Bitlayer: Security Audit Report

DogScan Security Team

DogScan

July 16th, 2025

Contents

DogScan Security Audit Report

Project	Bitlayer
Chain	Avalanche (ID 43114)
Proxy Contract	ERC1967Proxy
Implementation Contract	BridgedYBTCB
Audit Date	July 16th, 2025
Report Version	1.0

1. Executive Summary

We conducted a comprehensive security audit of the `BridgedYBTCB` token contract system deployed on the Avalanche network. The system implements a bridged version of Bitcoin using an ERC1967 upgradeable proxy pattern. The audit results reveal **some issues requiring attention**, primarily involving bridged asset supply management deficiencies.

Our analysis identified bridged asset supply mismatch issues and centralized governance structures. The contract is built on standard OpenZeppelin libraries providing a solid technical foundation, but certain features in its design may pose risks to bridged asset integrity.

Overall Risk Rating: Medium

We recommend that users can interact more safely with this token contract after implementing the suggested improvement measures.

2. Audit Scope

The audit scope encompasses the complete `BridgedYBTCB` contract system deployed on Avalanche:

Proxy Contract (ERC1967):

- Address: `0x2cd3CdB3bd68Eea0d3BE81DA707bC0c8743D7335`
- Pattern: OpenZeppelin ERC1967 Proxy
- Function: Transparent proxy delegating calls to implementation contract

Implementation Contract:

- Address: `0x786248B634B1ebC7B5fc809c74a1A212fc920d63`
- Contract: `BridgedYBTCB`
- Features: UUPS Upgradeable, ERC20 with Permit, Burnable, Role-based access control

Key Audit Areas:

- ERC1967 proxy implementation and security
- UUPS upgradeability mechanism
- Role-based access controls (minter/burner roles)
- Token minting and burning functionality
- Ownership and administrative controls
- Proxy-implementation interaction security

3. Audit Methodology

This audit employed a comprehensive multi-agent AI security analysis workflow specifically designed for smart contract security assessment:

1. Specialized Analysis Modules:

- **Access Control Specialist**: Evaluated upgrade mechanisms and administrative privileges
- **Core Logic Analyst**: Reviewed token transfer and business logic implementation
- **Honeypot Detection Expert**: Examined potential sell restriction mechanisms
- **Liquidity Risk Analyst**: Assessed liquidity manipulation risks
- **Reentrancy Specialist**: Analyzed external call patterns and state management
- **Economic Model Expert**: Evaluated token supply dynamics and economic model
- **User Rights Analyst**: Assessed restrictions on user token operations
- **DeFi Mathematics Expert**: Reviewed arithmetic operations and mathematical security

2. Advanced Research Team:

- Advanced attack vector pattern recognition
- Cross-contract interaction analysis
- Systemic risk assessment

3. Strategic Synthesis:

- Chief Security Strategist integration of all findings
- Risk prioritization and impact assessment
- Comprehensive reporting and recommendation development

4. Findings Summary

ID	Title	Severity	Status
M-01	Bridged asset supply management deficiency	**Medium**	**Recommended Fix**
I-01	Multisig governance for upgrade authority management	**Informational**	**Recommended Attention**

5. Detailed Findings

[M-01] Bridged asset supply management deficiency (Medium)

Severity: Medium

Description The contract inherits from ERC20BurnableUpgradeable, allowing any token holder to destroy their own tokens through the standard burn(uint256) function. For bridged assets, this breaks the 1:1 backing relationship with locked assets on the origin chain.

Impact

- Mismatch between bridged token supply and locked assets
- Potential for orphaned assets on the origin chain
- Accounting discrepancies in bridge infrastructure
- Loss of backing guarantee for remaining tokens

Technical Details The contract contains multiple functions that can reduce token total supply, all of which break the 1:1 backing relationship for bridged assets:

1. **Inherited burn function**: `burn(uint256 amount)` - Any user can destroy their own tokens
2. **Inherited burnFrom function**: `burnFrom(address account, uint256 amount)` - Requires authorization to destroy others' tokens

3. **Custom burn function**: `burn(address account, uint256 amount)` - Only burner role can call, no authorization required

Important Distinction: The custom `burn(address, uint256)` has similar functionality to the inherited `burnFrom(address, uint256)` but completely different permission models:

- `burnFrom`: Requires prior authorization (allowance) from the token holder
- Custom `burn`: Only requires burner role, can forcibly destroy tokens from any account

Code Evidence

```
contract BridgedYBTCB is UUPSUpgradeable, Ownable2StepUpgradeable,
    ERC20PermitUpgradeable, ERC20BurnableUpgradeable {

    // Inherited functions - user self-destruction
    // function burn(uint256 amount) public virtual
    // function burnFrom(address account, uint256 amount) public virtual

    // Custom function - forced destruction (no authorization required)
    function burn(address account, uint256 amount) public onlyBurner {
        _burn(account, amount);
    }
}
```

Recommendations

- **Override inherited burn functions to disable user-initiated destruction**:
  - Override `burn(uint256)` function to require only authorized bridge operators can call
  - Override `burnFrom(address, uint256)` function or restrict its use in bridging scenarios
- **Rename custom burn function to avoid confusion**:
  - Rename `burn(address, uint256)` to `bridgeBurn` or `forceBurn`
  - Clearly distinguish between user-initiated destruction and bridge operator forced destruction
- **Implement bridge-controlled burning only through authorized bridge operators**
- **Add supply reconciliation mechanisms to maintain asset backing**
- **Monitor total supply changes and implement automated bridge notifications**

[I-01] Multisig governance for upgrade authority management (Informational)

Severity: Informational

Description The contract system is controlled by a multisig wallet (Gnosis Safe) for upgrade authority, which is a relatively secure governance model. The current owner address `0x2728df4D22253004C017675bd609962cD641D797` is a multisig wallet that requires consensus from multiple signers to execute upgrades, significantly reducing single-point-of-failure risks.

