LI.FI Security Review

AcrossFacetPackedV3(v1.2.0), ReceiverAcrossV3(v1.0.1)

Independent Review By:

Sujith Somraaj (somraajsujith@gmail.com)

December 6, 2024

Contents

1 About Researcher

Sujith Somraaj is a distinguished security researcher and protocol engineer with over seven years of comprehensive experience in the Web3 ecosystem.

In addition to working as an external auditor/security researcher with LI.FI, Sujith is a protocol engineer and security researcher at Superform and Spearbit.

Learn more about Sujith on sujithsomraaj.xyz

2 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release, and does not give any warranties on finding all possible security issues of that given smart contract(s) or blockchain software. i.e., the evaluation result does not guarantee against a hack (or) the non existence of any further findings of security issues. As one audit-based assessment cannot be considered comprehensive, I always recommend proceeding with several audits and a public bug bounty program to ensure the security of smart contract(s). Lastly, the security audit is not an investment advice.

This review is done independently by the reviewer and is not entitled to any of the security agencies the researcher worked / may work with.

3 Scope

- src/Periphery/ReceiverAcrossV3.sol(v1.0.1)
- src/Facets/AcrossFacetPackedV3.sol(v1.2.0)

4 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

4.1 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.

4.2 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

4.3 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

5 Executive Summary

Over the course of 1 days in total, LI.FI engaged with the researcher to audit the contracts described in section 3 of this document ("scope").

This is a follow-on review, intended to focus on the hotfix made to the AcrossFacetPackedV3 and ReceiverAcrossV3 contracts in lifinance/contracts/pull/897

In this period of time a total of 8 issues were found.

Project Summary	
Project Name	LI.FI
Repository	lifinance/contracts
Commit Hashes	5f59b4957f.....85ccff488c
Type of Project	
Audit Timeline	December 6, 2024
Methods	Manual Review

Issues Found	
Critical Risk	0
High Risk	0
Medium Risk	1
Low Risk	3
Gas Optimizations	0
Informational	4
Total Issues	**8**

6 Findings

6.1 Medium Risk

6.1.1 Non-EOA bridging via non-packed functions breaks `speedUpV3Deposit`

Context: AcrossFacetPackedV3.sol#L120, AcrossFacetPackedV3.sol#L190

Description: When bridging through non-packed functions (**startBridgeTokensViaAcrossV3NativeMin**, **startBridgeTokensViaAcrossV3ERC20Min**), the depositor is set to `msg.sender`:

```
spokePool.depositV3(
    msg.sender, // depositor
    _parameters.receiver,
    sendingAssetId,
    // ...
);
```

If **msg.sender** is a contract, Across's **speedUpV3Deposit** function cannot be used since it requires the original depositor to sign messages, which non-EOA accounts cannot do. This may result in varied behavior among the different functions within the same `AcrossFacetPackedV3` contract.

Recommendation: Consider adding a new parameter called **depositor** to allow non-EOA users to safely use the non-packed functions by accessing the **speedUpV3Deposit** functionality of Across.

LI.FI: Fixed in 9a8ff404186a7df19d9dd5610a94b977bbfc233d

Researcher: Verified fix.

6.2 Low Risk

6.2.1 Invalid inputs could cause unexpected behavior in `AcrossFacetPackedV3`

Context: AcrossFacetPackedV3.sol#L15

Description: Multiple functions in `AcrossFacetPackedV3.sol` lack basic parameter validation:

```
function startBridgeTokensViaAcrossV3NativePacked() external payable {
    // No validation of msg.data length
    // No zero-address checks
    // No amount validation
}

function startBridgeTokensViaAcrossV3ERC20Min(
    PackedParameters calldata _parameters,
    address sendingAssetId,
    uint256 inputAmount
) external {
    // No validation for:
    // - zero addresses (_parameters.receiver, sendingAssetId)
    // - zero amounts (inputAmount)
    // - timestamp validity (_parameters.quoteTimestamp, fillDeadline)
    // - valid chain ID
}
```

Improper handling of **msg.data** / **calldata** can lead to unexpected behavior for users, including the potential loss of funds. Though, the functions lack validations from a gas stand point, basic sanity checks protects user funds.

Recommendation: Consider validating the input parameters for all functions, including:

- startBridgeTokensViaAcrossV3NativePacked
- startBridgeTokensViaAcrossV3NativeMin

- startBridgeTokensViaAcrossV3ERC20Packed
- startBridgeTokensViaAcrossV3ERC20Min

LI.FI: Acknowledged. Facet is only meant to be used with our API and **calldata** generated from the API.

