OpenZeppelin | security

Brickken v2 Token Audit



BRICKKEN

September 24, 2025

Table of Contents

Table of Contents 2

Summary 3

Scope 4

System Overview 5

Brickken v2 Token 5

Migration Contract 5

Privileged Roles and Trust Assumptions 6

Low Severity 7

L-01 Floating Pragma 7

L-02 Incomplete Docstring 7

L-03 withdrawExcessToken Forces Full Withdrawals 7

L-04 Blacklisted Addresses Cannot Be Configured Upon Deployment 8

L-05 The Migrator Blacklist Can Be Circumvented 8

Notes & Additional Information 9

N-01 Inconsistent Function Order 9

N-02 Ambiguous Calls to Parent Contract 9

N-03 Function Visibility Overly Permissive 9

Conclusion 11

Summary

Type	DeFi
Timeline	From 2025-09-18 To 2025-09-18
Languages	Solidity

Total Issues	8 (7 resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	0 (0 resolved)
Low Severity Issues	5 (4 resolved)
Notes & Additional Information	3 (3 resolved)

Scope

OpenZeppelin audited the Brickken/bkn-2.0 repository at commit 68e802a.

In scope were the following files:

```
src
├── Brickken.sol
└── Migrator.sol
```

System Overview

The audited codebase implements the new Brickken (BKN) v2 token, alongside a `Migrator` contract that simplifies the migration process for Brickken v1 token holders. The Brickken team is planning to deploy the new Brickken v2 token across various blockchains, notably Ethereum, Base, and BNB Smart Chain. The token will be deployed to a canonical address on all chains. The team is investigating adding support for cross-chain token transfers using CCIP integration and LayerZero OFT.

Brickken v2 Token

The Brickken v2 token extends the standard ERC-20 implementation with advanced functionalities for utility and governance. To ensure controlled supply management, it features a permissioned minting system where only authorized accounts can create new tokens. In addition, the token supports burning, allowing any holder to destroy their own tokens or use the `approve` mechanism to authorize other accounts to do so on their behalf.

For an improved user experience, it incorporates gasless approvals, enabling holders to sign an off-chain message to grant spending permission for a future transaction. The token also facilitates on-chain governance by using timestamp-based checkpoints on balances to grant holders voting power, allowing for direct participation in the protocol's decision-making process.

Migration Contract

The `Migrator` contract provides a migration process for upgrading BKN v1 tokens to v2. It implements a direct 1:1 token exchange and incorporates a blacklist system for regulatory compliance.

Privileged Roles and Trust Assumptions

This audit was conducted under the premise that all privileged account holders are trusted and will act benevolently.

The following privileged roles were identified in the codebase:

- **Default Admin:** The default admin role is assigned on the `Brickken` contract and acts as the admin role for all roles, which means that accounts with this role will be able to grant or revoke other roles.
- **Minter:** The minter is assigned on the `Brickken` contract and allows minting new tokens to a specified address.
- **Owner:** The owner role is assigned on the `Migrator` contract, and is privileged to add blacklist addresses as well as withdrawing excess ERC-20 tokens that were sent to the contract by accident.

In addition, the following trust assumptions were made:

- The `Brickken` v2 contract will be deployed before the `Migrator` contract.
- The `Migrator` contract will be initialized correctly and the correct supply of BKN v2 tokens will be provided at initialization.

Low Severity

L-01 Floating Pragma

The `Brickken` and `Migrator` contracts both use the `^0.8.27` floating pragma directive.

Consider using fixed pragma directives to clearly identify the Solidity version with which the contracts will be compiled.

Update: *Resolved in pull request #1 at commit c0227e1.*

L-02 Incomplete Docstring

Within `Migrator.sol`, the `Blacklisted` event's `status` parameter is not documented.

Consider thoroughly documenting all functions/events (and their parameters or return values) that are part of a contract's public API. When writing docstrings, consider following the Ethereum Natural Specification Format (NatSpec).

