Electisec Olympus OHM CCIP PR#69 Review

Review Resources:

- Olympus OHM CCIP PR#69

Auditors:

- Panda
- Watermelon

Table of Contents

Review Summary

Olympus OHM CCIP PR review

This review covers pull request #69 of the Olympus OHM codebase, which introduces a multi-chain bridge to EVM- and SVM-compatible chains.

The PR of the Olympus OHM CCIP Repo was reviewed over three days. Two auditors conducted a code review between May 28 and May 30, 2025. The review was limited to the latest pull request commit d3ba48d9b83b53f170ffd9a53d97cdd59c538383.

Scope

The scope of the review consisted of the following contracts at the specific commit:

```
src/
├── periphery/
│   ├── bridge/
│   │   └── CCIPCrossChainBridge.sol
│   ├── interfaces/
│   │   └── ICCIPCrossChainBridge.sol
│   └── PeripheryEnabler.sol
└── policies/
    ├── bridge/
    │   ├── BurnMintTokenPoolBase.sol
    │   └── CCIPBurnMintTokenPool.sol
    └── interfaces/
        └── ICCIPTokenPool.sol
```

This review is a code review to identify potential vulnerabilities in the code. The reviewers did not investigate security practices or operational security and assumed that privileged accounts could be trusted. The reviewers did not evaluate the security of the code relative to a standard or specification. The review may not have identified all potential attack vectors or areas of vulnerability.

Electisec and the auditors make no warranties regarding the security of the code and do not warrant that the code is free from defects. Electisec and the auditors do not represent nor imply to third parties that the code has been audited nor that the code is free from defects. By deploying or using the code, Olympus OHM CCIP and users of the contracts agree to use the code at their own risk.

Findings Explanation

Findings are broken down into sections by their respective impact:

- Critical, High, Medium, Low impact
    - These are findings that range from attacks that may cause loss of funds, impact control/ownership of the contracts, or cause any unintended consequences/actions that are outside the scope of the requirements.
- Undetermined
    - Findings whose impact could not be fully assessed within the time constraints of the engagement. These issues may range from low to critical severity, and while their exact consequences remain uncertain, they present enough potential risk to warrant attention and remediation.
- Gas savings
    - Findings that can improve the gas efficiency of the contracts.
- Informational
    - Findings including recommendations and best practices.

Critical Findings

None

High Findings

None

Medium Findings

None

Low Findings

1. Low - SVM trusted remotes can't be revoked

`CCIPCrossChainBridge.setTrustedRemote` is used to set a Solana address as a trusted recipient of a CCIP transfer.

Technical Details

Because the highlighted method always writes `isSet: true` for an SVM remote, the contract cannot remove trusted Solana addresses:

```
/// @inheritdoc ICCIPCrossChainBridge
function setTrustedRemoteSVM(uint64 dstChainSelector_, bytes32 to_) external
onlyOwner {
    _trustedRemoteSVM[dstChainSelector_] = TrustedRemoteSVM({remoteAddress: to_,
isSet: true}); // AUDIT isSet is always true
    emit TrustedRemoteSVMSet(dstChainSelector_, to_);
}
```

Impact

Low.

Recommendation

Modify the method to allow writing any boolean value to the remote's `isSet` field:

```
@@ -443,9 +443,9 @@ contract CCIPCrossChainBridge is CCIPReceiver, PeripheryEnabler,
Owned, ICCIPCro

     /// @inheritdoc ICCIPCrossChainBridge
-    function setTrustedRemoteSVM(uint64 dstChainSelector_, bytes32 to_) external
onlyOwner {
-        _trustedRemoteSVM[dstChainSelector_] = TrustedRemoteSVM({remoteAddress:
to_, isSet: true});
-        emit TrustedRemoteSVMSet(dstChainSelector_, to_);
+    function setTrustedRemoteSVM(uint64 dstChainSelector_, bytes32 to_, bool
isSet_) external onlyOwner {
+        _trustedRemoteSVM[dstChainSelector_] = TrustedRemoteSVM({remoteAddress:
to_, isSet: isSet_});
+        emit TrustedRemoteSVMSet(dstChainSelector_, to_, isSet_);
     }

@@ -49,7 +49,7 @@ interface ICCIPCrossChainBridge {

     event TrustedRemoteEVMSet(uint64 indexed dstChainSelector, address indexed to);

-    event TrustedRemoteSVMSet(uint64 indexed dstChainSelector, bytes32 indexed to);
+    event TrustedRemoteSVMSet(uint64 indexed dstChainSelector, bytes32 indexed to,
bool isSet);

     event MessageFailed(bytes32 messageId);
```

Developer Response

Fixed in PR#72.

Gas Saving Findings

None

Informational Findings

1. Informational - CCIPCrossChainBridge allows an empty cross-chain sender to pass verification

`CCIPCrossChainBridge._receiveMessage` is used to process incoming CCIP messages from any CCIP-enabled network.

Technical Details

The lack of clarity in CCIP's documentation regarding how the `Client.Any2EVMMessage.sender` field is set for non-EVM source networks does not allow us to conclude that such a field may not be set to `bytes32(0)`.

In such case, the decoding and verification steps at CCIPCrossChainBridge.sol#L347-L349 would succeed for message containing an unset `_trustedRemoteEVM[message_.sourceChainSelector]` value.

Impact

Informational.

Recommendation

Verify that `_trustedEVMRemote[message_.sourceChainSelector] != address(0)` to ensure that only messages originated from trusted remote chains are processed.

Developer Response

Fixed in PR#72.

2. Informational - `CCIPCrossChainBridge.supportsInterface` returns incorrect value for `type(IEnabler).interfaceId`

CCIPCrossChainBridge inherits from PeripheryEnabler, which implements the IEnabler interface.

Technical Details

`CCIPCrossChainBridge.supportsInterface` returns `false` when provided with IEnabler's interface id, which is incorrect.

Impact

Informational

Recommendation

Modify the `PeripheryEnabler` contract, adding a `supportInterface(bytes4)` method, e.g.:

```
@@ -88,4 +88,8 @@ abstract contract PeripheryEnabler is IEnabler {
        // Emit the disabled event
        emit Disabled();
    }
+
+    function supportsInterface(bytes4 interfaceId_) public view virtual
returns(bool) {
+            return interfaceId_ == type(IEnabler).interfaceId;
+    }
}

@@ -460,10 +460,11 @@ contract CCIPCrossChainBridge is CCIPReceiver,
PeripheryEnabler, Owned, ICCIPCro
        return i_ccipRouter;
    }

-    function supportsInterface(bytes4 interfaceId_) public view virtual override
returns (bool) {
+    function supportsInterface(bytes4 interfaceId_) public view virtual
override(CCIPReceiver, PeripheryEnabler) returns (bool) {
        return
            interfaceId_ == type(ICCIPCrossChainBridge).interfaceId ||
-            super.supportsInterface(interfaceId_);
+            CCIPReceiver.supportsInterface(interfaceId_) ||
+            PeripheryEnabler.supportsInterface(interfaceId_);
    }

    // ========= ENABLER FUNCTIONS ========= //
```

Developer Response

Fixed in PR#74.

Final remarks

The review found that the in-scope assets were correctly implemented and adequately integrated with CCIP for both EVM- and SVM-compatible blockchains.

Given the significant overhead required to conduct end-to-end tests for CCIP-integrated systems in local environments effectively, it is strongly recommended to undergo extensive testing in testnet environments before deployment to ensure correct functionality and integration with CCIP off-chain components.