddr, address l1Asset, address, uint256 amount) external {
34:         bridge.outboundTransfer(l1Asset, destAddr, amount, maxGas, gasPriceBid, "");
35:     }
```

- code/src/helpers/adapter/PolygonZkEvmBridgeAdapter.sol#L23-L25

```
23: function bridgeAsset(address destAddr, address l1Asset, address, uint256 amount) external {
24:         bridge.bridgeAsset(destNetwork, destAddr, amount, l1Asset, forceUpdateGlobalExitRoot,
"");
25:     }
```

- code/src/interfaces/bridge/IArbitrumGateway.sol#L6-L13

```
6: function outboundTransfer(
7:         address _token,
8:         address _to,
9:         uint256 _amount,
10:         uint256 _maxGas,
11:         uint256 _gasPriceBid,
12:         bytes calldata _data
13:     ) external payable returns (bytes memory);
```

- code/src/interfaces/bridge/IPolygonZkEvmBridge.sol#L71-L78

```
71: function bridgeAsset(
72:         uint32 destinationNetwork,
73:         address destinationAddress,
74:         uint256 amount,
75:         address token,
76:         bool forceUpdateGlobalExitRoot,
77:         bytes calldata permitData
78:     ) external payable;
```

Description

0xzoobi: If the `outboundTransfer` function is declared as `payable` in the interface, it signifies its ability to receive and manage `msg.value`. However, it appears that the present implementation of `bridgeAsset` fails to consider this, resulting in the inability to forward any `msg.value`.
To resolve this issue, it is essential to ensure that the implementation of `outboundTransfer` in the `BridgeAsset` contract appropriately handles the incoming `msg.value` and forwards it as needed.
In the current implementation, all calls made to `ArbitrumBridgeAdapter::bridgeAsset()` will revert because the existing code lacks the capability to transfer `msg.value`. A similar issue is also present in the `PolygonZkEvmBridgeAdapter::bridgeAsset()`.

Proof of Concept - Run via REMIX IDE

Steps

1. Deploy `SimpleWallet`
2. Deploy `SimpleWalletWrapper` by passing address of `SimpleWallet` in the constructor
3. First call the `depositHelper` function, the call will revert.
4. Now, call the `depositHelperOne`, which is the fix.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract SimpleWalletWrapper {

    SimpleWallet public testContract;
    constructor(address _simpleWallet) {
        testContract = SimpleWallet(_simpleWallet);
    }

    //The Current Issue
    function depositHelper() external {
        testContract.depositIT();
    }

    //The FIX
    function depositHelperOne() external payable {
        (bool success,) = address(testContract).call{value: msg.value}(abi.encodeWithSignature("de
positIT()"));
        require(success, "External call failed");
    }

}

contract SimpleWallet {
    mapping(address => uint256) public balances;
    event Deposit(address indexed depositor, uint256 amount);

    constructor() {
    }

    function depositIT() public payable {
        require(msg.value > 0, "Deposit amount must be greater than 0");
        balances[msg.sender] += msg.value;
        emit Deposit(msg.sender, msg.value);
    }

    function getContractBalance() external view returns (uint256) {
        return address(this).balance;
    }
}
```

Recommendation

0xzoobi: To resolve the issue, start by making the `bridgeAsset` function payable. Next, when invoking `bridge.outboundTransfer`, forward the received Ether as `msg.value`.

The call to the function will work even if the `msg.value == 0` , since the root cause of the issue is the syntax.
Sample Fix:

```
function bridgeAsset(address destAddr, address l1Asset, uint256 amount) external payable {
   (bool success, bytes memory data) = address(bridge).call{value: msg.value}(
            abi.encodeWithSignature("outboundTransfer(address,uint256,uint256,uint256,bytes)", l1A
sset, destAddr, amount, maxGas, gasPriceBid, "");
        );
        require(success, "External call to stakeNFT failed");
}
```

Client Response

0xzoobi: Fixed,For Arbitrum adapter, fixed via integrating ATM contract.
And for the PolygonZkEvm side, it's not actually accepts it. (ref: https://etherscan.io/tx/0x82e0c34ce8a4ce50c4b29b711163773c34868fe520200a89d5073c46295d5ab9)

MTS-3:OpenZeppelin's `SafeERC20.safePermit()` is not follow EIP-712

Category	Severity	Client Response	Contributor
Signature Forgery or Replay	Critical	Fixed	0xhuy0512, ravikiran_web3

Code Reference

- code/src/helpers/EETHDepositHelper.sol#L44-L47
- code/src/helpers/EETHDepositHelper.sol#L45
- code/src/helpers/EETHDepositHelper.sol#L54-L57
- code/src/helpers/EETHDepositHelper.sol#L55

```
44: function deposit(uint256 amount, address vault, uint256 deadline, uint8 v, bytes32 r, bytes32 s)
external {
45:         SafeERC20.safePermit(_eETH, _msgSender(), address(this), amount, deadline, v, r, s);
46:         _deposit(amount, vault);
47:     }
```

```
45: SafeERC20.safePermit(_eETH, _msgSender(), address(this), amount, deadline, v, r, s);
```

```
54: function redeem(uint256 amount, address vault, uint256 deadline, uint8 v, bytes32 r, bytes32 s)
external {
55:         SafeERC20.safePermit(IERC20Permit(vault), _msgSender(), address(this), amount, deadline,
v, r, s);
56:         _redeem(amount, vault);
57:     }
```

```
55: SafeERC20.safePermit(IERC20Permit(vault), _msgSender(), address(this), amount, deadline, v, r,
s);
```

Description

0xhuy0512: OpenZeppelin's **`SafeERC20.safePermit()`** got deprecated because of this issue: https://www.trust-security.xyz/post/permission-denied

ravikiran_web3: Signatures should be protected against replay attack with a nonce and block.chainid and domain to ensure the same signature cannot be replayed across domains or chains.
But the below functions ETHDepositHelper does not take into account the domain or chainid.

```
function deposit(uint256 amount, address vault, uint256 deadline, uint8 v, bytes32 r, bytes32 s) e
xternal {
        SafeERC20.safePermit(_eETH, _msgSender(), address(this), amount, deadline, v, r, s);
        _deposit(amount, vault);
    }
```

```
function redeem(uint256 amount, address vault, uint256 deadline, uint8 v, bytes32 r, bytes32 s) ex
ternal {
        SafeERC20.safePermit(IERC20Permit(vault), _msgSender(), address(this), amount, deadline,
v, r, s);
        _redeem(amount, vault);
    }
```

Recommendation

0xhuy0512: Consider not use safePermit()
ravikiran_web3: EIP-712 should be followed to ensure the signatures are not replayed.

