Bitlayer: Security Audit Report

DogScan Security Team

July 23rd, 2025

Contents

DogScan Security Audit Report

Project	Bitlayer
Chains	Ethereum (ID: 1), BSC (ID: 56)
Ethereum Contract Address	`BTR on Ethereum`
BSC Contract Address	`BTR on BSC`
Audit Date	July 23rd, 2025
Report Version	1.0

1. Executive Summary

The `BTR` token contract has completed its security audit. This contract is a well-implemented standard `ERC20` token that uses the thoroughly tested `OpenZeppelin` library. The contract has been deployed on both Ethereum mainnet and BSC with identical code implementations. The audit process found no critical, high, or medium-risk security vulnerabilities. The contract adopts a centralized governance structure including `owner`, `minter`, and `burner` roles, but these functions are technically correct and secure in their implementation.

The overall risk level is assessed as [Low Risk]. The contract is secure and reliable at the technical level, with centralized governance features being a design choice that users should understand before participating.

2. Audit Scope

The audit scope covers the BTR token contracts deployed on multiple chains with identical code implementations:

Ethereum Mainnet

- **Contract Address**: `0x6C76dE483F1752Ac8473e2B4983A873991e70dA7`
- **Contract Type**: Standard `ERC20` token contract (non-proxy contract)

BSC (Binance Smart Chain)

- **Contract Address**: `0x6C76dE483F1752Ac8473e2B4983A873991e70dA7`

- **Contract Type**: Standard `ERC20` token contract (non-proxy contract)

The contract includes the following main components:

- **Main Contract**: `BTR.sol`
- **Inherited Libraries**: `OpenZeppelin` contracts, including `ERC20`, `ERC1363`, `Ownable2Step`
- **Dependencies**: `EnumerableSet` utility library for role management
- **Functions**: Standard `ERC20` functions, as well as minting and burning capabilities

3. Audit Methodology

This audit was conducted using a multi-agent AI security analysis framework. The process involves multiple specialized AI agents conducting comprehensive reviews of the smart contract's source code and on-chain state. The methodology includes:

1. **Automated Vulnerability Detection**: Static analysis agents scan code for known vulnerability patterns, logic flaws, and deviations from best practices.
2. **Economic Model Analysis**: DeFi-focused agents evaluate the contract's economic incentives, potential manipulation risks, and mathematical soundness.
3. **Access Control Review**: Agents analyze the role-based access control (RBAC) implementation, identifying privileged functions and potential centralization risks.
4. **Manual Heuristic Review**: A chief security strategist AI synthesizes all agent findings, categorizing alerts, eliminating false positives, and assessing systemic impact of combined vulnerabilities through cross-referencing call chains and contract logic.

4. Findings Summary

ID	Title	Severity	Status
I-01	Centralized Governance Structure	**Informational**	**User Advisory**

5. Detailed Findings

[I-01] Centralized Governance Structure (Informational)

Severity: Informational

Description The contract adopts a centralized governance model with the following design features:

1. **Owner Privileges**: The contract owner can assign and revoke minter and burner roles through `setMinter` and `setBurner` functions.
2. **Minting Function**: Addresses with the `minter` role can call the `mint` function to create new tokens.
3. **Burning Function**: Addresses with the `burner` role can call the `burn` function to destroy tokens from any address.
4. **Ownership Transfer**: The owner can transfer ownership through the `transferOwnership` function (using a two-step process for security).

Impact Under this design pattern, the contract adopts a centralized management mechanism. Users need to understand:

- The project team has the ability to mint new tokens
- The project team has the ability to burn tokens from any address
- The owner can assign and revoke special roles
- This design provides flexible token management capabilities during the project's early stages

Code Locations

```
function setMinter(address minter, bool enabled) public onlyOwner { ... }

function setBurner(address burner, bool enabled) public onlyOwner { ... }

function mint(address to, uint256 amount) public onlyMinter { ... }

function burn(address account, uint256 amount) public onlyBurner { ... }
```

User Advisory Users should understand before participating in this token:

1. This is a centrally managed token contract where the project team has special privileges
2. The existence of minting and burning functions means the token supply may change
3. Users should evaluate the team's credibility and governance transparency
4. It is recommended to monitor the project's governance mechanisms and multi-signature management

6. Systemic Risks

After audit, this contract exhibits the following systemic characteristics:

1. **High Technical Security**: The contract uses standard `OpenZeppelin` library implementation and is secure and reliable at the technical level. All core functions (transfers, approvals, etc.) strictly follow the `ERC20` standard.
2. **Centralized Governance Structure**: The contract adopts role-based access control, with the owner having the authority to assign minter and burner roles. This design provides flexibility for the project but also means centralized control exists.
3. **Standard Compatibility**: The contract is fully compatible with the `ERC20` standard and additionally implements the `ERC1363` extension, providing token transfer callback functionality.

7. Architecture and Design Observations

The contract is built on a solid foundation using `OpenZeppelin` standard, secure contract implementations. The architecture is clear and follows industry best practices:

1. **Role Management**: The contract uses an `EnumerableSet`-based role management system, allowing efficient management of minter and burner roles. This design provides flexibility while maintaining code clarity.
2. **Standard Compatibility**: The contract is fully compatible with the `ERC20` standard and additionally implements the `ERC1363` extension, providing support for post-transfer callbacks.
3. **Centralized Governance**: The contract adopts a centralized governance structure. While this may raise concerns among some users, it is technically secure and correct in implementation. This design provides necessary flexibility during the project's initial stages.

8. Conclusion

The `BTR` token contract is a well-implemented standard `ERC20` token using the thoroughly tested `OpenZeppelin` library. The contract includes a centralized governance structure with `owner`, `minter`, and `burner` roles, which are technically correct and secure in implementation. While centralized features exist, this is a design choice and is well-implemented.

Safe for deployment and use: No security risks were found at the technical level, and all functions are correctly implemented. Users should fully understand the centralized governance features before participating in this token. It is recommended that users evaluate the project team's credibility, governance transparency, and token economic model.

Overall security assessment: [Low Risk] - Technical implementation is secure and reliable, centralized features are design choices, users need to fully understand the governance mechanisms.