Shape Token Contract

Security Assessment

May 2, 2025

Prepared for:
Shape Factory, Inc.

Prepared by: **Quan Nguyen**

Table of Contents

Project Summary

Contact Information

The following project manager was associated with this project:

Emily Doucette, Project Manager
emily.doucette@trailofbits.com

The following engineering director was associated with this project:

Benjamin Samuels, Engineering Director, Blockchain
benjamin.samuels@trailofbits.com

The following consultant was associated with this project:

Quan Nguyen, Consultant
quan.nguyen@trailofbits.com

Project Timeline

The significant events and milestones of the project are listed below.

Date	Event
April 11, 2025	Project kickoff call
April 22, 2025	Delivery of report draft
April 22, 2025	Report readout meeting
May 1, 2025	Completion of fix review (appendix D)
May 2, 2025	Delivery of final comprehensive report

Executive Summary

Engagement Overview

Shape Factory, Inc. engaged Trail of Bits to review the security of the Shape token contract and associated contracts. The $SHAPE token implements ERC-7802 to enable asset interoperability across the Optimism Superchain. It uses a bridge mechanism that has permission to mint and burn tokens during cross-chain transfers, allowing tokens to be fungible across the entire Superchain ecosystem. The $SHAPE token also functions as a governance token with voting capabilities and employs an immutable ownership model featuring a designated leader chain.

One consultant conducted the review from April 14 to April 18, 2025, for a total of one engineer-week of effort. Our testing efforts focused on the $SHAPE token implementation, immutable cross-chain ownership, and associated governance voting power mechanism. With full access to source code and documentation, we performed static and dynamic testing of the codebase, using automated tools and manual analysis techniques.

Observations and Impact

The $SHAPE token contract demonstrates solid engineering practices with clear separation of concerns and comprehensive testing. During our review, we did not identify any security issues that would directly lead to loss of funds for users.

The most severe finding, with a severity of medium, relates to the governance functionality (TOB-SHAPE-1). The token supply is fragmented across different L2 and L1 chains due to the token's multi-chain nature. However, voting power is only accounted for based on the current token supply on the leader chain. This limitation could result in underrepresentation of token holders on non-leader chains in governance decisions, potentially undermining the democratic principles of the governance system.

We also identified a security concern regarding cross-chain message validation. The absence of chain ID verification in the L2-to-L1 message flow could allow sophisticated attackers to bypass ownership verification (TOB-SHAPE-2). This could be exploited by deploying malicious code to the messenger's address on an alternative L2 chain, compromising the cross-chain ownership model's integrity.

Recommendations

Based on the codebase maturity evaluation and findings identified during the security review, Trail of Bits recommends that Shape Factory, Inc. take the following steps:

- **Remediate the findings disclosed in this report.** Specifically, implement proper chain ID validation in the cross-chain message flows to prevent potential ownership verification bypasses.

- **Reconsider the governance token design** to account for voting power across all chains where the token exists, not just the leader chain. This could involve implementing a cross-chain voting system aggregating voting power from all chains to ensure fair representation.

Finding Severities and Categories

The following tables provide the number of findings by severity and category.

EXPOSURE ANALYSIS

Severity	*Count*
High	0
Medium	1
Low	0
Informational	1
Undetermined	0

CATEGORY BREAKDOWN

Category	*Count*
Data Validation	1
Undefined Behavior	1

Project Goals

The engagement was scoped to provide a security assessment of the $SHAPE token contract. Specifically, we sought to answer the following non-exhaustive list of questions:

- Does the cross-chain token bridge mechanism correctly mint and burn tokens without allowing duplicate minting or unauthorized supply changes?
- Can an attacker manipulate the immutable cross-chain ownership model to gain unauthorized control over token functions on non-leader chains?
- Do the smart contracts handle edge cases appropriately when tokens are bridged between chains?
- Are there appropriate safeguards to prevent governance attacks through token-bridging operations?
- Is the $SHAPE token implementation compliant with the ERC-7802 specification?
- Do the smart contracts properly implement the intended design features?
- Does the codebase conform to industry best practices?