Client Response

0xhuy0512: Fixed,https://github.com/mitosis-org/evm/pull/45
ravikiran_web3: Fixed,https://github.com/mitosis-org/evm/pull/45

MTS-4:Missing control in bridgeAsset function

Category	Severity	Client Response	Contributor
Privilege Related	Critical	Fixed	ravikiran_web3

Code Reference

- code/src/helpers/adapter/ArbitrumBridgeAdapter.sol#L33-L35

```
33: function bridgeAsset(address destAddr, address l1Asset, address, uint256 amount) external {
34:         bridge.outboundTransfer(l1Asset, destAddr, amount, maxGas, gasPriceBid, "");
35:     }
```

- code/src/helpers/adapter/OptimismBridgeAdapter.sol#L19-L21

```
19: function bridgeAsset(address destAddr, address l1Asset, address l2Asset, uint256 amount) externa
l {
20:         bridge.depositERC20To(l1Asset, l2Asset, destAddr, amount, minGasLimit, "");
21:     }
```

- code/src/helpers/adapter/PolygonZkEvmBridgeAdapter.sol#L23-L25

```
23: function bridgeAsset(address destAddr, address l1Asset, address, uint256 amount) external {
24:         bridge.bridgeAsset(destNetwork, destAddr, amount, l1Asset, forceUpdateGlobalExitRoot,
"");
25:     }
```

Description

ravikiran_web3: bridgeAsset() function is used to send msg to the destination chain via bridge.
This function is being called by Crosschain deposit manager host in the _handle() function. The handle function has all the checks before the message is sent to the bridge.
The vulnerability is that for all the below three adaptors, there is no access restriction for bridgeAsset() function. This means any one can call this function without meeting the other checks implemented in dispatch() and handle() functions.
The cross bridge communication functions should be restrictions with permissions so that only qualified contracts can call after all checking is done. Leave this open ended means, attacker can dispatch messages on one side of the bridge breaking the symmetry of tokens across the bridges. Its means, without send tokens from one chain, there is a possibility to make a deposit on the other chain.

1. ArbitrumBridgeAdapter
2. OptimismBridgeAdapter
3. PolygonZkEvmBridgeAdapter

```
function bridgeAsset(address destAddr, address l1Asset, address, uint256 amount) external {
      bridge.outboundTransfer(l1Asset, destAddr, amount, maxGas, gasPriceBid, "");
  }
```

```
function bridgeAsset(address destAddr, address l1Asset, address l2Asset, uint256 amount) external
{
        bridge.depositERC20To(l1Asset, l2Asset, destAddr, amount, minGasLimit, "");
    }
```

```
 function bridgeAsset(address destAddr, address l1Asset, address, uint256 amount) external {
        bridge.bridgeAsset(destNetwork, destAddr, amount, l1Asset, forceUpdateGlobalExitRoot, "");
    }
```

Hence each of these bridgeAsset() function on adaptors should be restricted to CCDM Host contract.

Recommendation

ravikiran_web3: Restrict the bridgeAsset() so that it can only be called by Crosschain deposit manager host(CCDM Host) contract.

Client Response

ravikiran_web3: Fixed by make theses functions only callable by allowed accounts.

MTS-5:EETHDepositHelper contract deposit wrong amount, causing can't deposit forever

Category	Severity	Client Response	Contributor
Logical	Critical	Fixed	0xhuy0512, 0xzoobi

Code Reference

- code/src/helpers/EETHDepositHelper.sol#L66-L67
- code/src/helpers/EETHDepositHelper.sol#L66-L67

```
66: SafeERC20.safeIncreaseAllowance(_weETH, address(vault), weETHAmount);
67:         IVault(vault).deposit(amount, _msgSender());
```


```
66: SafeERC20.safeIncreaseAllowance(_weETH, address(vault), weETHAmount);
67:         IVault(vault).deposit(amount, _msgSender());
```


Description

0xhuy0512: Function `**_deposit()**` after warping eETH to weETH, it approves to vault with `**weETHAmount**` of weETH, but then deposit to vault `**amount**` of weETH. Note that `**weETHAmount**` and `**amount**` is different, the ratio got calculated in here, which is weETHAmount : amount = 0.969e18 : 1e18. Because `**amount > weETHAmount**`, this will make the transaction reverted because not enough approval token, breaking core functionality

```
    function _deposit(uint256 amount, address vault) internal {
        SafeERC20.safeTransferFrom(_eETH, _msgSender(), address(this), amount);
        SafeERC20.safeIncreaseAllowance(_eETH, address(_weETH), amount);
        uint256 weETHAmount = _weETH.wrap(amount);

        SafeERC20.safeIncreaseAllowance(_weETH, address(vault), weETHAmount); //<@@ approve weETHA
mount
        IVault(vault).deposit(amount, _msgSender()); //<@@ use amount
    }
```

0xzoobi: The `**_deposit**` function is intended for facilitating token deposits into a vault. However, there is a potential inconsistency in variable usage when calling the deposit function of the `**IVault**` contract. The correct approach is to invoke `**IVault(vault).deposit**` with the parameter `**weETHAmount**` instead of `**amount**`. Please refer to the specific location in the code for precise details. This implementation is likely to function correctly in most conditions, given the 1:1 mapping.
But, It is crucial to note a past scenario where `**stETH**` lost its peg with `**ETH**`; further details can be found in this blog: https://www.nansen.ai/research/on-chain-forensics-demystifying-steth-depeg
In the event of a similar scenario, it has the potential to generate unpredictable outcomes. It is advisable to prefer safety over risking adverse consequences.

Recommendation

0xhuy0512:

```
    function _deposit(uint256 amount, address vault) internal {
        SafeERC20.safeTransferFrom(_eETH, _msgSender(), address(this), amount);
        SafeERC20.safeIncreaseAllowance(_eETH, address(_weETH), amount);
        uint256 weETHAmount = _weETH.wrap(amount);

        SafeERC20.safeIncreaseAllowance(_weETH, address(vault), weETHAmount);
-       IVault(vault).deposit(amount, _msgSender());
+       IVault(vault).deposit(weETHAmount, _msgSender());
    }
```

0xzoobi: To fix the issue make the following changes in the `**EETHDepositHelper::_deposit()**`

```
SafeERC20.safeIncreaseAllowance(_weETH, address(vault), weETHAmount);
- IVault(vault).deposit(amount, _msgSender());
+ IVault(vault).deposit(weETHAmount, _msgSender());
```

Client Response

0xhuy0512: Fixed,We've added `**_wrap**` function and use its return value directly to `**_deposit**` - It would fix this issue
https://github.com/mitosis-org/evm/blob/9dedfb6b742b71af5460789ea8735695a454feb4/src/helpers/EETHDepositHelper.sol#L99
0xzoobi: Fixed,We've added _wrap function and use its return value directly to _deposit - It would fix this issue
https://github.com/mitosis-org/evm/blob/9dedfb6b742b71af5460789ea8735695a454feb4/src/helpers/EETHDepositHelper.sol#L99

