Backed Token Bridge Upgrade

Final Audit Report

September 19, 2025

Team Omega

Teamomega.eth.limo

Summary

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

We found **no high severity issue** - these are issues that can lead to a loss of funds, and are essential to fix. We classified **no** issues as “medium” - these are issues we believe you should definitely address. We did classify **1** issue as “low”, and an additional **6** issues were classified as “info” - we believe the code would improve if these issues were addressed as well.

Severity	**Number of issues**	**Number of resolved issues**
High	0	-
Medium	1	1
Low	1	1
Info	6	3

Scope of the Audit

Scope of the Audit
The audit concerns an upgrade of the Backed token contracts and the CCIP bridge.

This code is developed in the following repository:

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

The last version of the code we audited, in the report at https://github.com/OmegaAudits/audits/blob/main/202410-Backed-token-bridge.pdf, was

```
05e0ea7d87d914f02cca9acf6bf64056eddf8299
```

The current report regards the changes with respect to that commit and the following commit

```
f617d59c1e4897648650cabb2a8ec88121bb320f
```

Resolution

The issues we found were addressed in following commit:

```
9f93c8f548b943ab2cb8572d22c47764fb271699
```

We reviewed all the changes and marked the resolution of 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. We also inspected the context in which the contracts were developed, namely the Chainlink Bridge architecture.

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

General

G1. Pin solidity dependencies in package.json [low] [resolved]

It is desirable to have a deterministic outcome when compiling smart contracts. This makes the output of the compiler more predictable, which helps when verifying contract deployments, and reduces the chance of introducing new vulnerabilities when rebuilding the package.

Currently, `package.json` specifies a range for the OpenZeppelin dependency:

```
"@openzeppelin/contracts-upgradeable": "^5.4.0",
```

Recommendation: Pin a specific version of the dependency

```
    "@openzeppelin/contracts-upgradeable": "5.4.0",
```

Severity: Low
Resolution: The issue was resolved as recommended.

G2. No continuous integration [info] [not resolved]

The github repository does not have continuous integration actions defined, which makes it hard to guarantee that the code will compile, and the tests will run, on a standard configuration. Indeed, issue G3 is a direct consequence.
Recommendation: Configure continuous integration and run the tests in a github action.
Severity: Info
Resolution: The issue was not resolved.

G3. Contracts do not compile in the current configuration [info] [resolved]

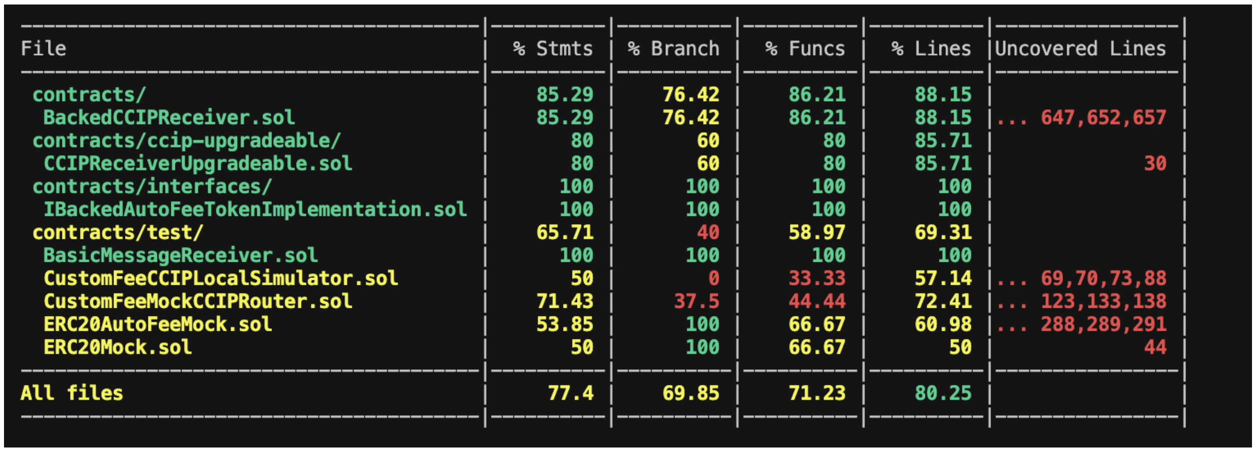
In the current configuration, the contracts do not compile (and the tests can not be run) due to an error in the hardhat configuration file:

```
~/omega/backed/backed-ccip-contract (202509-audit)> npm run compile

> compile
>     hardhat compile

Error HH8: There's one or more errors in your config file:

  * Invalid value
{"chainId":146,"forking":{},"mining":{"auto":true,"interval":1}} for
HardhatConfig.networks.hardhat - Expected a value of type
HardhatNetworkConfig.

To learn more about Hardhat's configuration, please go to
https://hardhat.org/config/

For more info go to https://hardhat.org/HH8 or run Hardhat with
--show-stack-traces
```

Recommendation: Fix the configuration file, and use CI so these errors are avoided (see G2).
Severity: Info
Resolution: The issue was resolved as recommended.

