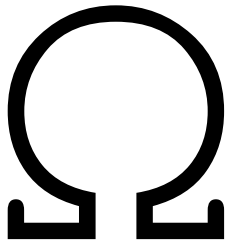
Backed ERC20 Contract

Final Audit Report

May 3, 2022



Team Omega

Teamomega.eth.limo

Summary

Backed Finance has asked Team Omega to audit the contracts that define the behavior of their ERC20 tokens.

We found **no high severity issues** - 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 **6** issues “low”, and an additional **4** issues were classified as “info” - we believe the code would improve if these issues were addressed as well.

Subsequently, Backed Finance addressed all of the issues we found, and no open issues remain.

Severity	**Number of issues**	**Number of resolved issues**
High	0	0
Medium	0	0
Low	6	6
Info	4	4

Scope of the Audit

Code from the following repository was audited:

- `https://github.com/backed-fi/backed-token-contract/commit/80444d9feb995788e5e6cef8947a8838c157e7f5`

And specifically the following Solidity contracts:

- `BackedFactory.sol`
- `BackedTokenImplementation.sol`
- `BackedTokenImplementationV2.sol`
- `ERC20PermitDelegateTransfer.sol`

Resolution

After receiving a preliminary report, Backed has addressed all the issues mentioned in this report in

`https://github.com/backed-fi/backed-token-contract/commit/15dce14123fd7358c907710c5592a22443d3e87d`

The changes were checked by Team Omega, and we have added a description of the resolution to each of the issues

Methods Used

Code Review

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

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

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

Privileged Roles

The Backed token contracts will be deployed as an OpenZeppelin transparent proxy using the BackedFactory helper class. We can identify the following major roles

1. The **Owner of the ProxyAdmin** contract
    a. Can upgrade the proxy
    b. Can transfer ownership of the ProxyAdmin contract to another account, and renounce ownership
2. The **Owner of the BackedFactory** contract
    a. Can deploy new tokens
    b. Can transfer ownership of the BackedFactory contract, and renounce ownership
3. The **Owner of a Token contract instance** can:
    a. Transfer ownership of the Token contract, and renounce ownership
    b. Change the accounts for the roles minter, burner and pauser
    c. Control the whitelist of delegators, or disable the use of the whitelist altogether
4. The **Minter** of a Token contract instance can:
    a. Set the mint allowance, and mint new tokens to any account (this is subject to a time delay)
5. The **Pauser** of a Token contract instance can:
    a. Pause the contract - this will disable the possibility to transfer, mint, or burn new tokens
6. A **whitelisted Delegator** of a Token contract instance can
    a. Execute user-signed approve and transfer transactions for a user
7. The **Burner** of a Token contract instancecan:
    a. Burn tokens that are owned by the burner address or by the token contract

Some of these roles are of critical importance - in the sense that if an account holding such a role is compromised, it can do extreme damage to the value of the token, and a compromise of these accounts would be categorized as “High Severity” on our severity scale.

- The owner of the ProxyAdmin contract can change the behavior of all aspects of the contract
- An account with the minter role can mint tokens that are not backed by collateral in put them in circulation
- An account that holds pauser role can disrupt transfers of the tokens - which, for example in situations of high volatility of the underlying asset - can lead to serious economic damage to token holders
- The owner, of course, controls both the minter and pauser roles

Backed.fi has indicated that these four critical roles will be held by 2 or 3 different Gnosis Multisig contracts, with user-controlled accounts as signers, and that the signers will be urged to use cold wallets to guard their private keys, so that abuse is less likely.
In addition, Backed.fi will consider the use of Gnosis Safe's `TimelockController` which could help to mitigate the damage if any of these role holders are compromised.

General

G1. Licensing information is ambiguous and incomplete [low] [resolved]

There is little information about the licensing in the repository. In `package.json`, there is a reference to theISC license:

```
"license": "ISC",
```

While the solidity files suggest that the code is released under the MIT license:

```
//SPDX-License-Identifier: MIT
```

The repository does not contain other licensing information.
Recommendation: Choose a license, and adapt the files as is appropriate, and include a `LICENSE` file with the complete text of the chosen license
Severity: Low
Resolution: Backed as decided to use the MIT license, and the repository is updated as recommended

G2. No copyright is claimed [low] [resolved]

Only the copyright holder can release software under a particular license. Under some jurisdictions (i.e. the USA) the copyright must be explicitly claimed. This is why most licenses are in the form of a template in which the copyright claim is included - cf. the MIT license:

```
MIT License

Copyright (c) [year] [fullname]

Permission is hereby granted [...]
```

Recommendation: Claim the copyright explicitly - i.e. fill in the company name and year in the licensing template, as well as in each of the solidity files
Severity: Low
Resolution: A copyright claim is added both to the LICENSE file as well as in each of the solidity files

G3. Remove unused solidity dependencies [low] [resolved]

