CrossChain

Security Review

January 2, 2024

Contents

1. Introduction

CrossChain is a cross-chain bridge aggregation protocol that supports any-2-any swaps by aggregating bridges and connecting them to DEX aggregators.

Disclaimer: This security review does not guarantee against a hack. It is a snapshot in time of contracts according to the specific commit. Any modifications to the code will require a new security review.

2. Risk classification

Severity level	Impact: High	Impact: Medium	Impact: Low
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

2.1 Impact

High - leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium - global losses < 10% or losses to only a subset of users, but still unacceptable.
Low - losses will be annoying but bearable--applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.

2.2 Likelihood

High - almost certain to happen, easy to perform, or not easy but highly incentivized
Medium - only conditionally possible or incentivized, but still relatively likely
Low - requires stars to align, or little-to-no incentive

2.3 Action required for severity levels

Critical - Must fix as soon as possible (if already deployed)

High - Must fix (before deployment if not already deployed)
Medium - Should fix
Low - Could fix

3. Executive Summary

Summary

Project Name	CrossChain
Type of Project	Bridge Aggregator, Bridge
Audit Timeline	December 20, 2023 - January 2, 2024
Methods	Manual Review

Issues Found

Severity	**Count**
Critical Risk	0
High Risk	17
Medium Risk	29
Low Risk	28
Total	**74**

4. Findings

4.1 High Risk

4.1.1 sends eth to arbitrary user

Severity: High Risk
Context: AcrossFacet.sol#120-146, AmarokFacet.sol#124-160, ArbitrumBridgeFacet.sol#172-190, ArbitrumBridgeFacet.sol#192-209, CBridgeFacet.sol#159-193, HyphenFacet.sol#91-116, MultichainFacet.sol#207-249, NXTPFacet.sol#126-172, StargateFacet.sol#192-216, TcrossFacet.sol#66-87, Tcross.sol#41-48
Description: Unprotected call to a function sending Ether to an arbitrary address.

```
function _startBridge(
    IApp.BridgeData memory _bridgeData,
    AcrossData memory _acrossData
) internal {

    ...

    spokePool.deposit{ value: isNative ? _bridgeData.preBridgeAmount: 0 }(
        _bridgeData.receiver,
        sendingAsset,
        _bridgeData.preBridgeAmount,
        _bridgeData.destinationChainId,
        _acrossData.relayerFeePct,
        _acrossData.quoteTimestamp,
        _acrossData.message,
        _acrossData.maxCount
    );

    emit TransferStarted(_bridgeData);
}
```

Recommendation: Ensure that an arbitrary user cannot withdraw unauthorized funds.
CrossChain: the contracts act as a proxy, and it calls to the bridge contract directly, it will be safe
Slither: Acknowledged.

4.1.2 has delegatecall inside a loop in a payable function

Severity: High Risk
Context: Multicall.sol#58
Description: Detect the use of delegatecall inside a loop in a payable function.

```
/// @inheritdoc IMulticall
function multicall(bytes[] calldata data) public payable override refundExcessNative
    results = new bytes[](data.length);
    for (uint256 i = 0; i < data.length; i++) {
        (bool success, bytes memory result) = address(this).delegatecall(data[i]);

        if (!success) {
            revertFromReturnedData(result);
        }

        results[i] = result;
    }
}
```

Recommendation: Carefully check that the function called by delegatecall is not payable/doesn't use msg.value.
CrossChain: the function called by delegatecall doesn't use msg.value
Slither: Acknowledged.

4.1.3 use msg.value in a loop

Severity: High Risk
Context: Executor.sol#241-263
Description: Detect the use of msg.value inside a loop.

```
function _fetchBalances(LibSwap.SwapData[] calldata _swapData)
    private
    view
    returns (uint256[] memory)
{
    uint256 numSwaps = _swapData.length;
    uint256[] memory balances = new uint256[](numSwaps);
    address asset;
    for (uint256 i = 0; i < numSwaps; ) {
        asset = _swapData[i].receivingAssetId;
        balances[i] = LibAsset.getOwnBalance(asset);

        if (LibAsset.isNativeAsset(asset)) {
            balances[i] -= msg.value;
        }

        unchecked {
            ++i;
        }
    }

    return balances;
}
```

Recommendation: Provide an explicit array of amounts alongside the receivers array, and check that the sum of all amounts matches msg.value.
CrossChain: there will be only one swap with ETH in a transaction
Slither: Acknowledged.

