Introduction

A time-boxed security review of the **Beam** protocol was done by **pashov**, with a focus on the security aspects of the application's smart contracts implementation.

Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where I try to find as many vulnerabilities as possible. I can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

About **pashov**

Krum Pashov, or **pashov**, is an independent smart contract security researcher. Having found numerous security vulnerabilities in various protocols, he does his best to contribute to the blockchain ecosystem and its protocols by putting time and effort into security research & reviews. Check his previous work here or reach out on Twitter @pashovkrum.

About **Beam**

Beam is a protocol consisting of an ERC20 token (`BeamToken`), a `TokenBurner` contract, a `Migrator` contract and a `BeamDAO` contract(which is not used and out of scope). The token is a fork of the MeritToken. The idea is to allow users to migrate `MeritToken` tokens to `BeamToken` tokens where 1 `MeritToken` == 100 `BeamToken`.

Observations

The protocol token's minting and burning is controlled by an admin controlled role, so centralization is present in the protocol.

Privileged Roles & Actors

- `BeamToken` admin - can give roles to other addresses in the system
- `BEAM` minter - can call `BeamToken::mint` to mint an arbitrary amount of tokens to an arbitrary address
- `BEAM` burner - can call `BeamToken::burn` to burn an arbitrary amount of tokens from an arbitrary address

Severity classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

Impact - the technical, economic and reputation damage of a successful attack

Likelihood - the chance that a particular vulnerability gets discovered and exploited

Severity - the overall criticality of the risk

Security Assessment Summary

***review commit hash* - a11ee9d60f6ab8413daf8fbb222335baaab95db1**

***fixes review commit hash* - b3749ec52abd9333638ecf6b1365c5a539fe1d57**

Scope

The following smart contracts were in scope of the audit:

- `Migrator`
- `BeamToken`
- `TokenBurner`

Findings Summary

ID	Title	Severity	Status
[M-01]	Minting and burning of `BeamToken` is centralized	Medium	Acknowledged
[L-01]	Using a vulnerable version of an external library	Low	Fixed
[L-02]	Using an older Solidity compiler version	Low	Fixed

Detailed Findings

[M-01] Minting and burning of `BeamToken` is centralized

Severity

Impact: High, as token supply can be endlessly inflated and user tokens can be burned on demand

Likelihood: Low, as it requires a malicious or compromised admin/minter/burner

Description

Currently the `mint` and `burn` methods in `BeamToken` are controlled by `MINTER_ROLE` and `BURNER_ROLE` respectively. Those roles are controlled by the `DEFAULT_ADMIN_ROLE` which is given to the `BeamToken` deployer. This means that if the admin or minter or burner account is malicious or compromised it can decide to endlessly inflate the token supply or to burn any user's token balance, which would lead to a loss of funds for users.

Recommendations

Give those roles only to contracts that have a Timelock mechanism so that users have enough time to exit their `BeamToken` positions if they decide that they don't agree with a transaction of the admin/minter/burner.

[L-01] Using a vulnerable version of an external library

In `package.json` we can see that the OpenZeppelin library version used is `"@openzeppelin/contracts": "^4.3.1"`. According to OpenZeppelin's Security Advisories you can see that this version contains multiple High and Moderate severity vulnerabilities. While the code in-scope is not using the vulnerable parts of the library code at the moment, it is highly recommended to update to the latest stable version that has no breaking changes, meaning version 4.9.3.

[L-02] Using an older Solidity compiler version

Currently the protocol contracts use `pragma solidity 0.8.6;` which is a version that according to the List of Known Bugs in Solidity contains some Low severity issues. It is highly suggested to update the compiler version to a more recent one to make use of bugfixes and optimizations.