G4.Limited test coverage [info] [not resolved]

The test coverage is generally high, but not complete. It is recommended to reach 100% test coverage to ensure all code is tested.



```
----------------------------------------|----------|----------|----------|----------|----------------|
File                                    |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
----------------------------------------|----------|----------|----------|----------|----------------|
 contracts/                             |    85.29 |    76.42 |    86.21 |    88.15 |                |
  BackedCCIPReceiver.sol                |    85.29 |    76.42 |    86.21 |    88.15 |... 647,652,657 |
 contracts/ccip-upgradeable/            |       80 |       60 |       80 |    85.71 |                |
  CCIPReceiverUpgradeable.sol           |       80 |       60 |       80 |    85.71 |             30 |
 contracts/interfaces/                  |      100 |      100 |      100 |      100 |                |
  IBackedAutoFeeTokenImplementation.sol |      100 |      100 |      100 |      100 |                |
 contracts/test/                        |    65.71 |       40 |    58.97 |    69.31 |                |
  BasicMessageReceiver.sol              |      100 |      100 |      100 |      100 |                |
  CustomFeeCCIPLocalSimulator.sol       |       50 |        0 |    33.33 |    57.14 |... 69,70,73,88 |
  CustomFeeMockCCIPRouter.sol           |    71.43 |     37.5 |    44.44 |    72.41 |... 123,133,138 |
  ERC20AutoFeeMock.sol                  |    53.85 |      100 |    66.67 |    60.98 |... 288,289,291 |
  ERC20Mock.sol                         |       50 |      100 |    66.67 |       50 |             44 |
----------------------------------------|----------|----------|----------|----------|----------------|
All files                               |     77.4 |    69.85 |    71.23 |    80.25 |                |
----------------------------------------|----------|----------|----------|----------|----------------|
```

Recommendation: Increase test coverage to 100%.
Severity: Info
Resolution: The issue was not resolved.

BackedCCIPRecevier.sol

BCR1. Storage layout changed from the last version [medium] [resolved]

In `BackedCCIPReceiver.sol` the storage layout changed with respect to the version we audited previously (`05e0ea7`). Specifically, the following variable was removed from the second slot of the storage layout:

```
uint256 private _defaultGasLimitOnDestinationChain;
```

And the following mapping was added between two existing mappings:

```
mapping(uint64 => ChainInfo) public chainInfos;
```

These contracts are upgradeable. If this new version is deployed as an upgrade to a deployment of the previous version, it will use the original storage layout - this means that the upgraded contract will read data from the wrong storage slot, with unpredictable consequences.
Severity: Medium
Resolution: The team communicated that the new code will not be deployed as an upgrade, but will be a fresh deployment, and so the issue does not apply.

BCR2. Inconsistent use of encode/encodePacked [info] [resolved]

In line 370, `encode` was replaced by `encodePacked`, but in 536 `abi.decode` is still used. This works because the data decoded is 2 `uint256`, so `encode` and `encodePacked` give the same result, but if the structure ever changes, this might break. It is safer to keep using `abi.encode` in line 370, or change the encoding and decoding in both places
Recommendation: Return to use `abi.encode` in line 370.
Severity: Info
Resolution: The issue was resolved as recommended.

BCR3. Clean up data when removing a destination chain [info] [resolved]

The function `removeDestinationChain` does not remove the information in `chainInfos[_destinationChainSelector]`, leaving unnecessary and outdated info in the contract.
Recommendation: When calling `removeDestinationChain`, also remove the corresponding `chainInfo` record. You could also consider merging the two mappings (i.e. `allowlistedDestinationChains` and `chainInfos`).
Severity: Info

Resolution: The issue was resolved as recommended. `chainInfo` entry is now removed when calling `removeDestinationChain`.

BCR4. Inconsistent behavior in view functions [info] [not resolved]

The following public view and setter functions will return values also for destination chains that are not listed.

```
getDeliveryFeeCost
updateGasLimit
gasLimit
```

The behavior of the contracts would be more consistent if these functions would revert when they are being called with unlisted destination chains as arguments.
Recommendation: use the `onlyAllowlistedDestinationChain` modifier on these functions.
Severity: Info
Resolution: The issue was not resolved.