4.1.4 IERC20 is re-used

Severity: High Risk
Context: N/A
Description: If a codebase has two contracts the similar names, the compilation artifacts will not contain one of the contracts with the duplicate name.

```
import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
import { IERC20 } from "@axelar-network/axelar-cgp-solidity/contracts/interfaces/IERC20.sol";
```

Recommendation: Rename the contract.
CrossChain: it's an interface contract imported by third-party libraries

Slither: Acknowledged.

4.2 Medium Risk

4.2.1 divide before multiply

Severity: Medium Risk
Context: LibBytes.sol#229, LibBytes.sol#196, LibBytes.sol#122-147, LibBytes.sol#511-584
Description: Solidity's integer division truncates. Thus, performing division before multiplication can lead to precision loss.

```
function concatStorage(bytes storage _preBytes, bytes memory _postBytes)
    internal
{
    assembly {
    ...
    mask := exp(0x100, sub(mc, end))

    sstore(sc, mul(div(mload(mc), mask), mask))
}
```

Recommendation: Consider ordering multiplication before division.
CrossChain: it's ok for the operation here.
Slither: Acknowledged.

4.2.2 reentrancy vulnerabilities

Severity: Medium Risk
Context: FeeCollector.sol#171-197, FeeCollector.sol#70-101
Description: Detection of the reentrancy bug. Do not report reentrancies that involve Ether.

```
function batchWithdrawIntegratorFees(address[] memory tokenAddresses,
    address[] memory toAddresses, uint256[] memory amounts)
    external
{
    uint256 length = tokenAddresses.length;
    uint256 balance;
    for (uint256 i = 0; i < length; ) {
        balance = _balances[msg.sender][tokenAddresses[i]];
        require(amounts[i] <= balance, "Amount Exceeded!");
        if(LibAsset.NULL_ADDRESS == toAddresses[i]){
            toAddresses[i] = msg.sender;
        }
        if(0 == amounts[i]){
            amounts[i] = balance;
        }
        _balances[msg.sender][tokenAddresses[i]] -= amounts[i];
        LibAsset.transferAsset(
            tokenAddresses[i],
            payable(toAddresses[i]),
            amounts[i]
        );
        emit IntegratorFeesWithdrawn(tokenAddresses[i], toAddresses[i], amounts[i]);
        unchecked {
            ++i;
        }
    }
}
```

Recommendation: use the Checks-Effects-Interactions pattern to avoid re-entrancy.
CrossChain: it's in a loop to withdraw fee in batch, the function is nonReentrant.
Slither: Acknowledged.

4.2.3 unchecked low level calls

Severity: Medium Risk
Context: Receiver.sol#203-213, Receiver.sol#238, Receiver.sol#257
Description: The return value of a low-level call is not checked.

```
function _swapAndCompleteBridgeTokens(
    bytes32 _transactionId,
    LibSwap.SwapData[] memory _swapData,
    address assetId,
    address payable receiver,
    uint256 amount,
    bool reserveRecoverGas
) private {
    uint256 _recoverGas = reserveRecoverGas ? recoverGas : 0;

    if (LibAsset.isNativeAsset(assetId)) {
        // case 1: native asset
        if (reserveRecoverGas && gasleft() < _recoverGas) {
            // case 1a: not enough gas left to execute calls
            receiver.call{ value: amount }("");

            emit TransferRecovered(
                _transactionId,
                assetId,
                receiver,
                amount,
                block.timestamp
            );
            return;
        }
```

Recommendation: Ensure that the return value of a low-level call is checked or logged..
CrossChain: send ETH here, will be reverted if not enough ETH.
Slither: Acknowledged.

4.2.4 uninitialized local variables

Severity: Medium Risk
Context: GenericSwapFacet.sol#50, SwapperV2.sol#206
Description: Uninitialized local variables.

```
function swapTokensGeneric(
    bytes32 _transactionId,
    string calldata _integrator,
    string calldata _referrer,
    address payable _receiver,
    uint256 _minAmount,
    uint256 _integratorFee,
    address _integratorAddress,
    LibSwap.SwapData[] calldata _swapData
) external payable nonReentrant {
    if (LibUtil.isZeroAddress(_receiver)) {
        revert InvalidReceiver();
    }

    TmpVariables memory tmpVars;

    tmpVars.postSwapBalance = _depositAndSwap(
        _transactionId,
        _minAmount,
        _swapData,
        _receiver,
        true,
        _integratorFee,
        _integratorAddress
    );
    tmpVars.receivingAssetId = _swapData[_swapData.length - 1]
        .receivingAssetId;
```

Recommendation: Initialize all the variables. If a variable is meant to be initialized to zero, explicitly set it to zero to improve code readability.
CrossChain: it's a temp struct-type variable, the member vars are initialized.
Slither: Acknowledged.

