

Shape Buyback Contract

Security assessment by HashEye · prepared for Shape Factory

HASHEYE AUDITED

PROJECT	Shape Buyback Contract
PERFORMED BY	HashEye Security Review Team - Sergey, Brian
RESEARCH	Sergey, Brian
CLIENT	Shape Factory
CATEGORY	Blockchain
PUBLISHED	September 1, 2025
REPORT ID	research-shape-buyback-contract-2025-09-01-1gudb7

This report was produced by HashEye's security review team under the HashEye review process, combining source review, deployment-state checks, and senior manual verification. Full report detail is available at hashey.io/audits/research-shape-buyback-contract-2025-09-01-1gudb7.

Shape Buyback Contract Security Assessment (Summary Report)

September 5, 2025

Prepared for: Shape Factory, Inc.

Prepared by: Quan Nguyen

HashEye

PUBLIC

Table of Contents Table of Contents 1 Project Summary 2 Project Targets 3 Executive Summary 4 Summary of Findings 5 Detailed Findings 6 1. Incorrect fee logic in getCurrentFee function 6 2. Missing zero address validation 8 A. Vulnerability Categories 9 B. Code Quality Recommendations 11 C. Fix Review Results 12 Detailed Fix Review Results 12 D. Fix Review Status Categories 13 About HashEye 14 Notices and Remarks 15

HashEye 1 Shape Buyback Contract

PUBLIC Security Assessment

Project Summary Contact Information The following project manager was associated with this project: Emily Doucette, Project Manager emily.doucette@hasheye.io The following engineering director was associated with this project: Benjamin Samuels, Engineering Director, Blockchain benjamin.samuels@hasheye.io The following consultant was associated with this project: Quan Nguyen, Consultant quan.nguyen@hasheye.io Project Timeline The significant events and milestones of the project are listed below.

Date	Event
August 20, 2025	Delivery of report draft
August 22, 2025	Completion of fix review (appendix C)
September 5, 2025	Delivery of final summary report

HashEye 2 Shape Buyback Contract

PUBLIC Security Assessment

Project Targets The engagement involved reviewing and testing the following target. shape-buyback Repository github.com/shape-network/shape-buyback Version `e90ce7f771ec910532b9d7b7359c3a17765eb20a` Type Solidity Platform EVM

HashEye 3 Shape Buyback Contract

PUBLIC Security Assessment

Executive Summary Engagement Overview Shape Factory, Inc. engaged HashEye to review the security of the ShapeBuyback contract, a system that allows Shape users to claim ETH in exchange for \$SHAPE tokens through a time-based epoch streaming mechanism. The contract implements an upgradeable proxy pattern with OpenZeppelin's upgradeable contracts and features a time-based streaming system where ETH becomes claimable at regular intervals in exchange for \$SHAPE token fees according to configurable streaming parameters. One consultant conducted the review from August 18 to August 19, 2025, for a total of two engineer-days of effort. With full access to source code, documentation, and test suites, we performed static and dynamic analysis of the codebase using automated and manual processes. Through our review, we sought to uncover timing manipulation vulnerabilities, epoch calculation errors, fee bypass mechanisms, reentrancy risks, and any attack vectors that could allow rapid or unauthorized ETH extraction. We specifically examined the `getClaimableEther` function's calculation logic, the relationship between `lastClaimed` timestamps and epoch progression, fee enforcement mechanisms, and access controls around critical parameters that govern fund release rates. Observations and Impact We identified one low-severity issue and one informational-severity issue during the review: incorrect fee logic in the `getCurrentFee` function that could mislead users about actual fees (TOB-SHAPE-BUYBACK-1) and missing zero address

validation for the buyback recipient (TOB-SHAPE-BUYBACK-2), which could result in permanent \$SHAPE token loss during the redemption process. Our analysis found the core epoch system to be sound with proper timestamp-based controls preventing rapid fund drainage; the `getClaimableEther` function correctly limits withdrawals based on elapsed epochs and available contract balance, and the streaming parameters are properly protected by owner-only access controls. The redemption mechanism appears to be robust against the primary fund drainage concerns raised by the client. Recommendations Based on the findings identified during the security review, HashEye recommends that Shape Factory, Inc. take the following steps:

- Remediate the findings disclosed in this report.
- Enhance code documentation around expected behaviors.