MTS-6:Cap.sol contract can't send crosschain using Hyperlane because it doesn't hold any native token as crosschain fee

Category	Severity	Client Response	Contributor
Logical	Critical	Fixed	0xhuy0512

Code Reference

- code/src/vault/Cap.sol#L232

```
232: _dispatch(domains[i], fee, msgCapFilled);
```

Description

0xhuy0512: Contract **`Cap.sol`** doesn't have any method to receive native token except **`handle()`** function which is can only be called by Hyperlane's Mailbox (only send native token when there's exceed fee paid). And because of that, there's no way for **`Cap`** contract to send crosschain message using Hyperlane to announce that the epoch cap is filled, which broke the core functionality

Recommendation

0xhuy0512: Add a **`receive()`** function in Cap contract so that it can hold native token

Client Response

0xhuy0512: Fixed,We've fixed this by integrating ATM with Cap

```
    function _broadcastEpoch(IATM atm_, uint32[] memory domains, bytes memory msg_) internal {
        for (uint256 i = 0; i < domains.length; i++) {
            uint256 fee = _quoteDispatch(domains[i], msg_);

            if (address(atm_) != address(0x0)) atm_.borrow(fee);

            _dispatch(domains[i], fee, msg_);
        }
    }
```

MTS-7:CCDM Host's token approval to wrong receiver

Category	Severity	Client Response	Contributor
Logical	Critical	Fixed	0xhuy0512

Code Reference

- code/src/helpers/ccdm/CCDMHost.sol#L147

```
147: SafeERC20.safeIncreaseAllowance(IERC20(l1Asset), address(this), msgBridge.amount);
```

Description

0xhuy0512: When bridging token in CCDMHost using native bridge, CCDMHost should approve token to bridge adapter in order to transfer it to native bridge. But for some reason, CCDMHost approve token to itself:

```
            SafeERC20.safeIncreaseAllowance(IERC20(l1Asset), address(this), msgBridge.amount);
```

Because of this, all of the bridge transaction will be reverted because there's no token approved for bridge to transfer

Recommendation

0xhuy0512: Approve to the bridge adapter instead:

```
+           SafeERC20.safeIncreaseAllowance(IERC20(l1Asset), address(this), msgBridge.amount);
-           SafeERC20.safeIncreaseAllowance(IERC20(l1Asset), address($._bridges[origin]), msgBrid
ge.amount);
```

Client Response

0xhuy0512: Fixed,https://github.com/mitosis-org/evm/blob/83a419a39c24f7cf81a671952c8bd33b00c186ec/src/helpers/ccdm/CCDMHost.sol#L243

MTS-8:Bridging token using native bridge send to wrong receiver

Category	Severity	Client Response	Contributor
Logical	Critical	Fixed	0xhuy0512

Code Reference

- code/src/helpers/ccdm/CCDMHost.sol#L149

```
149: $._bridges[origin].bridgeAsset(sender.toAddress(), l1Asset, l2Asset, msgBridge.amount);
```

Description

0xhuy0512: As the docs in Notion said:

- *Step X. Process bridge request from L2*
  - *CCDM Client -> Hyperlane -> CCDM Host -> Bridge Adapter -> Vault*

CCDM Host will use bridge adapter to send token to L2 vault. But in the code, CCDM Host will send token to L2 CCDM Client:

```
    function _handle(uint32 origin, bytes32 sender, bytes calldata rawMsg) internal override {
        ...
        } else if (msgType == MsgType.Bridge) {
            MsgBridge memory msgBridge = Message.decodeBridge(rawMsg);

            StorageV1 storage $ = _getStorageV1();

            address l1Asset = msgBridge.token.toAddress();
            address l2Asset = $._assetL1L2Map[origin][l1Asset];

            SafeERC20.safeIncreaseAllowance(IERC20(l1Asset), address(this), msgBridge.amount);

@@@@>       $._bridges[origin].bridgeAsset(sender.toAddress(), l1Asset, l2Asset, msgBridge.amoun
t);
        }
```

The receiver of token in L2 here is **`sender.toAddress()`** which is the address of caller that sent Bridge message to CCDM, which is L2 CCDM Client in **`adjust()`** function. And because CCDMClient is not desgined to hold any token, there's no way to take token out, leads to token stuck forever when bridging

Recommendation

0xhuy0512:

```
+            $._bridges[origin].bridgeAsset(msgBridge.receiver.toAddress(), l1Asset, l2Asset, msgB
ridge.amount);
-            $._bridges[origin].bridgeAsset(sender.toAddress(), l1Asset, l2Asset, msgBridge.amoun
t);
```

Client Response

0xhuy0512: Fixed,https://github.com/mitosis-org/evm/blob/9dedfb6b742b71af5460789ea8735695a454feb4/src/helpers/ccdm/CCDMHost.sol#L235

MTS-9:User who hold vault token by trading can't redeem token to earn back themself, make the vault token useless

Category	Severity	Client Response	Contributor
Logical	Medium	Mitigated	0xhuy0512

Code Reference

- code/src/vault/BasicVault.sol#L258-L263

```
258: DepositInfo storage info = $._deposits[receiver];
259:
260:         if (info.logs.length == 0) {
261:             info.resolved -= amount;
262:             return;
263:         }
```

Description

0xhuy0512: Because vault token is tradable, the original owner can transfer/trade to someone else. But the problem is that the new owner can't execute **`BasicVault.redeem()`** because the new owner's deposit logs is empty, hence will get DOS in **`_pruneDepositLog()`**. Because of that, the vault token in new owner will become useless. Malicious user can use this issues to sell vault token to naive seller

```
    function _pruneDepositLog(uint256 amount, address receiver) internal {
        UtilStorageV1 storage $ = _getUtilStorageV1();

        if ($._redeemPeriod <= 0) {
            return;
        }

        DepositInfo storage info = $._deposits[receiver];

        if (info.logs.length == 0) {
            info.resolved -= amount;  <@@ DOS here
            return;
        }

        ...
    }
```

Recommendation

0xhuy0512: There're 2 ways to fix this:

1. Quick way: make the vault token nontransferable, use msg.sender's DepositLog to deduct when redeem instead of receiver

2. Long and better way: make a logic that will transfer log from transferrer user to transferred user that will be executed when transferring vault token

Client Response

0xhuy0512: Mitigated,We've decided to bypass the redeem period by setting the parameter value to zero. Additionally, we're planning to switch to implementing an unstaking period rather than a redeem period. The main difference between unstaking and redeeming is that the unstaking period will start when a user executes a redeem, while the redeem period will start when a user executes a deposit.

