



LiFilIntentEscrowFacet v1.0.0 Security Review

Reviewed by: Goran Vladika

17th November - 18th November, 2025

LiFiIntentEscrowFacet v1.0.0 Security Review Report

Burra Security

November 20, 2025

Introduction

A time-boxed security review of the **LI.FI** protocol was done by **Burra Security** team, focusing on the security aspects of the smart contracts.

Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource, and expertise-bound effort where we try to find as many vulnerabilities as possible. We can not guarantee 100% security after the review or even if the review will find any vulnerabilities. Subsequent security reviews, bug bounty programs, and on-chain monitoring are recommended.

About Burra Security

Burra Sec offers security auditing and advisory services with a special focus on cross-chain and interoperability protocols and their integrations.

Security review team

Goran Vladika is a security researcher and smart contract engineer with five years of experience in the blockchain industry. After beginning his Web3 career in the DeFi space, Goran joined Offchain Labs as a blockchain engineer, where he contributed to the core smart contract components of Arbitrum. His work included the design, implementation, and security of Arbitrum's native bridge, token bridge

and rollup stack, critical infrastructure that secures billions of dollars in TVL. This bridging technology has since been adopted by dozens of applications and L2 and L3 chains built using the Arbitrum Orbit stack. Goran's experience building cross-chain systems at both the protocol and application layers has provided him with a strong foundation in blockchain security. As a security researcher, he has helped secure leading projects in the interoperability space including Centrifuge, LiFi, PancakeSwap, ZetaChain and DODO, as well as L1/L2 protocols such as Telcoin and Citrea.

About LiFiIntentEscrowFacet v1.0.0

LI.FI Intent Escrow uses a built in escrow as a deposit mechanism for its intents. The LI.FI Intent Escrow Facet deposits into the Escrow Input Settler, which will release the deposited funds to the solver when the fill has been proven. The system is highly self serve, with the facet wrapping the deposit logic to ensure the appropriate parameters are called for the user to receive their output.

Severity classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

Impact - The technical, economic, and reputation damage from a successful attack

Likelihood - The chance that a particular vulnerability gets discovered and exploited

Severity - The overall criticality of the risk

Informational - Findings in this category are recommended changes for improving the structure, usability, and overall effectiveness of the system.

Security Assessment Summary

review commit hash - 4e71699761d075f43f5d9cd66c4e1be082fa08ee

Scope

The following smart contracts were in the scope of the audit:

- src/Facets/LiFiIntentEscrowFacet.sol
 - src/Interfaces/IOpenIntentFramework.sol
 - src/Interfaces/IOrginSettler.sol
-

Findings Summary

ID	Title	Severity	Status
M-1	Users lose all positive slippage from swaps when creating intent orders	Medium	Resolved
L-1	Destination calls can be provided despite not being supported	Low	Resolved
I-1	Missing natspec for order's nonce	Info	Resolved
I-2	Facet restricts destination to EVM chains only	Info	Resolved

Detailed Findings

[M-01] Users lose all positive slippage from swaps when creating intent orders

Target

- LiFiIntentEscrowFacet.sol#L111

Severity

- Impact: Medium
- Likelihood: Medium

Description

[LiFiIntentEscrowFacet](#) offers users 2 ways to create intent orders for the LiFi Intents protocol:

- Direct bridging: deposit tokens directly into the intent escrow
- Swap and bridge: first swap tokens, then deposit the output into the intent escrow

The swap and bridge feature introduces an issue with how swap profit (ie. positive slippage or any swap outcome greater than minimum price) is handled. Users (or LiFi backend) first obtains an off-chain quote that specifies both input and output amounts, such as “send 2 USDC to receive 1.65 USDC on

destination.”. User then calls `swapAndStartBridgeTokensViaLiFiIntentEscrow()` with their minimum acceptable amount (2 USDC). If the swap execution is favorable and returns 2.15 USDC instead of 2 USDC, the code deposits this entire amount into the intent order:

```
1  _bridgeData.minAmount = _depositAndSwap(...); // Returns actual swap
    output: 2.15 USDC
2  _startBridge(_bridgeData, _lifiIntentData);    // Creates order with
    2.15 USDC input
```

However, the output amount remains fixed at 1.65 USDC from the original quote, because `_lifiIntentData.outputAmount` is set before the swap and never adjusted. This means the solver will: receive 2.15 USDC from escrow, but provide only 1.65 USDC on destination (as originally quoted). The solver captures the entire swap profit of 0.15 USDC as additional profit beyond their quoted spread. Users systematically lose all swap profits above their min accepted price.

