# LI.FI Security Review

AcrossFacetV3, AcrossFacetPackedV3, ReceiverAcrossV3

**Independent Review By:**

Sujith Somraaj (somraajsujith@gmail.com)

October 7, 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/Facets/AcrossFacetV3.sol

- src/Facets/AcrossFacetPackedV3.sol

- • src/Periphery/ReceiverAcrossV3.sol

# 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").

In this period of time a total of 11 issues were found.

| Project Summary | |
|---|---:|
| Project Name | LI.FI |
| Repository | lifinance/contracts |
| Commit Hashes | ef2d7bc478c....7864124fa85e |
| Type of Project | |
| Audit Timeline | September 16, 2024 |
| Methods | Manual Review |

| Issues Found | |
|---|---|
| Critical Risk | 0 |
| High Risk | 0 |
| Medium Risk | 1 |
| Low Risk | 3 |
| Gas Optimizations | 0 |
| Informational | 7 |
| **Total Issues** | **11** |

# 6 Findings

## 6.1 Medium Risk

### 6.1.1 Receiver address could be `address(0)` when destination call flag is enabled in `_startBridge` function.

**Context:** AcrossFacetV3.sol#L118

**Description:** In the `_startBridge` function of the `AcrossFacetV3` facet contract, the receiver address validation has an edge case allowing `receiverAddress` to be `address(0)` when the destination flag is enabled.

All bridging functions in the contract leverage the `validateBridgeData` modifier from the validatable contract to ensure the receiver address is not `address(0)`.

This ensures that the receiver address in bridge data is not `address(0)`; however, the receiver address passed to the bridge is the across-data receiver address. This cannot be zero if the `hasDestinationCall` is not enabled as the below validation will ensure the address cannot be zero.

```
if (!_bridgeData.hasDestinationCall &&
        (_bridgeData.receiver != _acrossData.receiverAddress)) revert InformationMismatch();
```

But if the `hasDestinationCall` flag is enabled, this check becomes redundant, allowing the user to bridge to zero address on the across facet.

Also, it emits a misleading event. In that case, the receiver address is passed in as `bridgeData.receiver`; however, the receiver is the across-receiver address.

**Recommendation:** Consider adding additional checks to make sure the address is not zero. Fix the event using `_acrossData.receiverAddress` as the final receiver address, overriding the `_bridgeData.receiver`

**Likelihood: LOW + Impact: MEDIUM = Severity: MEDIUM**

**LI.FI:** We provide reasonable guardrails to prevent inadvertent problems. If someone passes address(0) on purpose that is really on them.

**Researcher:** This validation could be a vital one to protect against user error, but however as the team decided against it and also their argument is valid, acknowledging it.

## 6.2 Low Risk

### 6.2.1 Remove `payable` keyword in ERC20 bridging functions of `AcrossFacetPackedV3`

**Context:** AcrossFacetPackedV3.sol#L135, AcrossFacetPackedV3.sol#L176

**Description:** The functions `startBridgeTokensViaAcrossV3ERC20Min` and `startBridgeTokensViaAcrossV3ERC20Packed` are used to bridge ERC20 tokens using across bridge.

These two functions are marked as `payable`, which will reduce gas overhead slightly. However, it will introduce a new issue: excess native tokens sent along these function calls are locked in the contract, which could be abused/used by other facets where there are no native token checks.

**Recommendation:** Consider removing the `payable` keyword.

**LI.FI:** Fixed in ddc45f13a2007025fb62f8983d417b9a1ed233d4

**Researcher:** Validated fix. Looks all good.

### 6.2.2 Add documentation about referrer address to calldata encoding functions in `AcrossFacetPackedV3`

**Context:** AcrossFacetPackedV3.sol#L267, AcrossFacetPackedV3.sol#L223

**Description:** The functions `encode_startBridgeTokensViaAcrossV3NativePacked` and `encode_startBridgeTokensViaAcrossV3ERC20Packed` are used by users to encode parameters to `msg.data` to be passed

to the `AcrossFacetPackedV3`. This function, however, has no documentation/information about the referrer's address.

