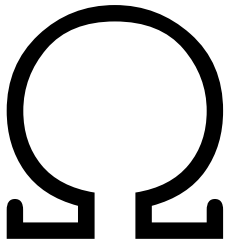
Backed CCIPReceiver Update

Final Audit Report

November 5, 2025



Team Omega

Teamomega.eth.limo

Summary

Backed Finance has asked Team Omega to audit an update of their contracts that define the behavior of a rebasing token and factory.

We found **no high severity issues** - these are issues that can lead to a loss of funds, and are essential to fix. We classified **2** issues as “medium” - these are issues we believe you should definitely address. We did classify **2** issues as “info” - we believe the code would improve if these issues were addressed as well.

After receiving a first version of this report, the Backed team addressed all issues.

Severity	**Number of issues**	**Number of resolved issues**
High	0	0
Medium	2	2
Low	0	0
Info	2	2

Scope of the Audit

The audit concerns an upgrade of the Backed token contract, specifically the following PR:

```
https://github.com/backed-fi/backed-ccip-contract/pull/14/files
```

The audit regards changes to the following files:

```
contracts/BackedCCIPReceiver.sol
contracts/interfaces/IBackedAutoFeeTokenImplementation.sol
```

Specifically, we will audit the files at the following commit:

```
d737e446a7c6951471bb10ac9ed9a16128381fd5
```

We audited a previous version of this code before - the audit report is here:

https://github.com/OmegaAudits/audits/blob/main/202510-Backed-Token-Bridge-Update.pdf

The last commit we audited was

```
9f93c8f548b943ab2cb8572d22c47764fb271699
```

This audit concerns changes since that last update. We audited the code at the following commit:

```
d737e446a7c6951471bb10ac9ed9a16128381fd5
```

Resolution

We delivered a preliminary report on October 29, 2025. The issues were subsequently addressed in the following commit:

```
3e68d60fe4751bc968325170e19187154c15a422
```

We updated each of the issues below

Methods Used

The contracts were compiled, deployed, and tested in a test environment.

Code Review

We manually inspected the source code to identify potential security flaws.

Automatic analysis

We have used static analysis tools to detect common potential vulnerabilities. No high severity issues were identified with the automated processes. Some low severity issues were found, and we have included them below in the appropriate parts of the report.

Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of

the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

Severity definitions

High	Vulnerabilities that can lead to loss of assets or data manipulations.
Medium	Vulnerabilities that are essential to fix, but that do not lead to assets loss or data manipulations
Low	Issues that do not represent direct exploit, such as poor implementations, deviations from best practice, high gas costs, etc
Info	Matters of opinion

Findings

BackedCCIPReceiver

BR1. Code expects ERC20.transferSharesFrom to return False on failure [medium] [resolved]

The function `_ccipReceive` tests if the call to to transfer shares on line 493 returns True. If it does not, it will emit an event.
However, the function `transferSharesFrom` in the `BackedTokenImplementation` will never return false on failure, but instead will revert on error in case the transfer fails, so the event will never be emitted on failure, and the code won't work as expected.
Recommendation: Do a `try` call to `transferSharesFrom` and handle the revert case.

Severity: Medium
Resolution: The code now reverts if the call to `transferSharesFrom` reverts.

BR2. Retry of message will not be possible if token transfer fails [medium] [resolved]

The `_ccipReceive` function is now meant to emit an event instead of reverting when token transfer has failed. This may not be desirable though, since there could be many reasons for a transfer failure, and it is likely that at least in some cases allowing retry would still be desirable.
Recommendation: Revert on token transfer failure instead of emitting the `InvalidMessageReceived` event.
Severity: Medium
Resolution: The issue was resolved as recommended

BR3. TokenInfo can be replaced with a simple uint64 [info] [resolved]

The `TokenInfo` struct now holds only a single variable - the `uint64 id`. The use of a struct for holding a single variable is completely unnecessary, and so it could be replaced with a simple `uint64` variable.
Recommendation: Replace the `TokenInfo` struct with just a `uint64` id.
Severity: Info
Resolution: The issue was resolved as recommended

BR4. Remove MULTIPLIER_MISMATCH from InvalidMessageReason [info] [resolved]

The `MULTIPLIER_MISMATCH` type in `InvalidMessageReason` is no longer used in the code and can be removed.
Recommendation: Remove `MULTIPLIER_MISMATCH` from `InvalidMessageReason`.
Severity: Info
Resolution: The issue was resolved as recommended