MTS-10:Unsupported Opcode in Multi-Chain Deployment

Category	Severity	Client Response	Contributor
Language Specific	Medium	Fixed	ret2basic

Code Reference

- code/src/helpers/ccdm/CCDMClient.sol#L2

```
2: pragma solidity 0.8.23;
```

- code/src/helpers/ccdm/CCDMHost.sol#L2

```
2: pragma solidity 0.8.23;
```

- code/src/vault/BasicVault.sol#L2

```
2: pragma solidity 0.8.23;
```

- code/src/vault/BasicVaultFactory.sol#L2

```
2: pragma solidity 0.8.23;
```

- code/src/vault/Cap.sol#L2

```
2: pragma solidity 0.8.23;
```

Description

ret2basic: The primary concern identified in the smart contracts relates to the Solidity compiler version used, specifically `**`pragma solidity 0.8.23;`**`. This version, along with every version after `**`0.8.19`**`, introduces the use of the `**`PUSH0`**` opcode. This opcode is not universally supported across all Ethereum Virtual Machine (EVM)-based Layer 2 (L2) solutions. For instance, ZKSync, one of the targeted platforms for this protocol's deployment, does not currently support the `**`PUSH0`**` opcode.
The consequence of this incompatibility is that contracts compiled with Solidity versions higher than `**`0.8.19`**` may not function correctly or fail to deploy on certain L2 solutions.
The impact of using a Solidity compiler version that includes the PUSH0 opcode is significant for a protocol intended to operate across multiple EVM-based chains. Chains that do not support this opcode will not be able to execute the contracts as intended, resulting in a range of issues from minor malfunctions to complete deployment failures. This limitation directly affects the protocol's goal of wide compatibility and interoperability, potentially excluding it from deployment on key L2 solutions like ZKsync.

Recommendation

ret2basic: To mitigate this issue and ensure broader compatibility with various EVM-based L2 solutions, it is recommended to downgrade the Solidity compiler version used in the smart contracts to 0.8.19. This version does

not utilize the PUSH0 opcode and therefore maintains compatibility with a wider range of L2 solutions, including ZKsync.

```
- pragma solidity 0.8.23;
+ pragma solidity 0.8.19;
```

This change will allow the protocol to maintain a consistent and deterministic bytecode across all targeted chains, ensuring functionality and deployment success on platforms that currently do not support the PUSH0 opcode.

Client Response

ret2basic: Fixed,We've fixed by changing hardfork version of compiler to Paris

MTS-11:Underpayaing the L2 gas may lead to loss of funds

Category	Severity	Client Response	Contributor
DOS	Medium	Fixed	0xzoobi

Code Reference

- code/src/helpers/adapter/OptimismBridgeAdapter.sol#L9
- code/src/helpers/adapter/OptimismBridgeAdapter.sol#L20

```
9: IOptimismGateway public immutable bridge;
```

```
20: bridge.depositERC20To(l1Asset, l2Asset, destAddr, amount, minGasLimit, "");
```

Description

0xzoobi: In the current implementation, the **`minGasLimit`** is declared as **`immutable`**. Consequently, if there is a need to update this value for any reason, it is not possible. This limitation could have significant consequences, especially in scenarios where there is a sudden spike in gas prices on the current Layer 2 (L2). In such a situation, if the current gas limit is considerably higher than the set value of **`minGasLimit`**, users may incur losses when using **`bridge.depositERC20To`**.
Similar instance reported in a **Spearbit Security Review of Li.Fi Bridge Issue 5.2.7** - https://github.com/spearbit/portfolio/blob/master/pdfs/LIFI-Spearbit-Security-Review.pdf

Recommendation

0xzoobi: To address the issue, introduce a setter function for **`minGasLimit`**. This will provide the flexibility to update the value when needed.
Here's an example:

```
    function setMinGasLimit(uint256 _newMinGasLimit) external onlyOwner {
        minGasLimit = _newMinGasLimit;
    }
```

Client Response

0xzoobi: Fixed by introducing setters for each parameters
https://github.com/mitosis-org/evm/blob/9dedfb6b742b71af5460789ea8735695a454feb4/src/helpers/adapter/ArbitrumBridgeAdapter.sol#L106

MTS-12:The epoch assertion in `_checkRemoteStateAndAdvance()` function is not behave as expected

Category	Severity	Client Response	Contributor
Logical	Medium	Fixed	0xhuy0512

Code Reference

- code/src/vault/Cap.sol#L261-L272

```
261: // 1. find the minimum epoch among all remote chains.
262:         // 2. check the every remote chain's epoch is same as the min epoch.
263:         for (uint256 i = 1; i < domains.length; i++) {
264:             uint256 epochForDomain = $._epoch[domains[i]];
265:
266:             if (epochForDomain < nextEpoch) {
267:                 nextEpoch = epochForDomain;
268:             }
269:             if (epochForDomain != nextEpoch) {
270:                 return;
271:             }
272:         }
```

Description

0xhuy0512: As the dev's comment said:

```
// 1. find the minimum epoch among all remote chains.
// 2. check the every remote chain's epoch is same as the min epoch.
```

But the logic execute differently:

```
        for (uint256 i = 1; i < domains.length; i++) {
            uint256 epochForDomain = $._epoch[domains[i]];

            if (epochForDomain < nextEpoch) {
                nextEpoch = epochForDomain;
            }
            if (epochForDomain != nextEpoch) {
                return;
            }
        }
```

The loop actually just check if the domains array is an descending array or not:

- **`domains = [3,2,2,1]`** -> not return
- **`domains = [3,2,1,0]`** -> not return
- **`domains = [3,2,2,3]`** -> return

Moreover, in `1. find the minimum epoch among all remote chains.` should be calculate in a separate loop to find the minimum epoch

Recommendation

0xhuy0512:

```
        for (uint256 i = 1; i < domains.length; i++) {
            uint256 epochForDomain = $._epoch[domains[i]];

            if (epochForDomain < nextEpoch) {
                nextEpoch = epochForDomain;
            }
-           if (epochForDomain != nextEpoch) {
-               return;
-           }
        }

+        for (uint256 i = 1; i < domains.length; i++) {
+            uint256 epochForDomain = $._epoch[domains[i]];

+           if (epochForDomain != nextEpoch) {
+               return;
+           }
+       }
```

Client Response

0xhuy0512: Fixed by recommended approach.
https://github.com/mitosis-org/evm/blob/9dedfb6b742b71af5460789ea8735695a454feb4/src/vault/Cap.sol#L324-L338