HashEye 4 Shape Buyback Contract

PUBLIC Security Assessment

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

ID	Title	Type	Severity
1	Incorrect fee logic in <code>getCurrentFee</code> function	Undefined Behavior	Low
2	Missing zero address validation	Configuration	Informational

HashEye 5 Shape Buyback Contract

PUBLIC Security Assessment

Detailed Findings 1. Incorrect fee logic in `getCurrentFee` function Severity: Low Difficulty: Low Type: Undefined Behavior Finding ID: TOB-SHAPE-BUYBACK-1 Target: `src/ShapeBuyback.sol`

Description The `getCurrentFee` function returns incorrect fee values due to flawed conditional logic. When no fee change is scheduled (`nextFeeTimestamp == 0`), the function incorrectly returns `nextFee` (which is 0) instead of the current fee. The `getCurrentFee` function is intended to provide users with a view of the current fee they would be charged when calling the `claim` function. It is a view function that should mirror the behavior of the `_checkFee` function, but it currently uses opposite conditional logic. The `_checkFee` function applies `nextFee` as the current fee when `block.timestamp ≥ nextFeeTimestamp` and always returns the current fee. However, `getCurrentFee` uses completely different logic that produces inconsistent results, causing users to see incorrect fee information that does not reflect what they will actually be charged during claims.

```
function _checkFee() external view returns (uint256) { return (nextFeeTimestamp > 0 && block.timestamp < nextFeeTimestamp) ? fee : nextFee; }
```

// Code omitted for brevity

```
function _checkFee() internal returns (uint256) { if (nextFeeTimestamp > 0 && block.timestamp ≥ nextFeeTimestamp) { fee = nextFee; nextFee = 0; nextFeeTimestamp = 0; } return fee; }
```

Figure 1.1: The `getCurrentFee` and `_checkFee` functions in the `ShapeBuyback` contract

HashEye 6 Shape Buyback Contract

PUBLIC Security Assessment

Exploit Scenario Alice calls `getCurrentFee` when no fee change is scheduled (`nextFeeTimestamp == 0`). The function incorrectly returns `nextFee` (which is 0) instead of the actual fee (e.g., 1,000 tokens). Alice believes claiming is free and does not approve any tokens. When Alice calls `claim()`, the transaction reverts due to insufficient allowance since the actual fee is 1,000 tokens.

Recommendations Short term, fix the conditional logic to match the `_checkFee` function. Long term, add unit tests that verify that view functions return the same values as their corresponding state-changing functions under all conditions.

HashEye 7 Shape Buyback Contract

PUBLIC Security Assessment

2. Missing zero address validation Severity: Informational Difficulty: High Type: Configuration Finding ID: TOB-SHAPE-BUYBACK-2 Target: `src/ShapeBuyback.sol`

Description The `ShapeBuyback` contract lacks zero address validation for the `_buybackRecipient` parameter in the `initialize` function and the `setBuybackRecipient` function, which is intended to set the address that receives the \$SHAPE tokens during claims. While other critical addresses like

_owner are validated against the zero address in the initialize function, the _buybackRecipient parameter is not checked. Similarly, the setBuybackRecipient function allows the recipient to be set to the zero address without validation. Setting the buyback recipient to the zero address would cause all \$SHAPE tokens to be permanently lost when users call the claim function, as tokens would be transferred to the zero address. function setBuybackRecipient(address _buybackRecipient) external onlyOwner { buybackRecipient = _buybackRecipient; emit BuybackRecipientSet(_buybackRecipient); } Figure 2.1: The setBuybackRecipient function in the ShapeBuyback contract

Recommendations Short term, add zero address validation for _buybackRecipient to both the initialize and setBuybackRecipient functions. Long term, implement comprehensive input validation for all setter functions and consider using a checklist for critical parameter validation during code reviews.

HashEye 8 Shape Buyback Contract

PUBLIC Security Assessment

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

Vulnerability 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

HashEye 9 Shape Buyback Contract

PUBLIC Security Assessment

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.