Hence, if the user directly uses the output of this function to the facet, it'll result in unexpected behavior, as the extra 28-byte data will be cut off from the message / msg.data.

**Recommendation:** Add clear documentation/warnings about its usage and the need to encode the referrer address off-chain. Or encode an empty 28 bytes at the end to avoid such unexpected behavior.

**LI.FI:** Referrer handling was removed in `f8cb0d8c4bfdba35686e63849095f63c516c5784`

**Researcher:** Fix validated. Looks all good.

### 6.2.3 `transactionId` **casting in**`encode_startBridgeTokensViaAcrossV3NativePacked` **and** `encode_start-BridgeTokensViaAcrossV3ERC20Packed` **will lead to data losses**

**Context:** AcrossFacetPackedV3.sol#L245, AcrossFacetPackedV3.sol#L296

**Description:** The functions `encode_startBridgeTokensViaAcrossV3NativePacked` and `encode_start-BridgeTokensViaAcrossV3ERC20Packed` in `AcrossFacetPackedV3` accepts a `bytes32` transactionId and cast it to `bytes8`, which will result in 75% loss in information.

```
bytes32 => 0x1234567890abcdef1234567890abcdef1234567890abcdef1234567890abcdef
bytes8 => 0x1234567890abcdef
```

As a result, the 'LiFiAcrossTransfer' event emits a transaction ID that will be misleading / not informative enough to track the transaction.

**Recommendation:** Consider using the `bytes32` transaction ID as it is (or) avoid it as the information could be misleading/redundant.

**LI.FI** This is a known issue and was implemented to optimize gas usage. We are aware of the implications and accept the information loss. Thanks for pointing it out though.

**Researcher:** Acknowledged.

## 6.3 Informational

### 6.3.1 Add exclusive relayer support

**Context:** AcrossFacetPackedV3.sol#L85, AcrossFacetPackedV3.sol#L123, AcrossFacetPackedV3.sol#L155, AcrossFacetPackedV3.sol#L204

**Description:** The Across protocol supports a feature called `exclusive relayer` where the execution can be limited for certain relayers to fill the user bridging intents.

However, this feature is not supported in the current `AcrossV3Facet` implementation and is hard coded to `address(0)`, which prevents the user from using it.

**Recommendation:** Consider allowing the user to specify the exclusive relayer while bridging.

**LI.FI:** Fixed in 3ebb9d747dbed87ca69939fa53db5c8b16c0e557

**Researcher:** Verified.

### 6.3.2 **Optimize** `pullToken` **function in** `ReceiverAcrossV3`

**Context:** ReceiverAcrossV3.sol#L90

**Description:** If the recipient address is a contract, i.e., ReceiverAcrossV3.sol, the spoke pool contract will always send the wrapped asset. Hence, there is no requirement to have the `receive()` function and `pullToken` functions in the `ReceiverAcrossV3` contracts as they'll be primarily redundant and won't be used.

**Recommendation:** Consider removing the native token handling int the `pullToken` function and removing the `receive` fallback.

Please refer: https://vscode.blockscan.com/ethereum/0x08c21b200ed06d2e32cec91a770c3fca8ad5f877# L1223 for more information

**LI.FI:** Not every DEX call supports sending funds/refunds directly to the user, so we prefer to keep the flexibility, especially since it does not add any risk. I will close this issue for now. Thanks for pointing it out, though.

**Researcher:** Acknowledged.

### 6.3.3 Inconsistent cacheGasLeft check in `_swapAndCompleteBridgeTokens` in `ReceiverAddressV3.sol`

**Context:** ReceiverAcrossV3.sol#L134, ReceiverAcrossV3.sol#L117

**Description:** The `_swapAndCompleteBridgeTokens` function in the `ReceiverAddressV3` facet has an inconsistent gas check in the following two lines,

```
....
if (cacheGasLeft < recoverGas) {
    // case A: not enough gas left to execute calls
    // @dev: we removed the handling to send bridged funds to receiver in case of insufficient gas
    //       as it's better for AcrossV3 to revert these cases instead
    revert InsufficientGasLimit(cacheGasLeft);
}
....
{} catch {
    cacheGasLeft = gasleft();
    if (cacheGasLeft <= recoverGas)
            revert InsufficientGasLimit(cacheGasLeft);
....
```

The first check reverts only if the `cacheGasLeft` is strictly less than the `recoverGas`. But the second check reverts if the `cacheGasLeft` is less than or equals the `recoverGas`. This also contradicts other facets like the `StargateV2Facet` where there is no check in the catch blocks of the swap.