MTS-13:Not pulling token to Adapter contracts lead to to can't bridge token crosschain using native bridge

Category	Severity	Client Response	Contributor
Logical	Medium	Fixed	0xhuy0512

Code Reference

- code/src/helpers/adapter/ArbitrumBridgeAdapter.sol#L34

```
34: bridge.outboundTransfer(l1Asset, destAddr, amount, maxGas, gasPriceBid, "");
```

- code/src/helpers/adapter/OptimismBridgeAdapter.sol#L20

```
20: bridge.depositERC20To(l1Asset, l2Asset, destAddr, amount, minGasLimit, "");
```

- code/src/helpers/adapter/PolygonZkEvmBridgeAdapter.sol#L24

```
24: bridge.bridgeAsset(destNetwork, destAddr, amount, l1Asset, forceUpdateGlobalExitRoot, "");
```

- code/src/helpers/ccdm/CCDMHost.sol#L147-L150

```
147: SafeERC20.safeIncreaseAllowance(IERC20(l1Asset), address(this), msgBridge.amount);
148:
149:             $._bridges[origin].bridgeAsset(sender.toAddress(), l1Asset, l2Asset, msgBridge.amoun
t);
150:         }
```

Description

0xhuy0512: This issue happens in both 3 adapter contract: ArbitrumBridgeAdapter, OptimismBridgeAdapter, PolygonZkEvmBridgeAdapter
In CCDMHost contract, when bridging token using native bridge, it will first approve token to adapter then call to adapter's `**bridgeAsset()**`:

```
        } else if (msgType == MsgType.Bridge) {
            MsgBridge memory msgBridge = Message.decodeBridge(rawMsg);

            StorageV1 storage $ = _getStorageV1();

            address l1Asset = msgBridge.token.toAddress();
            address l2Asset = $._assetL1L2Map[origin][l1Asset];

            SafeERC20.safeIncreaseAllowance(IERC20(l1Asset), address(this), msgBridge.amount);

            $._bridges[origin].bridgeAsset(sender.toAddress(), l1Asset, l2Asset, msgBridge.amoun
t);
```

But both 3 adapters lack a logic that pulling back the token to themselves and approve that token to the native bridge. Because of that, bridging will always get reverted, breaking the core function

Recommendation

0xhuy0512: Add this line to both three adapters:

```
    function bridgeAsset(...) external {
+       IERC20(l1Asset).safeTransferFrom(msg.sender, address(this), amount);
+       IERC20(l1Asset).safeApprove(bridge, amount);
        ...
    }
```

Client Response

0xhuy0512: Fixed by adding approve statement https://github.com/mitosis-org/evm/blob/9dedfb6b742b71af5460789ea8735695a454feb4/src/helpers/adapter/ArbitrumBridgeAdapter.sol#L132L135

MTS-14:zkEVM bridge need an extra step in L2 to successfully bridged

Category	Severity	Client Response	Contributor
Logical	Low	Acknowledged	0xhuy0512

Code Reference

- code/src/helpers/adapter/ArbitrumBridgeAdapter.sol#L33-L35

```
33: function bridgeAsset(address destAddr, address l1Asset, address, uint256 amount) external {
34:         bridge.outboundTransfer(l1Asset, destAddr, amount, maxGas, gasPriceBid, "");
35:     }
```

- code/src/helpers/adapter/OptimismBridgeAdapter.sol#L19-L21

```
19: function bridgeAsset(address destAddr, address l1Asset, address l2Asset, uint256 amount) externa
l {
20:         bridge.depositERC20To(l1Asset, l2Asset, destAddr, amount, minGasLimit, "");
21:     }
```

- code/src/helpers/adapter/PolygonZkEvmBridgeAdapter.sol#L23-L25

```
23: function bridgeAsset(address destAddr, address l1Asset, address, uint256 amount) external {
24:         bridge.bridgeAsset(destNetwork, destAddr, amount, l1Asset, forceUpdateGlobalExitRoot,
"");
25:     }
```

Description

0xhuy0512: https://docs.polygon.technology/zkEVM/architecture/protocol/zkevm-bridge/flow-of-assets/#asset-flow-from-l1-l2 :

> *"5. In order to complete the bridging process, the user calls the Claim function of the Bridge SC and provides a Merkle proof to the fact that the correct exit leaf was included and represented in the Global Exit Root."*

zkEVM bridge need user to call to **`claimAsset()`** in L2 in order to successfully bridged. As the token receiver in L2 is BasicVault contract, we should make a fuction in PolygonZkEVM's BasicVault to claim the asset

Recommendation

0xhuy0512: Make a function in BasicVault that call to this PolygonZkEVMBridge's function: https://github.com/0xPolygonHermez/zkevm-contracts/blob/53e95f3a236d8bea87c27cb8714a5d21496a3b20/contracts/PolygonZkEVMBridge.sol#L311C1-L322C37

Client Response

0xhuy0512: Acknowledged, Executing claim has allowed for third-party. So we can just running our bot or depending on other bots like Zenland:deployer
the situation like the claim can be callable by third party and still needs to fetch merkle proof from offchain makes the CCDMClient cannot from their deposit. Hence the Low severity is more appropriate

MTS-15:Use `Ownable2StepUpgradeable` instead of `OwnableUpgradeable` contract

Category	Severity	Client Response	Contributor
Privilege Related	Low	Fixed	Meliclit, 0xzoobi

Code Reference

- code/src/vault/BasicVault.sol#L74
- code/src/vault/BasicVault.sol#L74

```
74: OwnableUpgradeable,
```

```
74: OwnableUpgradeable,
```

- code/src/vault/BasicVaultFactory.sol#L41
- code/src/vault/BasicVaultFactory.sol#L41

```
41: contract BasicVaultFactory is IVaultFactory, OwnableUpgradeable, BasicVaultFactoryStorageV1 {
```

```
41: contract BasicVaultFactory is IVaultFactory, OwnableUpgradeable, BasicVaultFactoryStorageV1 {
```

- code/src/vault/Cap.sol#L36
- code/src/vault/Cap.sol#L36

```
36: contract Cap is ICap, OwnableUpgradeable, Router, CapStorageV1 {
```

```
36: contract Cap is ICap, OwnableUpgradeable, Router, CapStorageV1 {
```

- code/src/vault/VaultHub.sol#L5
- code/src/vault/VaultHub.sol#L5

```
5: import {OwnableUpgradeable} from "@ozu/access/OwnableUpgradeable.sol";
```

```
5: import {OwnableUpgradeable} from "@ozu/access/OwnableUpgradeable.sol";
```

Description

Meliclit: Use the Ownable2Step variant of the Ownable contract to better safeguard against accidental transfers of access control.
0xzoobi: The contracts does not implement a 2-Step-Process for transferring ownership.
So ownership of the contract can easily be lost when making a mistake when transferring ownership.
Since the privileged roles have critical function roles assigned to them. Assigning the ownership to a wrong user can be disastrous.
So Consider using the Ownable2StepUpgradeable contract from OZ

(https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/v5.0.0/contracts/access/Ownable2StepUpgradeable.sol instead.
The way it works is there is a `transferOwnership` to transfer the ownership and `acceptOwnership` to accept the ownership. Refer the above Ownable2Step.sol for more details.

