Smart Contract
Security Audit Report

Prepared for River

Prepared by Supremacy

December 03, 2025

Contents

1 Introduction

Given the opportunity to review the design document and related codebase of the River, we outline in the report our systematic approach to evaluate potential security issues in the smart contract(s) implementation, and provide additional suggestions or recommendations for improvement. Our results show that the given version of smart contracts can be further improved due to the presence of several issues related to either security or performance. This document outlines our audit results.

1.1 About Client

River is building a chain-abstraction stablecoin system connecting liquidity across ecosystems. Powered by omni-CDP's satUSD, users can deposit collateral on one chain and mint satUSD on another—no bridges, no wrapping.

Item	Description
Client	River
Project	RiverPointConverter2
Website	https://river.inc
Type	Smart Contract
Languages	Solidity
Platform	EVM-compatible

1.2 Audit Scope

In the following, we show the Git repository of reviewed file and the commit hash used in this security audit:

- Repository: https://github.com/Satoshi-Protocol/tokenomics/blob/feat/pts-conversion-2.0/src/community/pointConverter/RiverPointConverter2.sol
- Commit Hash: d82a4e3d0fcebaf7a92acd3c8565c505a44449a7

1.3 Changelogs

Version	Date	Description
0.1	November 29, 2025	Initial Draft
0.2	December 02, 2025	Release Candidate #1
1.0	December 03, 2025	Final Release

1.4 About Us

Supremacy is a leading blockchain security firm, composed of industry hackers and academic researchers, provide top-notch security solutions through our technology accumulation and innovative research.

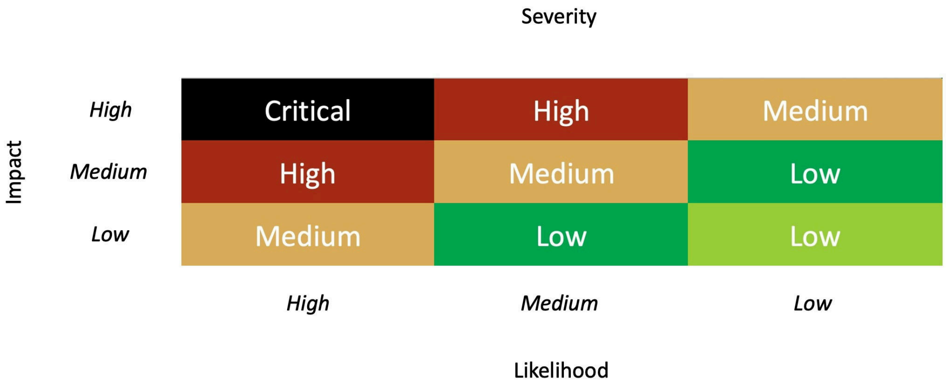
We are reachable at X (https://x.com/SupremacyHQ), or Email (contact@supremacy.email).

1.5 Terminology

For the purpose of this assessment, we adopt the following terminology. To classify the severity of our findings, we determine the likelihood and impact (according to the CVSS risk rating methodology).

- Likelihood represents the likelihood of a finding to be triggered or exploited in practice
- Impact specifies the technical and business-related consequences of a finding
- Severity is derived based on the likelihood and the impact

We categorize the findings into four distinct categories, depending on their severity. These severities are derived from the likelihood and the impact using the following table, following a standard risk assessment procedure.



Severity

Impact	High	Critical	High	Medium
	Medium	High	Medium	Low
	Low	Medium	Low	Low
		High	Medium	Low

Likelihood

As seen in the table above, findings that have both a high likelihood and a high impact are classified as critical. Intuitively, such findings are likely to be triggered and cause significant disruption. Overall, the severity correlates with the associated risk. However, every finding's risk should always be closely checked, regardless of severity.

2 Invariants Assessed

During our audit of the protocol, we performed invariant testing on the protocol's core functions using Foundry. Due to the protocol's highly complex dynamic interactions and the risk of unanticipated edge cases, invariant testing was critical to ensure the correctness of multiple system invariants.

ID	Description	Initial	Remediated	Run Count
1	`invariant_RateMonotonicity()`	✓	✓	50,000+
2	`invariant_PreviewAccuracy()`	✓	✓	50,000+
3	`invariant_RateDecreaseInSameBlock()`	✓	✓	50,000+
4	`invariant_AccessControl()`	✓	✓	50,000+
5	`invariant_DailyLimits()`	✓	✓	50,000+
6	`invariant_TokenConservation()`	✓	✓	50,000+
7	`invariant_UserStateTracking()`	✓	✓	50,000+
8	`invariant_TimestampAlignment()`	✓	✓	50,000+
9	`invariant_RecoveryRateBounds()`	✓	✓	50,000+
10	`invariant_PointBurning()`	✓	✓	50,000+
11	`invariant_ConvertAndStake()`	✓	✓	50,000+
12	`invariant_SlippageProtection()`	✓	✓	50,000+
13	`invariant_PauseState()`	✓	✓	50,000+
14	`invariant_SingleConversionLimit()`	✓	✓	50,000+

3 Findings

The table below summarizes the findings of the audit, including status and severity details.