Project Targets

The engagement involved reviewing and testing the following target.

shape-token

Repository	github.com/shape-network/shape-token
Version	`0fb5da0c3d98fc831487a7ca20a5999f988665e0`
Type	Solidity
Platform	EVM

Project Coverage

This section provides an overview of the analysis coverage of the review, as determined by our high-level engagement goals. Our approaches included the following:

- **$SHAPE token implementation:** We assessed the $SHAPE token contract, which serves as the core token contract implementing ERC-7802 to enable asset interoperability across the Superchain. Our review focused on validating the token's compliance with the standard and its ability to maintain consistent state across multiple chains. We examined the minting and burning mechanisms triggered during cross-chain transfers to ensure proper supply management and token fungibility.
- **Immutable cross-chain ownership model:** We reviewed the implementation of the leader chain ownership model, where the owner on the designated leader chain can send messages to other chains to perform ownership actions. We assessed the security of this mechanism, focusing on authorization checks, potential centralization risks, and the integrity of cross-chain administrative commands.
- **Governance functionality:** We examined the governance mechanisms of the $SHAPE token, including the voting power calculation. We specifically analyzed how the system accounts for token balances on different chains when determining voting power; through this review, we found that only tokens on the leader chain contribute to governance influence.

Coverage Limitations

Because of the time-boxed nature of testing work, it is common to encounter coverage limitations. During this project, we were unable to perform comprehensive testing of the following system elements, which may warrant further review:

- While we evaluated the core cross-chain functionality of the $SHAPE token, we did not perform extensive testing of its interaction with various types of AMMs and DeFi protocols across different chains. These integrations may present unique challenges that require specific review.

Codebase Maturity Evaluation

Trail of Bits uses a traffic-light protocol to provide each client with a clear understanding of the areas in which its codebase is mature, immature, or underdeveloped. Deficiencies identified here often stem from root causes within the software development life cycle that should be addressed through standardization measures (e.g., the use of common libraries, functions, or frameworks) or training and awareness programs.

Category	Summary	Result
Arithmetic	The contracts do not include any arithmetic operations.	**Not Applicable**
Auditing	The contracts emit appropriate events for cross-chain transfers and minting, burning, and ownership-related actions. These events provide sufficient visibility for off-chain monitoring systems to track token movements across the Superchain ecosystem.	**Satisfactory**
Authentication / Access Controls	The immutable ownership model implements robust authentication for cross-chain administrative actions. Access controls correctly limit sensitive operations to authorized roles, with proper validation of cross-chain messages.	**Satisfactory**
Complexity Management	The contracts use the well-established OpenZeppelin library, specifically its `ERC20BurnableUpgradeable`, `ERC20PermitUpgradeable`, and `ERC20VotesUpgradeable` contracts, to implement an upgradeable governance ERC-20 token.	**Satisfactory**
Decentralization	The immutable ownership model with a designated leader chain introduces centralization concerns. The token contract is upgradeable, which will allow the proxy admin to update the logic at any time. The token contract itself has a token owner role that controls the minting of new tokens on arbitrary chains. Additionally, the governance mechanism counts voting power only on the leader chain, further centralizing decision-making authority.	**Moderate**
Documentation	The contracts are thoroughly documented with NatSpec	**Moderate**

	comments. The repository also has higher-level documentation describing the system; however, it lacks details about which address will control the token owner role and proxy admin owner roles. This information is critical for understanding the trust model and centralization risks of the system.	
Low-Level Manipulation	The contracts do not perform any low-level manipulation.	**Not Applicable**
Testing and Verification	The project has a thorough test suite that includes a variety of unit, integration, and fuzz test scenarios.	**Satisfactory**
Transaction Ordering	We did not identify any transaction ordering vectors that could maliciously affect user balances or the system as a whole.	**Satisfactory**

Summary of Findings

The table below summarizes the findings of the review, including details on type and severity.