Recommendation

Return excess tokens from swap outcome to the user rather than including them in the intent order:

```
1  function swapAndStartBridgeTokensViaLiFiIntentEscrow(...) external {
2      uint256 swapOutcome =
3          _depositAndSwap(_bridgeData.transactionId, _bridgeData.
              minAmount, _swapData, payable(msg.sender));
4
5      // Return positive slippage to user if any
6      if (swapOutcome > _bridgeData.minAmount) {
7          uint256 excess = swapOutcome - _bridgeData.minAmount;
8          LibAsset.transferAsset(_bridgeData.sendingAssetId, payable(
              msg.sender), excess);
9      }
10
11      _startBridge(_bridgeData, _lifiIntentData);
12 }
```

This ensures users benefit from favorable swap execution while maintaining compatibility with the off-chain quote matching system.

Client

Fixed: <https://github.com/lifinance/contracts/pull/1361/commits/a56c940cbeb562456f7921b1f89f2319efc73b83>

BurraSec

Fix verified

[L-01] Destination calls can be provided despite not being supported

Target

- LiFiIntentEscrowFacet.sol#L158

Severity

- Impact: Medium
- Likelihood: Low

Description

LiFi's integration explicitly prohibits destination calls through the `doesNotContainDestinationCalls` (`_bridgeData`) modifier on both entry functions. This modifier ensures that `_bridgeData.hasDestinationCall` is set to false, signaling to users that destination callbacks are not supported by this integration.

However, the implementation still accepts and forwards arbitrary callback data through the `_lifiIntentData.outputCall` parameter without any validation:

```
1 function startBridgeTokensViaLiFiIntentEscrow(...)
2     external
3     doesNotContainDestinationCalls(_bridgeData) // Explicitly forbids
        destination calls
4 {
5     // ...
6     outputs[0] = MandateOutput({
7         // ...
8         callbackData: _lifiIntentData.outputCall, // But still passes
            user's callback data
9         // ...
10    });
11 }
```

Recommendation

Since destination calls are not yet supported, add validation to ensure `outputCall` is empty.

Client

Fixed: <https://github.com/lifinance/contracts/commit/67d51637efd974878b4851290e71a3ac6a1082b7>
Destination calls will be soon supported by LiFi intent protocol

BurraSec

Fix verified

[I-01] Missing natspec for order's nonce**Target**

- LiFiIntentEscrowFacet.sol#L50

Severity

INFO

Description

Field `nonce` of struct `LiFiIntentEscrowData` is missing the natspec description.

Recommendation

Add missing natspec

Client

Fixed: <https://github.com/lifinance/contracts/pull/1361/commits/4f8a64f3bd9635f5cd68f3e2be9389d6c0a69e28>

BurraSec

Verified

[I-02] Facet restricts destination to EVM chains only

Target

- LiFiIntentEscrowFacet.sol#L131

Severity

INFO

Description

The underlying intent protocol is designed to support multiple blockchain types, as evidenced by its use of `bytes32` for cross-chain identifiers instead of EVM-specific `address`:

```
1 struct MandateOutput {
2     bytes32 oracle;
3     bytes32 settler;
4     bytes32 token;
5     bytes32 recipient;
6     ...
```

This design choice allows the protocol to handle both EVM addresses (20 bytes) and non-EVM addresses like Solana (32 bytes). However, LiFi's integration adds an EVM-specific validation that breaks this multi-chain capability:

```
1 if (address(uint160(uint256(_lifiIntentData.receiverAddress))) !=
   _bridgeData.receiver) {
2     revert InvalidReceiver();
3 }
```

Users attempting to bridge to Solana or other non-EVM chains will have their transactions revert. Even if Solana is not currently supported as the destination, the address check could still be performed in a way so it doesn't break once Solana does get supported.

Recommendation

Do EVM specific address check only if destination chain is EVM compatible, for example:

```
1 if (_bridgeData.receiver != LiFiData.NON_EVM_ADDRESS) {
2     if (address(uint160(uint256(_lifiIntentData.receiverAddress))) !=
        _bridgeData.receiver) {
3         revert InvalidReceiver();
```

```
4     }  
5 }
```

Client

Fixed: <https://github.com/lifinance/contracts/pull/1361/commits/16d7fa45c916158a6687d3a3a22727f9e8c973fc>

BurraSec

Verified