Researcher: Users directly calling this contract should be more careful about the generation of the call data and is suggested to use the encode functions provided.

6.2.2 Bad swapData in `handleV3AcrossMessage` will leave dust in the `ReceiverAcrossV3`

Context: ReceiverAcrossV3.sol#L110

Description: If the **swapData** in **handleV3AcrossMessage** fails to utilize all the received bridged tokens, some will remain in the contract. These tokens must be transferred or recovered manually by the admin, which is a task-intensive and slow process.

Recommendation: Consider sweeping unused tokens at the end of the transaction to the receiver as following:

```
function _swapAndCompleteBridgeTokens(
    bytes32 _transactionId,
    LibSwap.SwapData[] memory _swapData,
    address assetId,
    address payable receiver,
    uint256 amount
) private {
  ....
  // refund unused tokens
  uint256 dust = assetId.balanceOf(address(this);
  if(dust > 0) assetId.safeTransfer(receiver, dust);
  // reset approval to 0
  assetId.safeApprove(address(executor), 0);
}
```

LI.FI: Acknowledged in favor of gas-saving

Researcher: Consider adding documentation surrounding this issue to communicate this kind-of expected behavior to the users.

6.2.3 Emit events for critical state changing functions

Context: ReceiverAcrossV3.sol#L78, AcrossFacetPackedV3.sol#L71

Description: Multiple critical functions lack events in the two contracts under audit scope.

- The **pullToken** function in `ReceiverAcrossV3.sol` handle refunds for locked tokens. However, this function does not emit events, making tracking admin withdrawals difficult.
- The **setApprovalForBridge** function in `AcrossFacetPackedV3.sol` pre-approves the bridge contract to save on gas. However, there is no events associated with this function call, which might be useful for monitoring purpose.

Recommendation: Consider emitting an **event** for the above-mentioned functions to ensure smooth off-chain tracking and monitoring.

LI.FI: Acknowledged. I don't believe we care to track this.

Researcher: Might be needed in future requirements for onchain monitoring. Good to add though.

6.3 Informational

6.3.1 Re-entrancy protection in `handleV3AcrossMessage`

Context: ReceiverAcrossV3.sol#L51

Description: While there are currently no practical re-entrancy attack vectors imaginable due to the implementation of the **nonReentrant** modifier in the SpokePool contract of Across before calling the receiver (ReceiverAcrossV3) and trusted components, it is still recommended to implement **nonReentrant** as a security best practice.

The security benefit from implementing re-entrancy guards is comparable to the slightly increased gas costs.

Recommendation: Consider utilizing OpenZeppelin's Re-entrancy Guard library and apply the `nonReentrant` modifier to the **handleV3AcrossMessage** function.

LI.FI: Acknowledged.

Researcher: Though there are no imminent risks, adding a modifier would've made the implementation more robust.

6.3.2 Use `safeTransferETH` to transfer native asset

Context: ReceiverAcrossV3.sol#L85

Description: The **pullToken** function in `ReceiverAcrossV3.sol` is used to refund native tokens using the **call** method. However, switching to safeTransferETH from the SafeTransferLib is considered generally safe and gas efficient.

Recommendation: Consider using **safeTransferETH** from solady library to send native tokens.

LI.FI: Acknowledged.

Researcher: Acknowledged.

6.3.3 `receive` fallback in `ReceiverAcrossV3` is redundant

Context: ReceiverAcrossV3.sol#L136

Description: The **receive()** fallback allows a smart contract to accept native tokens. Since the contract doesn't require native tokens, accepting them may lead to unwanted transfers, which would need to be refunded using the **pullToken** function.

Furthermore, the cross-spoke pool will never send native tokens to a contract; it only transfers the wrapped version. Removing the **receive()** function will decrease wrongful transactions of native tokens to the receiver peripheral contract.

Recommendation: Consider removing the **receive()** function

LI.FI: Acknowledged.

Researcher: Acknowledged.

6.3.4 Using `safeApproveWithRetry` instead of resetting and approving using `safeApprove`

Context: ReceiverAcrossV3.sol#L108-L109

Description: The **_swapAndCompleteBridgeTokens** function in `ReceiverAcrossV3.sol` initially sets the approval to zero and later sets the approvals to a stipulated value using the **safeApprove** function. The **SafeTransferLib handles this operation** through the **safeApproveWithRetry**, which is optimized and clean.

Recommendation: Consider using the **safeApproveWithRetry** as following:

```
- assetId.safeApprove(address(executor), 0);
- assetId.safeApprove(address(executor), amount);
+ assetId.safeApproveWithRetry(address(executor), amount);
```

LI.FI: Acknowledged.

Researcher: Acknowledged.