ID	Title	Type	Severity
1	Fragmented voting power across chains	Undefined Behavior	**Medium**
2	Missing chain ID verification in L2 leader chain to L1 message flow	Data Validation	**Informational**

Detailed Findings

1. Fragmented voting power across chains	
Severity: **Medium**	Difficulty: **Low**
Type: Undefined Behavior	Finding ID: TOB-SHAPE-1
Target: `InteroperableGovernanceToken.sol`	

Description
The governance mechanism of the $SHAPE token fails to account for tokens bridged to follower chains, resulting in inaccurate voting power distribution and potential governance manipulation.

The Superchain ERC-20 implementation allows $SHAPE tokens to be transferred across multiple chains in the Superchain ecosystem while maintaining their financial fungibility. When tokens are bridged from the leader chain to follower chains, the bridge mechanism mints equivalent tokens on the destination chain while burning them on the source chain.

However, the current implementation only considers token balances on the leader chain when calculating voting power for governance decisions. This creates a disconnect between the total token supply and actual governance representation, as tokens on follower chains maintain their financial utility but lose their governance rights. Consequently, governance decisions can be made by a minority of token holders who keep their tokens on the leader chain, undermining the democratic principles of the governance system.

Exploit Scenario
A malicious actor could monitor the distribution of $SHAPE tokens across chains and strategically time a governance proposal when a significant portion of tokens have been bridged to follower chains. For example, if 60% of the total token supply is bridged to other networks for DeFi activities, the attacker could pass a harmful proposal with only 21% of the total token supply (becoming a majority of the 40% remaining on the leader chain). This allows for governance capture with a minority position of the total token supply.

Recommendations
Short term, add clear warnings in the UI and documentation about the governance implications of bridging tokens away from the leader chain.

Long term, redesign the governance mechanism to account for token balances across all chains in the Superchain ecosystem.

2. Missing chain ID verification in L2 leader chain to L1 message flow

Severity: **Informational**	Difficulty: **High**
Type: Data Validation	Finding ID: TOB-SHAPE-2
Target: `ImmutableCrossChainOwnable.sol`	

Description

The `_checkOwner` function in the `ImmutableCrossChainOwnable` contract lacks chain ID validation when processing cross-chain messages from an L2 leader chain to the L1 chain, creating a potential vulnerability in the ownership verification system.

The function implements different validation logic based on the sender. Case 4 specifically handles L2-to-L1 cross-chain messages when the leader chain is an L2, verifying that the sender is the `L1CrossChainMessengerForLeaderChain`. This messenger address is set during initialization, and the implementation assumes the current execution is occurring on the L1 chain.

```
// Case 4: L2 to L1 cross-chain message from owner when the leader chain is L2
if (sender == $._l1CrossDomainMessengerForLeaderChain) {
    address l2Sender = ICrossDomainMessenger(sender).xDomainMessageSender();
    bool fromOwner = l2Sender == $._owner;
    bool fromLeaderChain = true; // True given that the sender is the L1 cross domain
messenger for the leader chain

    if (fromOwner && fromLeaderChain) {
        return;
    }

    // Unauthorized L2-L1 cross-chain message
    revert ImmutableCrossChainOwnableUnauthorizedAccount(l2Sender, $._leaderChainId);
}
```

Figure 2.1: Excerpt of the `_checkOwner` function in `ImmutableCrossChainOwnable.sol`

However, the function does not verify that the value of `block.chainId` is equal to the value of `_l1ChainId`, which creates a security gap. While the `L1CrossChainMessengerForLeaderChain` is a trusted contract with specific behavior on the L1 chain, there is no guarantee that the same address on another L2 chain contains identical code. An attacker could deploy malicious code at the messenger's address on a different L2 chain and exploit this oversight to bypass ownership verification.

