Audit Report

Version 1.0

–

July 20, 2024

Dega Claim Contract Audit Report

storming0x

July 20, 2024

Prepared by: storming0x

Auditors:

- storming0x

Table of Contents

Summary

The Dega claim smart contract is designed to allow users to claim their Dega tokens which is a standard compatible ERC20 token with no added custom funcionality. The contract uses an authorized signature scheme with EIP712 implementation to allow users to claim their tokens. These signing keys are a critical point of failure and must be secured properly to ensure the tokens hold by the contract are safe from threat actors. The contract also uses access control and pausable functionality extended from OpenZepellin libraries. No critical issues were found during the audit.

NOTE: This security review only covers the DegaTokenClaim.sol contract logic. It does not cover any systems or infrastructure related to securing the signing keys or process to generate the signature which is outside the scope for this code review. It is recommended that the DEGA team use best in class security practices to secure the signing keys and the signing process.

Disclaimer

storming0x makes all effort to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the findings provided in this document. An audit is not an endorsement of the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the Solidity implementation of the contracts.

Risk Classification

		Impact		
		High	Medium	Low
	High	H	H/M	M
Likelihood	Medium	H/M	M	M/L
	Low	M	M/L	L

For details on severity matrix see the documentation for more details.

Audit Details

Scope

DegaTokenClaim.sol

Commit Hash:

https://github.com/DEGAorg/TGE-claim-contract/commit/e7a939dde0c159608973fb3a308e8da56b217542

Roles

Default Admin Role: Role that can add new admins Admin Role: Role that can pause the contract and add authorized signer keys

Findings Summary

Issues found

The following number of issues were found, categorized by their severity:

- Critical & High: 0 issue
- Medium: 1
- Low: 0
- Info: 1

Detailed Findings

High

Medium

[M-1] Authorized Signer can be swapped by ADMIN_ROLE

Description

The contract has an Authorized Signer address that will sign off chain messages to allow users to claim their tokens. The ADMIN_ROLE can add and remove authorized signers. If an unauthorized user

gains access to the ADMIN_ROLE they can remove the current authorized signer and add their own address. This would allow them to sign off chain messages and claim tokens unrestricted. They can also unpause the contract if it is paused.

Many addresses can have the ADMIN_ROLE which makes it difficult to properly secure the ADMIN_ROLE. It is recommended that the process of adding the authorized signer be restricted to the DEFAULT_ADMIN_ROLE only and ensure only one account can swap the authorized signer.

Impact Authorized signer can be swapped by any address with ADMIN_ROLE which can have potential security implications. It is considered this to have a low probability in practice, but regardless could have high impact on funds held by the contract.

Recommended mitigation Restrict the ability to add authorized signers to the DEFAULT_ADMIN_ROLE only. Ensure only one account can swap the authorized signer. Implement the AccessControlDefaultAdminRules.sol from OZ to increase the security of the DEFAULT ADMIN ROLE.

Team Response Fixed here:

https://github.com/DEGAorg/TGE-claim-contract/commit/a1035611ef62ac7940d6e0c0fe5e6feb48fa6796

Low

Informational

[I-1] Implement best in class security practices to secure the signing keys and the signing process

Description

The contract has an Authorized Signer address that will sign off chain messages to allow users to claim their tokens. This is a key security point in the contract and must be secured properly to ensure the tokens held by the contract are safe from threat actors. Although this review does not cover the process to secure the signing keys or the process to generate the signature off chain, it is recommended that the DEGA team use best in class security practices to secure the signing keys and the signing process.

Some possible practices as examples could be:

- Usage of HSMs (Hardware Security Modules) to store the signing keys without exposing them to the outside world. Many cloud providers offer this service. Additionally team users that can access this service should have 2fa enabled and be restricted to strict limited access.
- Access to the signing API should be restricted via IP and a secure authentication method to ensure the API is not exposed to the public.

- Rotation of Authorized Signer in some time interval to prevent long term exposure of the signing keys.
- Limit the amount of tokens held in the claim contract at any given point in time to avoid large losses in case of a breach.

This list is not intented to be exhaustive and the DEGA team should consult with web2 security professionals to ensure the signing keys are properly secured.