Ω

Team Omega

https://teamomega.eth.limo

Giza Token Contract Audit

Final Report

Februari 10, 2025

Giza has asked team Omega to audit their Solidity contracts. The current document is a preliminary audit report. Giza will read this and fix any issues they choose to fix, and Omega will subsequently review the fixes and write a final report.

Team Omega
Team Omega (https://teamomega.eth.limo/) specializes in Smart Contract security audits on the Ethereum ecosystem.

Giza
Giza.tech (https://www.gizatech.xyz/) is developing a platform for AI solutions in web3

Scope of the Audit
The software that was audited is being developed in the following repository and branch:

https://github.com/giza11111techxyz/giza-token/tree/audit-scope

Giza has asked us to review the following commit:

```
a4c54ffa68ab810dc0c8846400380a2772a275b1
```

- The solidity code in the `src` directory (excluding the "deprecated" directory)
- The deployment scripts
- The integration with layerzero

Methodology
The audit report has been compiled on the basis of the findings from different auditors (Jelle and Ben). The auditors work independently. Each of the auditors has several years of experience in developing smart contracts and web3 applications.

Liability

The audit is on a best-effort basis. In particular, the audit does not represent any guarantee that the software is free of bugs or other issues, nor is it an endorsement by Team Omega of any of the functionality implemented by the software.

Resolution

The issues were addressed in the following commit

```
b973b990cdf9381005297ad081d17ea3c98404d7
```

Summary

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 **4** issues as “low”, and an additional **7** issues were classified as “info” - we believe the code would improve if these issues were addressed as well.

Most of these issues were addressed.

Severity	**Number of issues**	**Number of resolved issues**
High	0	-
Medium	0	-
Low	4	3
Info	7	4

General

G1. Licensing is not completely implemented [low] [resolved]

The audited solidity files contain the following SPDX header info:

```
// SPDX-License-Identifier: BUSL-1.1
```

If your intention is to distribute the files under the BUSL license, you must follow a number of steps, as outlined in the license file itself: https://spdx.org/licenses/BUSL-1.1.html
Specifically, you must:

- claim copyright
- state a change date (when the license moves from BUSL to GPL)
- add the GPL license text

Recommendation: Fix the license as described
Severity: Low
Resolution: The issue was resolved as recommended

G2. Avoid code duplication when possible [info] [resolved]

The functions `stake` and `stakeOnBehalf` in `GizaStaking.sol` and the functions `claim` and `claimAndStake` in `AirdropDistributor.sol` are nearly identical, and contain a lot of duplicated code. Consider using helper `_stake` and `_claim` internal functions to manage the duplicated code in a single place.
Recommendation: Avoid code duplication and adhere to the DRY principle by using helper internal functions.
Severity: Info
Resolution: This issue was resolved as recommended.

Deployment

D1. Set the delegate of the OFTAdapter to the multisig [low] [resolved]

In `1_DeployGizaToken.s.sol`, the `OFTAdapter` is deployed, and both its owner and delegate variables are set to the deployer address. Subsequently, in `3_TransferOwnership.s.sol`, the ownership of the contract is transferred to the multisig. However, the value of the `delegate` variable remains unchanged, and is still set to the `deployer` address.

Note that the `delegate` role has farleading powers; according to the documentation:

```
	/// @notice delegate is authorized by the oapp to configure anything in
layerzero
```

Recommendation: Set the `delegate` address to the multisig address. Or, alternatively, set it to the 0 address on deployment - the owner can always set a new delegate in a later stage using `setDelegate`.
Severity: Low
Resolution: This issue was resolved as recommended.

D2. Use a multisig with a non-trivial threshold for ownership [low] [not resolved]

In the scripts, ownership of the contracts is set to the multisig that is deployed at address `0xa87dd67893AB39aCe5c55501f6f9C1B390F86ae0` on Ethereum mainnet and Base.

As of writing, these multisig wallets have a threshold of 1 out of 4 signers. Such a configuration is a setup that is more vulnerable than assigning ownership to a single EOA account.

Recommendation: Configure the multisig addresses to have a threshold of at least 2 signers.
Severity: Low
Resolution: This issue was acknowledged by the team, and while not resolved for now they have assured it will be handled before live deployment.

D3. Ensure no duplicate addresses in Merkle tree [info] [resolved]

There are various scripts in the project which are used to process airdrops data and generate a Merkle tree. A good security measure to avoid issues with claiming after deployment is to ensure no address appears in the data more than once (comparing addresses regardless of the checksum). This will ensure no issues when claiming as the contract is not designed to support claiming multiple entries of the same user in the same tree.
Recommendation: Check the data to ensure no address appears more than once in the data, comparing the entries in a non case sensitive way.
Severity: Info
Resolution: This issue was resolved as recommended.

AirdropDistributor.sol

AD1. Contract can be simplified [info] [not resolved]

The variable `isDistributing` is set to true when the contract is funded with `sendReward`, and must be true for claiming to succeed. It's main function seems to be to let the owner control when distribution starts.
Recommendation: Allowing the owner to set a `distributionStartTime` variable directly is more straightforward, and would give the owner more fine-grained control over when distribution starts.
Severity: Info
Resolution: This issue was acknowledged by the team, but was not resolved.

GizaStaking.sol

GS1. GizaStaking should inherit from IGizaStaking [info] [not resolved]

The `GizaStaking` contract currently does not inherit from the `IGizaStaking` interface. To do so is useful both for clarity, as well as to make sure that in future updates, the contract adheres to the given interface.
Recommendation: `GizaStaking` should inherit from `IGizaStaking`.
Severity: Info
Resolution: This issue was acknowledged by the team, but was not resolved.

GS2. Constructor should enforce the maximum cooldown [low] [resolved]

The constructor is currently allowing the deployer to pass any length for the cooldown, and does not verify the initial cooldown is less than the maximum cooldown, which could allow an initial cooldown longer than expected by users.
Recommendation: Ensure the cooldown set in the constructor is lower than the maximum cooldown.
Severity: Low
Resolution: This issue was resolved as recommended.

GizaMainChain.sol

GMC1. Use revert with custom errors instead of require [info] [resolved]

Almost all code under review uses `revert` with custom errors to handle contract reverts, except one instance in the `GizaMainChain` contract, which uses require instead. It would be better to replace the `require` in the `mint` function with the more gas efficient `revert` with custom errors.
Recommendation: Replace the `require` in the `mint` function with `revert` with custom error.
Severity: Low
Resolution: This issue was resolved as recommended.

Ownable.sol

O1. Use OpenZeppelin Ownable2Step instead of your own implementation [info] [not resolved]

The OpenZeppelin standard library offers the `Ownable2Step` contract, which matches the functionality you are looking for with your own implementation. It is better to stick to OpenZeppelin code when possible as it is extremely well reviewed and maintained.
Recommendation: Use OpenZeppelin's `Ownable2Step` instead of your own implementation.
Severity: Info
Resolution: This issue was acknowledged by the team, but was not resolved.

O2. Read from memory instead of storage when possible [info] [resolved]

In line 50, you could save some gas by passing the event the memory variable `newOwner` instead of the storage variable `pendingOwner`.
Recommendation: Use `newOwner` instead of `pendingOwner` when emitting the `OwnershipTransferStarted` event.
Severity: Info
Resolution: This issue was resolved as recommended.