The file `package.json` contains a reference to a version of the package `@openzeppelin/contracts`

```
"dependencies": {
      "@openzeppelin/contracts": "4.5.0",
      "@openzeppelin/contracts-upgradeable": "4.5.2"
}
```

This dependency is to a deprecated version (4.5.0) and is not used anywhere in the code (which uses the upgradeable versions of the contracts).
Recommendation: Remove this dependency, as it is confusing
Severity: Low
Resolution: The unused dependency was removed as recommended

G4. Test Coverage is not complete [low] [resolved]

Test coverage is not 100%

```
----------------------------------|----------|----------|----------|----------|----------------
-|
File                              |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines
|
----------------------------------|----------|----------|----------|----------|----------------
-|
 contracts/                       |      100 |    92.11 |      100 |      100 |
|
  BackedFactory.sol               |      100 |    66.67 |      100 |      100 |
|
  BackedTokenImplementation.sol   |      100 |      100 |      100 |      100 |
|
  BackedTokenImplementationV2.sol |      100 |     87.5 |      100 |      100 |
|
  ERC20PermitDelegateTransfer.sol |      100 |      100 |      100 |      100 |
|
----------------------------------|----------|----------|----------|----------|----------------
-|
All files                         |      100 |    92.11 |      100 |      100 |
|
----------------------------------|----------|----------|----------|----------|----------------
-|
```

Recommendation: We recommend writing tests that cover all lines and branches in the code. This would have avoided issue F1.
Severity: Low
Resolution: Test coverage is now 100%

BackedFactory.sol

F1. Unnecessary checks for trying to deploy twice to the same address [low] [resolved]

The `deployToken` function deploys Token contracts to a predictable address that is based on the name and symbol of the newly created token. It contains the following code that is apparently meant to prevent that a contract is redeployed at the same address:

```
BackedTokenImplementation newToken =
BackedTokenImplementation(address(newProxy));
require(!deployedTokens[address(newToken)], "Factory: Shouldn't deploy
same address");
deployedTokens[address(newToken)] = true;
```

This code can be omitted - an attempt to deploy a contract to the same address will fail in the previous line.
Recommendation: Remove these lines
Severity: Info
Resolution: The issue was resolved as suggested

BackedTokenImplementation.sol

B1. missing __gap state variable [low] [resolved]

The contracts are expected to be deployed as OpenZeppelin transparent upgradeable proxies. It is recommended for this use case to include a state variable (by convention called `__gap`) that is used to reserve storage space in the proxy contract, that can be used at a later date when the contracts are upgraded.
The `__gap` space is not reserved in the `BackedTokenImplementation` contract, which will make it harder to add new state variables in a future upgrade of that contract.
Recommendation: Add the following variable definition to the contract, after all other state variables are defined:

```
uint256[50] private __gap;
```

Resolution: The issue was resolved as suggested

B2. Inconsistent naming [info] [resolved]

The public mapping of the whitelist for delegators is called `delegateWhitelist`, while the function that is used to manage the list is called `setDelegationWhitelist` and the event is named `DelegationWhitelistChange`
Similarly, there is a state variable called `delegateMode` which is set with a function called `setDelegateMode`, while the event is called `DelegationModeChanged`
Recommendation: Use a consistent language in the ABI - i.e. choose between “delegate” or “delegation” but do not use both
Severity: Info
Resolution: The terminology is consistent now

B3. Missing checks for zero address [info] [resolved]

The functions `setMinter`, `setBurner` and `setPauser` do not check if the address is unequal to the zero address (which is the default value that will be used if a caller (by mistake) does not provide an argument).
Recommendation: Add requirements that these addresses are not equal to `address(0)`, as is common practice.
Severity: Info - these functions are callable only by the owner of the contract, which is a trusted address, and are reversible.
Resolution: Zero checks were added to each of the three functions

B4. Define functions as external where appropriate [info] [resolved]

Almost all `public` functions defined in this contract (`mint`, `setMintAllowance`, `burn`, `setPause`, etc etc) are not called internally to the contract, and so should be declared as `external`
Recommendation: Define functions as external where appropriate
Severity: Info - in certain cases, declaring functions as external rather than public can save some gas for the callers, but that is not the case for the functions defined in this contract.
Resolution: Functions are now declared external where appropriate

B5. Confusing comments [info] [resolved]

There are some places in the code where the comments are inappropriate or in contradiction with the actual code.

```
1.59  // initialize, call initializer to lock the implementation instance.
```

This comment makes sense in its first occurrence, but reappears here above the `initialize` function, where it makes no sense

```
// Permit, uses super, allowed only if delegationMode is true, or if the
relayer is zz:
```

This should be `delegateMode` (but see B2), and it would be clearer to write “msg.sender” or “the sender” instead of “relayer”
Recommendation: The comments were fixed, and more documentation strings were added throughout the code

BackedTokenImplementationV2.sol

We did not find any issues in the `BackedTokenImplementationV2.sol` file.

ERC20PermitDelegateTransfer.sol

We did not find any issues in the `ERC20PermitDelegate.sol` file.