**Recommendation:** Make the check uniform to ensure everything is clear. As in both cases, the required gas should be greater than the `recover` to handle the transfer of tokens to the user.

**LI.FI:** The first gas check intentionally reverts instead of just sending the tokens to the user. This is specific for AcrossV3. Initially we had it like in the other receiver contracts but the Across relayers will price the calldata on src chain and then execute it on dst. For some reason they would often estimate insufficiently and a lot of the dst calls would just never go into execution but the bridged tokens would be refunded to the users which is strongly undesired behaviour. It was the Across team that suggested this change to force a revert so that relayers cannot use this "simple path".

The second gas check on the other hand is designed to catch cases where the dst call ran into out-of-gas during its execution. We wanted to be able to differentiate this case from all other "revert" cases (e.g. slippage or any other reverts in third party contracts). Therefore this check is designed to catch out-of-gas errors only and then revert, for the same reason as above: to force relayers to estimate gas sufficiently or not be able to execute the tx at all.

**Researcher:** Acknowledged; additional documentation added in `bda55b5248c48a8ed798cf85d17adb689fede651`

### 6.3.4 `ACROSS_REFERRER_DELIMITER` in `AcrossFacetPackedV3` contract could be `bytes8`

**Context:** AcrossFacetPackedV3.sol#L19

**Description:** The constant value `ACROSS_REFERRER_DELIMITER` in the `AcrossFacetPackedV3` contract is exactly 8 bytes long. Using the more precise bytes8 type would better represent the data and potentially save gas.

**Recommendation:** Consider changing `ACROSS_REFERRER_DELIMITER` from `bytes` to `bytes8`

**LI.FI** Fixed in `4e59f570b3412ced04b24e96912a6d14e74805b8`

**Researcher:** Fix validated. Looks good.


### 6.3.5    Remove unused `SafeERC20` import in `AcrossFacetPackedV3.sol`

**Context:** [AcrossFacetPackedV3.sol#L9](AcrossFacetPackedV3.sol#L9)

**Description:** The`SafeERC20` contract is imported from the openzeppelin library but is not used anywhere in the code in the `AcrossFacetPackedV3` contract.

**Recommendation:** Consider removing the unused import.

**LI.FI:** Fixed in `f073ea9954660c3c29095c180b1932922159f6f4`

**Researcher:** Fix validated. Looks all good.


### 6.3.6    `handleV3AcrossMessage` is not payable

**Context:** [ReceiverAcrossV3.sol#L58](ReceiverAcrossV3.sol#L58)

**Description:** ReceiverAcrossV3 implements `handleV3AcrossMessage` function to receive arbitrary messages from across v3 through its spoke pool contract. However, this function is not payable per across implementation, but ReceiverAcrossV3 declared the function `payable`.

Refer on [blockscan](blockscan)

**Recommendation:** Consider removing the `payable` keyword from `handleV3AcrossMessage` function in `Receiver-AcrossV3` contract.

**LI.FI:** Fixed in `c87faf6dc3aea2bcfc9126143da0ef51e92e49c0`

**Researcher:** Validated fix. Looks all good.


### 6.3.7    Remove unused `ERC20` import in `ReceiverAcrossV3`

**Context:** [ReceiverAcrossV3.sol#L11](ReceiverAcrossV3.sol#L11)

**Description:** The `ERC20` contract is imported from `solady` library but is not used anywhere in the code.

**Recommendation:** Consider removing the unused import.

**LI.FI:** Fixed in `554b40fd76c465ab8463581d7a2656797ee6fedd`

**Researcher:** Fix validated. Looks good.