Update: *Resolved in pull request #2 at commit 375f9d6.*

L-03 `withdrawExcessToken` Forces Full Withdrawals

In the `Migrator` contract, the `withdrawExcessToken` function can be used by the owner to withdraw tokens from the contract.

To avoid potential issues when sweeping BKN v2 tokens mistakenly sent to the contract by users, consider adding an extra parameter to the `withdrawExcessToken` that details how many tokens the owner wants to withdraw. This avoids situations where the owner has to withdraw the entire remaining BKN v2 supply from the migrator to sweep tokens mistakenly sent to the contract, which would temporarily block migrations.

Update: *Resolved in pull request #3 at commit d4d4551.*

L-04 Blacklisted Addresses Cannot Be Configured Upon Deployment

The `Migrator` contract facilitates the migration of BKN v1 to BKN v2 tokens. The `migrate` function prevents blacklisted addresses from migrating. To blacklist an account, the owner of the contract must call the `blacklist` function. This means that addresses can only be blacklisted after deployment in a separate transaction, considering the deployment will be initialized using an EOA. This non-atomic process allows accounts that should be blacklisted to back-run the deployment transaction, or front-run the blacklist function call, and migrate their tokens before being blacklisted.

To avoid such a scenario upon deployment, consider allowing the deployer to blacklist known non-compliant accounts during deployment.

Update: *Resolved in pull request #4 at commit c5cc2c6.*

L-05 The `Migrator` Blacklist Can Be Circumvented

The `migrate` function can be called by BKN v1 holders to migrate their tokens 1:1 to the v2 token contract. This function implements a blacklist which prevents blacklisted addresses from migrating their tokens. However, this blacklist can be circumvented by transferring the tokens to another address and migrating before the new address is blacklisted. For example, a blacklisted address could do the following in a single transaction:

1. Delegate to a `multicall` contract using EIP-7702
2. Call itself to do two calls:
    1. Approve the contract `B` with its BKN v1 tokens
    2. Call the contract `B`, which in turns calls the `migrate` function to migrate its tokens

Since the above happens atomically, there is no way for the owner to blacklist the address of contract `B` before it migrates.

To improve the robustness of the blacklisting mechanism, consider switching to a whitelist mechanism with a Merkle tree instead.

Update: *Acknowledged, will resolve. The Brickken team stated:*

> *We will likely implement a merkle tree instead of a blacklist mapping. Given that this doesn't affect the token itself but only the migrator contract we will take some time to study all possibilities. One tradeoff that we want to consider is the fact that any transfer of BKN v1 after the snapshot is taken to build the tree will create the need to recompute the tree itself.*

Notes & Additional Information

N-01 Inconsistent Function Order

The `Brickken` contract deviates from the Solidity Style Guide due to having inconsistent ordering of functions.

To improve the project's overall legibility, consider standardizing ordering throughout the codebase as recommended by the Solidity Style Guide (Order of functions).

Update: *Resolved in pull request #5 at commit 58e41bd.*

N-02 Ambiguous Calls to Parent Contract

Within `Brickken.sol`, the `super._update` and `super.nonces` functions perform ambiguous calls to the parent contracts.

Consider avoiding ambiguous calls to parent contracts, explicitly specifying which parent contract's function to call.

Update: *Resolved in pull request #6 at commit 70a1036.*

N-03 Function Visibility Overly Permissive

The `mint` function in `Brickken.sol` with `public` visibility could be limited to `external`.

To better convey the intended use of functions, consider changing a function's visibility to be only as permissive as required.

Update: *Resolved in pull request #7 at commit 4e1a740.*

Conclusion

The audited contracts include a new version of the Brickken token and a contract to simplify the token migration process.

No major issues were identified, while some improvements were suggested for the blacklist feature. Overall, the codebase was found to be simple and well-written.

The Brickken team is appreciated for their responsiveness during this engagement.