Recommendations

Short term, add an explicit check in case 4 of the `_checkOwner` function to verify that `block.chainId` is equal to `_l1ChainId` when validating L2-to-L1 messages

Long term, implement comprehensive test coverage for chain ID validation across all cross-chain communication paths.

A. Vulnerability Categories

The following tables describe the vulnerability categories, severity, and difficulty levels used in this document.

Vulnerability Categories	
Category	**Description**
Access Controls	Insufficient authorization or assessment of rights
Auditing and Logging	Insufficient auditing of actions or logging of problems
Authentication	Improper identification of users
Configuration	Misconfigured servers, devices, or software components
Cryptography	A breach of system confidentiality or integrity
Data Exposure	Exposure of sensitive information
Data Validation	Improper reliance on the structure or values of data
Denial of Service	A system failure with an availability impact
Error Reporting	Insecure or insufficient reporting of error conditions
Patching	Use of an outdated software package or library
Session Management	Improper identification of authenticated users
Testing	Insufficient test methodology or test coverage
Timing	Race conditions or other order-of-operations flaws
Undefined Behavior	Undefined behavior triggered within the system

Severity Levels	
Severity	**Description**
Informational	The issue does not pose an immediate risk but is relevant to security best practices.
Undetermined	The extent of the risk was not determined during this engagement.
Low	The risk is small or is not one the client has indicated is important.
Medium	User information is at risk; exploitation could pose reputational, legal, or moderate financial risks.
High	The flaw could affect numerous users and have serious reputational, legal, or financial implications.

Difficulty Levels	
Difficulty	**Description**
Undetermined	The difficulty of exploitation was not determined during this engagement.
Low	The flaw is well known; public tools for its exploitation exist or can be scripted.
Medium	An attacker must write an exploit or will need in-depth knowledge of the system.
High	An attacker must have privileged access to the system, may need to know complex technical details, or must discover other weaknesses to exploit this issue.

B. Code Maturity Categories

The following tables describe the code maturity categories and rating criteria used in this document.

Code Maturity Categories	
Category	**Description**
Arithmetic	The proper use of mathematical operations and semantics
Auditing	The use of event auditing and logging to support monitoring
Authentication / Access Controls	The use of robust access controls to handle identification and authorization and to ensure safe interactions with the system
Complexity Management	The presence of clear structures designed to manage system complexity, including the separation of system logic into clearly defined functions
Cryptography and Key Management	The safe use of cryptographic primitives and functions, along with the presence of robust mechanisms for key generation and distribution
Decentralization	The presence of a decentralized governance structure for mitigating insider threats and managing risks posed by contract upgrades
Documentation	The presence of comprehensive and readable codebase documentation
Low-Level Manipulation	The justified use of inline assembly and low-level calls
Testing and Verification	The presence of robust testing procedures (e.g., unit tests, integration tests, and verification methods) and sufficient test coverage
Transaction Ordering	The system's resistance to transaction-ordering attacks

Rating Criteria	
Rating	**Description**
Strong	No issues were found, and the system exceeded industry standards.
Satisfactory	Minor issues were found, but the system is compliant with best practices.
Moderate	Some issues that may affect system safety were found.
Weak	Many issues that affect system safety were found.
Missing	A required component is missing, significantly affecting system safety.
Not Applicable	The category does not apply to this review.
Not Considered	The category was not considered in this review.
Further Investigation Required	Further investigation is required to reach a meaningful conclusion.

C. Code Quality Recommendations

The following recommendations are not associated with specific vulnerabilities. However, implementing them can enhance the code's readability and may prevent the introduction of vulnerabilities in the future.

ImmutableCrossChainOwnable

1. **Using `crossDomainMessageContext` could save gas.** The `_checkOwner` function uses two separate external calls to retrieve the source and sender values when processing L2-to-L2 cross-chain messages from the owner on the leader chain. However, `L2_TO_L2_MESSENGER` provides a `crossDomainMessageContext` function that can retrieve both values in a single call, which would eliminate one external call and reduce gas consumption.