Recommendation

Meliclit: Use `Ownable2StepUpgradeable`
0xzoobi: Implement 2-Step-Process for transferring ownership via Ownable2Step.

Client Response

Meliclit: Fixed,https://github.com/mitosis-org/evm/pull/44
0xzoobi: Fixed,https://github.com/mitosis-org/evm/pull/44

MTS-16:The `bridgeGateway` addresses are hardcoded

Category	Severity	Client Response	Contributor
DOS	Low	Fixed	0xzoobi

Code Reference

- code/src/helpers/adapter/ArbitrumBridgeAdapter.sol#L11

```
11: IArbitrumGateway public immutable bridge;
```

- code/src/helpers/adapter/OptimismBridgeAdapter.sol#L9

```
9: IOptimismGateway public immutable bridge;
```

- code/src/helpers/adapter/PolygonZkEvmBridgeAdapter.sol#L9

```
9: IPolygonZkEVMBridge public immutable bridge;
```

Description

0xzoobi: In the existing setup, the bridge gateway addresses for Arbitrum, Optimism, and Polygon have been set as **`immutable`**. This means that once defined during the constructor call, they cannot be changed. Consider a scenario in the future where a newer version is released, either to address issues in the old version or to introduce new features, leading to the deprecation of the old version. In such a case, the current implementation lacks a mechanism to update these gateway addresses.
As a consequence to this, if this happens, the contracts need to be redeployed.
One simple example to better explain this issue is **1inch**

1. 1inch Router v4 Announcement in 2021 10 Nov - https://blog.1inch.io/the-1inch-router-v4-is-rolled-out/
2. 1inch Router v5 Announcement in 2022 15 Nov - https://blog.1inch.io/the-1inch-router-v5-is-released/

Recommendation

0xzoobi: Add a setter function for the gateway addresses in all the bridge gateway contracts.

Client Response

0xzoobi: Fixed via parameterization
https://github.com/mitosis-org/evm/blob/9dedfb6b742b71af5460789ea8735695a454feb4/src/helpers/adapter/ArbitrumBridgeAdapter.sol#L110

MTS-17:No whitelist token in CCDM Host lead to user can accidentally losing token

Category	Severity	Client Response	Contributor
Logical	Low	Fixed	0xhuy0512

Code Reference

- code/src/helpers/ccdm/CCDMClient.sol#L125-L148

```
125: function _handle(uint32, bytes32, bytes calldata rawMsg) internal {
126:         MsgDeposit memory msg_ = Message.decodeDeposit(rawMsg);
127:
128:         StorageV1 storage $ = _getStorageV1();
129:
130:         VaultInfo storage info = $._vaults[$._vaultIdxByL1Asset[msg_.token]];
131:
132:         ISudoVault vault = info.vault;
133:
134:         if (address(vault) == address(0x0)) {
135:             revert Error.InvalidDepositRequest("non-registered asset");
136:         }
137:
138:         try vault.manualDeposit(msg_.amount, msg_.receiver.toAddress()) returns (uint256 spent)
{
139:             if (spent < msg_.amount) {
140:                 _processRefund(msg_.receiver, msg_.token, msg_.amount - spent);
141:             }
142:             _processSpent(msg_.receiver, msg_.token, spent);
143:             emit DepositSuccess(msg_.receiver.toAddress(), info.l1Asset, info.l2Asset, spent);
144:         } catch {
145:             _processRefund(msg_.receiver, msg_.token, msg_.amount);
146:             emit DepositFailure(msg_.receiver.toAddress(), info.l1Asset, info.l2Asset, msg_.amo
unt);
147:         }
148:     }
```

- code/src/helpers/ccdm/CCDMHost.sol#L75-L106

```
75: function deposit(uint32 domain, address token, address receiver, uint256 amount) external payabl
e {
76:         StorageV1 storage $ = _getStorageV1();
77:
78:         bytes memory enc = Message.encodeDeposit(MsgDeposit(receiver.toBytes32(), token.toBytes3
2(), amount));
79:
80:         uint256 hplFee = _quoteDispatch(domain, enc);
81:         uint256 userFee = _calcFee($._fee.gas, $._fee.adjustment);
82:         if (hplFee + userFee < msg.value) {
83:             revert Error.InsufficientFee();
84:         }
85:
86:         uint256 refund = msg.value - hplFee - userFee;
87:
88:         if (userFee > 0) {
89:             (bool ok, bytes memory ret) = payable($._fee.receiver).call{value: userFee}("");
90:             if (!ok) {
91:                 revert Error.EthTransferFailed(userFee, ret);
92:             }
93:         }
94:
95:         _dispatch(domain, hplFee, enc);
96:
97:         if (refund > 0) {
98:             (bool ok, bytes memory ret) = payable(_msgSender()).call{value: refund}("");
99:             if (!ok) {
100:                 revert Error.EthTransferFailed(refund, ret);
101:              }
102:          }
103:
104:          SafeERC20.safeTransferFrom(IERC20(token), _msgSender(), address(this), amount);
105:     }
```

Description

0xhuy0512: CCDMHost's **`deposit()`** doesn't check if the deposited token is supported or not:

```
    function deposit(uint32 domain, address token, address receiver, uint256 amount) external paya
ble {
        StorageV1 storage $ = _getStorageV1();

        <<<<@@@ NOT CHECK `TOKEN`

        bytes memory enc = Message.encodeDeposit(MsgDeposit(receiver.toBytes32(), token.toBytes32
(), amount));

        uint256 hplFee = _quoteDispatch(domain, enc);
        uint256 userFee = _calcFee($._fee.gas, $._fee.adjustment);
        if (hplFee + userFee < msg.value) {
            revert Error.InsufficientFee();
        }

        uint256 refund = msg.value - hplFee - userFee;

        if (userFee > 0) {
            (bool ok, bytes memory ret) = payable($._fee.receiver).call{value: userFee}("");
            if (!ok) {
                revert Error.EthTransferFailed(userFee, ret);
            }
        }

        _dispatch(domain, hplFee, enc);

        if (refund > 0) {
            (bool ok, bytes memory ret) = payable(_msgSender()).call{value: refund}("");
            if (!ok) {
                revert Error.EthTransferFailed(refund, ret);
            }
        }

        SafeERC20.safeTransferFrom(IERC20(token), _msgSender(), address(this), amount);
    }
```