4.2.5 ignores return value

Severity: Medium Risk
Context: AmarokFacet.sol#124-160, AmarokFacet.sol#124-160, ArbitrumBridgeFacet.sol#172-190, ArbitrumBridgeFacet.sol#192-209, MultichainFacet.sol#207-249, NXTPFacet.sol#126-172, NXTPFacet.sol#126-172, StargateFacet.sol#169-185, TcrossFacet.sol#66-87, AxelarExecutor.sol#55-77, AxelarExecutor.sol#84-139, Executor.sol#141-213, FeeCollector.sol#70-101, Receiver.sol#167-197
Description: The return value of an external call is not stored in a local or state variable.

```
function _startBridge(
    BridgeData memory _bridgeData,
    AmarokData calldata _amarokData
) private {
    // give max approval for token to Amarok bridge, if not already
    LibAsset.maxApproveERC20(
        IERC20(_bridgeData.sendingAssetId),
        address(connextHandler),
        _bridgeData.preBridgeAmount
    );

    // initiate bridge transaction
    if (_amarokData.payFeeWithSendingAsset) {
        connextHandler.xcall(
            _amarokData.destChainDomainId,
            _amarokData.callTo,
            _bridgeData.sendingAssetId,
            _amarokData.delegate,
            _bridgeData.preBridgeAmount - _amarokData.relayerFee,
            _amarokData.slippageTol,
            _amarokData.callData,
            _amarokData.relayerFee
        );
```

Recommendation: Ensure that all the return values of the function calls are used.
CrossChain: some external call from bridge contract doesn't return any value, it will be reverted when failed
Slither: Acknowledged.

4.3 Low Risk

4.3.1 missing zero address validation

Severity: Low Risk
Context: Tcross.sol#24, AcrossFacet.sol#43, TransferrableOwnership.sol#24, Executor.sol#70, Receiver.sol#49, Receiver.sol#51, Receiver.sol#69, Receiver.sol#76, Receiver.sol#205
Description: Detect missing zero address validation.

```
function setAmarokRouter(address _amarokRouter) external onlyOwner {
    amarokRouter = _amarokRouter;
    emit AmarokRouterSet(_amarokRouter);
}
```

Recommendation: Check that the address is not zero.
CrossChain: the function is invoked in BO, and the arguments have been checked before.
Slither: Acknowledged.

4.3.2 calls inside a loop

Severity: Low Risk
Context: Multicall.sol#58
Description: Calls inside a loop might lead to a denial-of-service attack.

```
/// @inheritdoc IMulticall
function multicall(bytes[] calldata data) public payable override refundExcessNative
    results = new bytes[](data.length);
    for (uint256 i = 0; i < data.length; i++) {
        (bool success, bytes memory result) = address(this).delegatecall(data[i]);

        if (!success) {
            revertFromReturnedData(result);
        }

        results[i] = result;
    }
}
```

Recommendation: Favor pull over push strategy for external calls.
CrossChain: it's multicall entry
Slither: Acknowledged.

4.3.3 reentrancy vulnerabilities

Severity: Low Risk
Context: SwapperV2.sol#141-184, SwapperV2.sol#192-235, WithdrawFacet.sol#79-87, WithdrawFacet.sol#34-56, ServiceFeeCollector.sol#122-140, FeeCollector.sol#106-151, ServiceFeeCollector.sol#56-72, FeeCollector.sol#70-101, LibSwap.sol#32-86, CBridgeFacet.sol#124-151
Description: Detects reentrancies that allow manipulation of the order or value of events.

```
function _withdrawAsset(
    address _assetAddress,
    address _to,
    uint256 _amount
) internal {
    address sendTo = (LibUtil.isZeroAddress(_to)) ? msg.sender : _to;
    LibAsset.transferAsset(_assetAddress, payable(sendTo), _amount);
    emit LogWithdraw(_assetAddress, sendTo, _amount);
}
```

Recommendation: use the Checks-Effects-Interactions pattern to avoid re-entrancy.
CrossChain: the order of events does not affect the functionality of the business.
Slither: Acknowledged.