D. Fix Review Results

When undertaking a fix review, Trail of Bits reviews the fixes implemented for issues identified in the original report. This work involves a review of specific areas of the source code and system configuration, not comprehensive analysis of the system.

On May 1, 2025, Trail of Bits reviewed the fixes and mitigations implemented by the Shape team for the issues identified in this report. We reviewed each fix to determine its effectiveness in resolving the associated issue.

In summary, the Shape team has resolved one issue and has not resolved the remaining issue. For additional information, please see the Detailed Fix Review Results below.

ID	Title	Severity	Status
1	Fragmented voting power across chains	Medium	Unresolved
2	Missing chain ID verification in L2 leader chain to L1 message flow	Informational	Resolved

Detailed Fix Review Results

TOB-SHAPE-1: Fragmented voting power across chains
Unresolved. The team acknowledged the issue and decided not to resolve it immediately, providing the following explanation:

> *We don't intend to enable the token on other chains until we've done an upgrade to enable cross-chain governance.*

TOB-SHAPE-2: Missing chain ID verification in L2 leader chain to L1 message flow
Resolved in commit `5bf111ec0b`. The team implemented an explicit check in case 4 of the `_checkOwner` function to verify that `block.chainId` is equal to `_l1ChainId`.

E. Fix Review Status Categories

The following table describes the statuses used to indicate whether an issue has been sufficiently addressed.

Fix Status	
Status	**Description**
Undetermined	The status of the issue was not determined during this engagement.
Unresolved	The issue persists and has not been resolved.
Partially Resolved	The issue persists but has been partially resolved.
Resolved	The issue has been sufficiently resolved.

About Trail of Bits

Founded in 2012 and headquartered in New York, Trail of Bits provides technical security assessment and advisory services to some of the world's most targeted organizations. We combine high-end security research with a real-world attacker mentality to reduce risk and fortify code. With 100+ employees around the globe, we've helped secure critical software elements that support billions of end users, including Kubernetes and the Linux kernel.

We maintain an exhaustive list of publications at https://github.com/trailofbits/publications, with links to papers, presentations, public audit reports, and podcast appearances.

In recent years, Trail of Bits consultants have showcased cutting-edge research through presentations at CanSecWest, HCSS, Devcon, Empire Hacking, GrrCon, LangSec, NorthSec, the O'Reilly Security Conference, PyCon, REcon, Security BSides, and SummerCon.

We specialize in software testing and code review projects, supporting client organizations in the technology, defense, and finance industries and government entities. Notable clients include HashiCorp, Google, Microsoft, Western Digital, and Zoom.

Trail of Bits also operates a center of excellence with regard to blockchain security. Notable projects include audits of Algorand, Bitcoin SV, Chainlink, Compound, Ethereum 2.0, MakerDAO, Matic, Uniswap, Web3, and Zcash.

To keep up to date with our latest news and announcements, please follow @trailofbits on X and explore our public repositories at https://github.com/trailofbits. To engage us directly, visit our "Contact" page at https://www.trailofbits.com/contact or email us at info@trailofbits.com.

Trail of Bits, Inc.
228 Park Ave S #80688
New York, NY 10003
https://www.trailofbits.com
info@trailofbits.com

Notices and Remarks

Copyright and Distribution

Test Coverage Disclaimer

Trail of Bits performed all activities associated with this project in accordance with a statement of work and an agreed-upon project plan.

Security assessment projects are time-boxed and often rely on information provided by a client, its affiliates, or its partners. As a result, the findings documented in this report should not be considered a comprehensive list of security issues, flaws, or defects in the target system or codebase.

Trail of Bits uses automated testing techniques to rapidly test software controls and security properties. These techniques augment our manual security review work, but each has its limitations. For example, a tool may not generate a random edge case that violates a property or may not fully complete its analysis during the allotted time. A project's time and resource constraints also limit their use.