It will sent the crosschain message to L2 CCDMClient, which will reverted without any refund:

```
    function _handle(uint32, bytes32, bytes calldata rawMsg) internal {
        MsgDeposit memory msg_ = Message.decodeDeposit(rawMsg);

        StorageV1 storage $ = _getStorageV1();

        VaultInfo storage info = $._vaults[$._vaultIdxByL1Asset[msg_.token]];

        ISudoVault vault = info.vault;

        if (address(vault) == address(0x0)) {
            revert Error.InvalidDepositRequest("non-registered asset");   <<<<<<<<<<REVERTED HERE
        }

        ...
    }
```

This issue will lead user can accidentally sent a wrong token and not get it back whatsoever

Recommendation

0xhuy0512: Add a check whether support this token or not using `**_assetL1L2Map**`:

```
    function deposit(uint32 domain, address token, address receiver, uint256 amount) external paya
ble {
        StorageV1 storage $ = _getStorageV1();

+       if ($._assetL1L2Map[domain][token] == address(0)) revert;

        bytes memory enc = Message.encodeDeposit(MsgDeposit(receiver.toBytes32(), token.toBytes32
(), amount));

        uint256 hplFee = _quoteDispatch(domain, enc);
        uint256 userFee = _calcFee($._fee.gas, $._fee.adjustment);
        if (hplFee + userFee < msg.value) {
            revert Error.InsufficientFee();
        }

        uint256 refund = msg.value - hplFee - userFee;

        if (userFee > 0) {
            (bool ok, bytes memory ret) = payable($._fee.receiver).call{value: userFee}("");
            if (!ok) {
                revert Error.EthTransferFailed(userFee, ret);
            }
        }

        _dispatch(domain, hplFee, enc);

        if (refund > 0) {
            (bool ok, bytes memory ret) = payable(_msgSender()).call{value: refund}("");
            if (!ok) {
                revert Error.EthTransferFailed(refund, ret);
            }
        }

        SafeERC20.safeTransferFrom(IERC20(token), _msgSender(), address(this), amount);
    }
```

Client Response

0xhuy0512: Fixed, https://github.com/mitosis-org/evm/blob/83a419a39c24f7cf81a671952c8bd33b00c186ec/src/helpers/ccdm/CCDMHost.sol#L173-L175

MTS-18:Missing 0 amount check

Category	Severity	Client Response	Contributor
Logical	Low	Fixed	ravikiran_web3

Code Reference

- code/src/helpers/ccdm/CCDMHost.sol#L75-L105

```
75: function deposit(uint32 domain, address token, address receiver, uint256 amount) external payabl
e {
76:         StorageV1 storage $ = _getStorageV1();
77:
78:         bytes memory enc = Message.encodeDeposit(MsgDeposit(receiver.toBytes32(), token.toBytes3
2(), amount));
79:
80:         uint256 hplFee = _quoteDispatch(domain, enc);
81:         uint256 userFee = _calcFee($._fee.gas, $._fee.adjustment);
82:         if (hplFee + userFee < msg.value) {
83:             revert Error.InsufficientFee();
84:         }
85:
86:         uint256 refund = msg.value - hplFee - userFee;
87:
88:         if (userFee > 0) {
89:             (bool ok, bytes memory ret) = payable($._fee.receiver).call{value: userFee}("");
90:             if (!ok) {
91:                 revert Error.EthTransferFailed(userFee, ret);
92:             }
93:         }
94:
95:         _dispatch(domain, hplFee, enc);
96:
97:         if (refund > 0) {
98:             (bool ok, bytes memory ret) = payable(_msgSender()).call{value: refund}("");
99:             if (!ok) {
100:                 revert Error.EthTransferFailed(refund, ret);
101:              }
102:          }
103:
104:          SafeERC20.safeTransferFrom(IERC20(token), _msgSender(), address(this), amount);
105:     }
```

Description

ravikiran_web3: CCDMHost::deposit() function accepts the below four parameters. But there are not enough validation to handle incorrect data.

1. domain,
2. token,
3. receiver,
4. amount

Of the above parameter, domain is indirectly validated while reading the gas price as below. But, other parameters are not validated.

1. **token**: The token can be validated before initiating a dispatch using the storage map. If there is an L2 token mapped for token, then it can be assumed valid.
2. **receiver:** The receiver can be 0 address. The problem with low level call function is that, for 0 address, it will return bool as successful and hence cannot be caught by the check implemented. It would be better to validated receiver address to be no zero.
3. **amount**: Caller can potentially pass 0 and complete this transaction with out issues and hence commits 0 tokens to the CCDM Host contract.

```
SafeERC20.safeTransferFrom(IERC20(token), _msgSender(), address(this), amount);
```

Recommendation

ravikiran_web3: To address the potential for abuse or incorrect processing,
Token: Validate the token to have a corresponding L2 token before initiating a dispatch call. The l2Asset from the below should be a valid address.

```
address l2Asset = $._assetL1L2Map[origin][token];
```

receiver: validate for non zero address.
amount: Amount of the commitment for the caller to deposit tokens that are being relied to the destination chain. This should be greater than 0.

Client Response

ravikiran_web3: Fixed,We've fixed by adding nonZero modifier for every functions that have amount

MTS-19:Ensure no native asset value is sent in `payable` method that can handle ERC20 transfers as well

Category	Severity	Client Response	Contributor
Logical	Low	Fixed	0xzoobi

Code Reference

- code/src/helpers/adapter/PolygonZkEvmBridgeAdapter.sol#L24

```
24: bridge.bridgeAsset(destNetwork, destAddr, amount, l1Asset, forceUpdateGlobalExitRoot, "");
```

Description

0xzoobi: The **`bridgeAsset`** method of **`PolygonZkEVMBridge`** is designated as payable, allowing it to interact with both the native asset and ERC20 tokens. However, in the code path where it verifies that the token is an ERC20 token rather than the native asset, there is currently no validation ensuring that the user did not mistakenly include a transaction value. Although the probability of this occurrence is relatively low, as it necessitates a user error, if it does happen, the native asset value could become trapped within the PolygonZkEVMBridge contract.

Recommendation

0xzoobi: To address the issue and prevent the inadvertent transfer of native asset value when bridging ERC20 tokens, you can implement the following fix by incorporating the provided code:

```
    // Ensure no native asset value is sent
    require(msg.value == 0, "PolygonZkEVMBridge::bridgeAsset: Expected zero native asset value whe
n bridging ERC20 tokens");
```

Client Response

0xzoobi: Fixed,bridgeAsset is no more payable.