HashEye 10 Shape Buyback Contract

PUBLIC Security Assessment

B. 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.

- Address division by zero risk. The getClaimableEther function performs division by the streamingEpoch value without checking if it is zero. If streamingEpoch is set to zero through the setStreamingEpoch function, this will cause a division by zero error and revert all claim attempts. Consider adding validation in the setStreamingEpoch and initialize functions to prevent the epoch from being set to zero.
- Remove unnecessary payable modifier. The resetLastClaimed function (line 146) is marked as payable without any apparent reason for accepting ETH. This could confuse users and waste gas if they accidentally send ETH with a transaction. Consider removing the payable modifier unless there is a specific business requirement for having it.
- Implement gas optimization in event emission. The NextFeeSet event (line 139) uses the state variable nextFeeTimestamp instead of the parameter _nextFeeTimestamp. Using the parameter directly would save gas by avoiding an unnecessary storage read operation.
- Ensure easier storage gap maintainability. The storage gap (line 50) is declared

as uint256[1_000] private __gap, which hard codes the gap size. To make future maintenance easier, consider calculating the gap size as uint256[1000 - existing_slots] private __gap, where existing_slots represents the number of state variables already declared. This approach would make it easier to maintain the total storage layout when adding new state variables in future upgrades.

HashEye 11 Shape Buyback Contract

PUBLIC Security Assessment

C. Fix Review Results When undertaking a fix review, HashEye 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 August 22, 2025, HashEye 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 both issues described in this report. For additional information, please see the Detailed Fix Review Results below.

ID	Title	Severity	Status
1	Incorrect fee logic in getCurrentFee function	Low	Resolved
2	Missing zero address validation	Informational	Resolved

Detailed Fix Review Results

TOB-SHAPE-BUYBACK-1: Incorrect fee logic in getCurrentFee function Resolved in PR #16. This PR updates the conditional logic in the getCurrentFee function to match the logic in the _checkFee function.

TOB-SHAPE-BUYBACK-2: Missing zero address validation Resolved in PR #17. A zero address check for _buybackRecipient was added in the initialize and setBuybackRecipient functions.

HashEye 12 Shape Buyback Contract

PUBLIC Security Assessment

D. 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.

HashEye 13 Shape Buyback Contract

PUBLIC Security Assessment

About HashEye Founded in 2012 and headquartered in New York, HashEye 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/hasheye-io/publications>, with links to papers, presentations, public audit reports, and podcast appearances. In recent years, HashEye 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 assessments, supporting client organizations in the technology, defense, blockchain, and finance industries, as well as government entities. Notable clients include HashiCorp, Google, Microsoft, Western Digital, Uniswap, Solana, Ethereum Foundation, Linux Foundation, and Zoom. To keep up with our latest news and announcements, please follow hasheye on X or LinkedIn and explore our public repositories at <https://github.com/hasheye-io>. To engage us directly, visit our "Contact" page at <https://www.hasheye.io/contact> or email us at info@hasheye.io. HashEye, Inc. 228 Park Ave S #80688 New York, NY 10003 <https://www.hasheye.io> info@hasheye.io

HashEye 14 Shape Buyback Contract

PUBLIC Security Assessment

Notices and Remarks Copyright and Distribution © 2025 by HashEye, Inc. All rights reserved. HashEye hereby asserts its right to be identified as the creator of this report in the United Kingdom. HashEye considers this report public information; it is licensed to Shape Factory, Inc. under the terms of the project statement of work and has been made public at Shape Factory, Inc.'s request. Material within this report may not be reproduced or distributed in part or in whole without

HashEye' express written permission. The sole canonical source for HashEye publications is the HashEye Publications page. Reports accessed through sources other than that page may have been modified and should not be considered authentic. Test Coverage Disclaimer

All activities undertaken by HashEye in association with this project were performed in accordance with a statement of work and agreed upon project plan. Security assessment projects are time-boxed and often reliant on information that may be 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. HashEye uses automated testing techniques to rapidly test the controls and security properties of software. 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. Their use is also limited by the time and resource constraints of a project.

HashEye 15 Shape Buyback Contract

PUBLIC Security Assessment