Backed Rebasing Token Update
Solidity Contracts

Final Audit Report

November 6, 2024

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 **1 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 **2** issues as “low” - we believe the code would improve if these issues were addressed as well.

After delivering a first report on October 22, the issues were addressed by the Backed Team. We updated the report accordingly.

Severity	**Number of issues**	**Number of resolved issues**
High	1	1
Medium	0	0
Low	2	2
Info		

Scope of the Audit

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

```
https://github.com/backed-fi/backed-token-contract/pull/36
```

Specifically, commit:

```
85e9019602d2c927d72e966dcbe367f29bdd7c6d
```

This audit is limited to the changes in the following file:

```
contracts/BackedAutoFeeTokenImplementation.sol
```

We audited a previous version of this file. That audit report was published in our repository:

```
https://github.com/OmegaAudits/audits/blob/main/202407-Backed-RebasingTok
ens.pdf
```

The final commit in that audit report was

```
b317bdf1ff9f9db0dfef0407c99bf6f7b1acf12f
```

We audited the changes with respect to that audit.

Resolution

The Backed team subsequently addressed all issues in the following commit

```
cb13ffe50ac4a04e97279762e14420d0a7d522c6
```

We reviewed the changes and updated the report accordingly

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

BackedAutoFeeTokenImplementation

B1. Fees are not applied if newMultiplierActivationTime is in the future [high] [resolved]

If `newMultiplierActivationTime` is set to some time in the future (using `updateMultiplierValue` or `updateMultiplierWithNonce`), then `getCurrentMultiplier` will simply return the `lastMultiplier` value without applying any fees:

```
        if(block.timestamp < newMultiplierActivationTime) {
            return (lastMultiplier, 0, lastMultiplierNonce);
        }
```

This is clearly not a desired side-effect of scheduling an update of the multiplier. Fees should be applied regardless.

Recommendation: The current code is confusing, and seems to use the newly introduced `newMultiplier` state variable both (a) to schedule updates of the multiplier and (b) as the basis of the calculation that applies the fees (in `getCurrentMultiplier`)
We'd recommend refactoring this code and making a clear separation of concerns.
Resolution: With the new changes, it is guaranteed that the `newMultiplierActivationTime` is within the current period, and so `lastMultiplier` takes the fees into account.

B2. Initialize_v3 can be called also after initialization [low] [resolved]

The function `initialize_v3` is meant to be called only on initialization. It tries to enforce this by checking that the value of `newMultiplier` is not equal to 0. However, this check is not sufficient, as the value of `newMultiplier` can be set to 0 also after initialization.
This could be a problem if the `multiplierUpdater` wants to set the multiplier to 0 in the future by setting `newMultiplier`, as an attacker could then call `initialize_v3` and undo the programmed change.
Recommendation: We assume that the multiplier should never be set to 0 - if that is the case, this check can be enforced in `_updateMultiplier`. If instead there is a use case for setting the multiplier to 0, the initialization should be controlled with a separate flag.
Severity: Low
Resolution: A check that enforces that `newMultiplier` can never be set to 0 was added.

B3. Nonce should be checked when new multiplier is activated [low] [resolved]

The nonce provided with the update of the multiplier is used to make sure that in cross-chain messages, the multiplier is applied at the same "time" across chains (where "time" is measured in number of `periodLength` periods plus amount of previous multiplier updates).
There are several points in the code where that behavior is not respected:

a. In `updateMultiplierWithNonce`, it is checked that the Nonce is higher than the current nonce at the moment the function is called (and not when the new multiplier is actually activated, i.e. at `pendingNewMultiplierActivationTime`)
b. In `getCurrentMultiplier`, if a multiplier update is scheduled (i.e. if `newMultiplierActivationTime`), the last-saved nonce is returned

There is also a general problem with using nonces for time measurement together with timestamps - in theory, it is possible that the two run out of sync.

Recommendation: Make sure the Nonce is used and updated as intended throughout the code. Also consider replacing the timestamp-based scheduling with “period-based” scheduling (i.e. do not say “I want the multiplier to update at this date and hour and minute and seconds” but say “I want to update the multiplier after X numbers of periods passed”))
Resolution: These issues were resolved by enforcing that `pendingNewMultiplierActivationTime` falls within the current period