MTS-20:Typo error

Category	Severity	Client Response	Contributor
Code Style	Informational	Fixed	ravikiran_web3

Code Reference

- code/src/helpers/ccdm/CCDMClient.sol#L27

```
27: contract CCDMClientStrorageV1 {
```

Description

ravikiran_web3: 1. Minor typo while naming **CCDMClientStrorageV1** contract
2. Minor typo while naming **CCDMHostStrorageV1** contract

Recommendation

ravikiran_web3: 1. Correct as **CCDMClientStorageV1** contract
2. Correct as **CCDMHostStorageV1** contract

Client Response

ravikiran_web3: Fixed,Renamed

MTS-21:Open TODOs

Category	Severity	Client Response	Contributor
Code Style	Informational	Fixed	Meliclit, ravikiran_web3

Code Reference

- code/src/entrypoint/Entrypoint.sol#L11-L13
- code/src/entrypoint/Entrypoint.sol#L12

```
11: function _handle(uint32 origin, bytes32 sender, bytes calldata message) internal override {
12:         /// TODO: implement me
13:     }
```

```
12: /// TODO: implement me
```

Description

Meliclit: Code architecture, incentives, and error handling/reporting questions/issues should be resolved before deployment
ravikiran_web3: The _handle() function is not implemented in Entrypoint contract. This contract is doing nothing useful, but was listed for review.

```
function _handle(uint32 origin, bytes32 sender, bytes calldata message) internal override {
        /// TODO: implement me
    }
```

Recommendation

Meliclit: Remove todo
ravikiran_web3: Review with the team if this contract is needed. If yes, what is its intention as this contract is not design for upgradability.
So, what is the need to include this contract in the project at this point.
It is not clear from the repo as to what purpose this contract is serving.

Client Response

Meliclit: Fixed
ravikiran_web3: Fixed,removed

MTS-22:Gas Optimization

Category	Severity	Client Response	Contributor
Gas Optimization	Informational	Fixed	0xhuy0512

Code Reference

- code/src/vault/BasicVault.sol#L282

```
282: return;
```

- code/src/vault/Cap.sol#L50-L52

```
50: $._epoch[localDomain] = 0;
51:         $._load = 0;
52:         $._ready = false;
```

Description

0xhuy0512: No need to return at the end of the function **`_pruneDepositLog()`** in BasicVault contract:

```
    function _pruneDepositLog(uint256 amount, address receiver) internal {
        UtilStorageV1 storage $ = _getUtilStorageV1();

        if ($._redeemPeriod <= 0) {
            return;
        }

        DepositInfo storage info = $._deposits[receiver];

        if (info.logs.length == 0) {
            info.resolved -= amount;
            return;
        }

        uint256 high = info.logs.length;
        uint256 low = info.lastLogIdx;

        while (low < high) {
            uint256 mid = Math.average(low, high);
            if (info.logs[mid].at + $._redeemPeriod > block.timestamp) {
                high = mid;
            } else {
                low = mid + 1;
            }
        }

        uint256 extracted = info.logs[high - 1].cumulative - info.logs[info.lastLogIdx].cumulativ
e;

        info.lastLogIdx = high - 1;
        info.resolved = (info.resolved + extracted) - amount;

        return;       //<@@@@ no need
    }
```

0xhuy0512: The default value of uint256 is 0, default value of boolean is false. So there's no need to set it to default value when init

```
    function initialize(address owner, address hook, address ism) public initializer {
        _MailboxClient_initialize(hook, ism, owner);

        StorageV1 storage $ = _getStorageV1();
        $._epoch[localDomain] = 0;    //<@@@ no need
        $._load = 0;    //<@@@ no need
        $._ready = false;    //<@@@ no need
    }
```

Recommendation

0xhuy0512:

```
-    return;
```

0xhuy0512: Delete those line to save some gas and better clarity

```
    function initialize(address owner, address hook, address ism) public initializer {
        _MailboxClient_initialize(hook, ism, owner);

        StorageV1 storage $ = _getStorageV1();
-       $._epoch[localDomain] = 0;
-       $._load = 0;
-       $._ready = false;
    }
```

Client Response

0xhuy0512: Fixed via removing return statement as recommended
0xhuy0512: Fixed via removing redundant assignments as recommended, excluding assignment of DEFAULT_EPOCH (which is 1) to $.epoch[localDomain]

MTS-23:ArbitrumBridgeAdapter::bridgeAsset() ignores the seqNumber returned by underlying bridge on outbound transfer

Category	Severity	Client Response	Contributor
Code Style	Informational	Fixed	ravikiran_web3

Code Reference

- code/src/helpers/adapter/ArbitrumBridgeAdapter.sol#L33-L35

```
33: function bridgeAsset(address destAddr, address l1Asset, address, uint256 amount) external {
34:         bridge.outboundTransfer(l1Asset, destAddr, amount, maxGas, gasPriceBid, "");
35:     }
```

Description

ravikiran_web3: The call on bridge to transfer assets to Arbitrum chain ignores the returned data. The reason could be because only Arbitrum bridge api returns a value. For other chains, the apis does not return any value and hence to keep it consistent across chains, the return type might be ignored.

```
bridge.outboundTransfer(l1Asset, destAddr, amount, maxGas, gasPriceBid, "");
```

https://docs.arbitrum.io/devs-how-tos/bridge-tokens/how-to-bridge-tokens-custom-gateway

Recommendation

ravikiran_web3: At this point, there is no risk identified, but if the id was significant for tracking any errors in the cross chain communication, it would be helpful to emit events.
These events will come handy incase of any issues and needs the return value as reference for tracking.
The recommendation is to capture the return value and emit an event.

Client Response

ravikiran_web3: Fixed,https://github.com/mitosis-org/evm/blob/83a419a39c24f7cf81a671952c8bd33b00c186ec/src/helpers/adapter/ArbitrumBridgeAdapter.sol#L142-L147

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Invoices, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Invoice. This report provided in connection with the services set forth in the Invoices shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Invoice. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without Secure3's prior written consent in each instance.

This report is not an "endorsement" or "disapproval" of any particular project or team. This report is not an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Secure3 to perform a security assessment. This report does not provide any warranty or guarantee of free of bug of codes analyzed, nor do they provide any indication of the technologies, business model or legal compliancy.

This report should not be used in any way to make decisions around investment or involvement with any particular project. Instead, it represents an extensive assessing process intending to help our customers increase the quality of their code and high-level consistency of implementation and business model, while reducing the risk presented by cryptographic tokens and blockchain technology.

Secure3's position on the final decisions over blockchain technologies and corresponding associated transactions is that each company and individual are responsible for their own due diligence and continuous security.

The assessment services provided by Secure3 is subject to dependencies and under continuing development. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.