Impact Multisig governance provides the following security benefits:

- Prevents malicious operations or key compromise risks from a single account
- Requires consensus from multiple signers to execute critical operations
- Provides better decentralization compared to single EOA control
- Reduces the possibility of immediate malicious upgrade execution

However, considerations remain:

- Transparency of multisig wallet signer composition and threshold settings
- Lack of timelock mechanisms, upgrades can still be executed relatively quickly

Technical Details The contract uses UUPS (Universal Upgradeable Proxy Standard) with multisig wallet-controlled upgrade authorization. Although the _authorizeUpgrade function only requires owner permission, since the owner is a multisig wallet, it actually requires consensus from multiple signers.

Code Evidence

```
function _authorizeUpgrade(address newImplementation) internal override onlyOwner
    {}

function setBurner(address burner, bool enabled) public onlyOwner {
    if (enabled) {
        burners.add(burner);
    } else {
        burners.remove(burner);
    }
    emit BurnerSet(burner, enabled);
}
```

```
11
12  function burn(address account, uint256 amount) public onlyBurner {
13      _burn(account, amount);
14  }
```

Recommendations

- **Publicly disclose multisig wallet signer information and threshold settings for transparency**
- **Consider adding timelock mechanisms to provide users with reaction time for proposed changes**
- **Consider migrating to decentralized governance systems (DAO) in the long term**
- **Regularly audit multisig wallet signer composition**
- **Establish pause mechanisms and procedures for emergency situations**

6. Architecture and Design Assessment

Design Strengths

1. **Use of Standard Libraries**: Contract built on battle-tested OpenZeppelin contracts
2. **Proper Initialization Protection**: Current implementation correctly uses `initializer` modifier
3. **Clear Role Separation**: Minter and burner roles are clearly separated
4. **ERC20 Compliance**: Basic token functionality conforms to standards
5. **UUPS Upgrade Pattern**: Uses standard upgradeable proxy pattern

Key Architectural Issues

1. **Bridged Asset Integrity**: Multiple burn functions (burn, burnFrom, custom burn) may break 1:1 backing with origin chain assets
2. **Function Overloading Confusion**: Custom `burn(address, uint256)` has similar functionality to inherited `burnFrom(address, uint256)` but different permission models, potentially causing developer integration confusion
3. **Governance Transparency**: Currently controlled by multisig wallet, recommend improving transparency of signer composition and threshold settings

Recommended Improvements

1. **Bridge-Specific Design**: Override all inherited burn functions to maintain bridging integrity
2. **Function Naming Optimization**: Rename custom burn function to avoid confusion with standard ERC20 functions
3. **Supply Monitoring Mechanisms**: Implement automated monitoring and bridge notification systems
4. **Governance Transparency Optimization**: Publicly disclose multisig wallet signer information and threshold settings, consider adding timelock mechanisms

7. Conclusion

This security audit identified **key security issues** in the `BridgedYBTCB` token contract system. The contract is built on standard OpenZeppelin libraries providing a solid technical foundation, but there are some design and implementation issues that require attention.

Key Findings Summary:

- **Bridged Asset Management**: Multiple burn functions (including inherited burn, burnFrom, and custom burn) may break 1:1 backing relationship with origin chain assets
- **Function Design Issues**: Function overloading may cause developer integration confusion, especially with same-name functions having different permission models
- **Governance Structure**: Current multisig wallet governance provides good security assurance, recommend further improving transparency

Overall Risk Rating: Medium

No obvious malicious code was found in the current implementation, and the contract follows industry standard practices. Main concerns focus on the special nature of bridged assets and long-term governance considerations.

Recommended Improvement Measures:

1. Override all inherited burn functions (burn, burnFrom) to maintain bridged asset integrity
2. Rename custom burn function to avoid confusion with standard functions
3. Implement supply reconciliation mechanisms to ensure asset backing
4. Improve multisig wallet governance transparency by publicly disclosing signer information and threshold settings
5. Monitor total supply changes and implement bridge notifications

We recommend that this token contract can provide safer bridging services for users after implementing the suggested improvement measures.

8. Disclaimer

This audit report is for reference only and does not constitute investment advice. The analysis is based on smart contract source code provided at a specific point in time and is not an endorsement of the project or a guarantee of its security. Smart contracts have inherent risks, including the possibility of undiscovered vulnerabilities. Users should conduct their own research and exercise extreme caution when interacting with any smart contract. The findings in this report highlight severe security concerns that pose significant risks to user funds.