ID	Severity	Description	Status
1	Medium	Potential Disable Conversion	Fixed
2	Medium	Potential Overflow	Fixed
3	Medium	Missing Coverage Validation	Fixed
4	Informational	Incorrect Variable Naming	Fixed
5	Informational	Lack of Event Records	Fixed
6	Informational	Redundant Code	Fixed
7	Informational	Centralization risk	Acknowledged

3.1 Medium

1. Potential Disable Conversion [Medium]

`Severity: Medium` `Likelihood: Medium` `Impact: Medium`

`Status: Fixed`

Description

The `setConversionDeadline` function allows the owner to set `globalConfig.conversionDeadline` to any value, including a past timestamp, without validation against the current `block.timestamp`. In `_validateConversion`, conversions revert if `block.timestamp > globalConfig.conversionDeadline`.

Impact

A malicious actor could immediately halt all conversions, locking users out of redeeming points and potentially stranding funds. This could be used to manipulate the protocol or prevent redemptions during critical periods, leading to loss of user trust and economic harm.

Recommendation

Add a validation in `setConversionDeadline` to ensure the new deadline is in the future.

Feedback: Fixed in `aa2269c`.

2. Potential Overflow [Medium]

`Severity: Medium` `Likelihood: Medium` `Impact: Medium`

`Status: Fixed`

Description

In the `previewTokens` and `_updateRateState`, PRB Math's exp function is used with `exponent = (k * points) / epochPoints`. If the `owner` sets an excessively large `k` or if `points` are large relative to `epochPoints`, the exponent could exceed PRB Math's safe range (133.084258667509499441 for UD60x18 before overflow), causing reversions. While `_validateConfig` checks `k > 0` and `epochPoints > 0`, there are no upper bounds.

Impact

Conversions could revert for users with large points balances or after owner misconfiguration, leading to DoS for affected users. In extreme cases, it could halt the protocol if not fixed via upgrade.

Recommendation

Add upper bounds in `_validateConfig`, e.g., `if (_cfg.k > 10e18) revert EInvalidIntegralConfig();`.

Feedback: Fixed in `af4bae6`.

3. Missing Coverage Validation [Medium]

`Severity: Medium` `Likelihood: Medium` `Impact: Medium`

`Status: Fixed`

Description

In the `setBaseRatePeriods`, `periods` are validated for sequentiality and non-decreasing rates within each period, but there is no check that the periods collectively cover from `integralConfig.epochStart` to `globalConfig.conversionDeadline`. If gaps exist or periods start after `epochStart`, `getBaseRate` falls back to boundary values, which may not align with intended rates.

Impact

Misconfigured periods could lead to unexpectedly low/high rates during parts of the epoch, resulting in unfair token distributions or economic imbalances.

Recommendation

Enhance `_validateBaseRatePeriods` to check `if (_periods[0].startTime > _cfg.epochStart) revert;` and ensure the last period's `endTime >= deadline`. Make this check dynamic in `setBaseRatePeriods` by referencing current `configs`.

Feedback: Fixed in `975b950`.

3.2 Informational

4. Incorrect Variable Naming [Informational]

`Status: Fixed`

Description

The variable `dailyConversionCount` and function `getDailyConversionCount` imply they track the number of conversions per day. However, they actually accumulate the total points converted per day (`dailyConversionCount += points` in `_updateDailyLimit`).

Recommendation

Rename `dailyConversionCount` to `dailyPointsConverted` and `getDailyConversionCount` to `getDailyPointsConverted`. Update related functions like `getRemainingPointsConvertibleToday` for consistency.

Feedback: Fixed in `355782f`.

5. Lack of Event Records [Informational]

`Status: Fixed`

Description

Functions like `setConversionDeadline` and `setMaxPointsPerConversion` update configs but do not emit events. This omission reduces the contract's transparency and hinders off-chain monitoring tools or users from tracking data effectively.

Recommendation

Consider adding event logging.

Feedback: Fixed in `c349651`.

6. Redundant Code [Informational]

`Status: Fixed`

Description

In `_convert`, `previewTokens` calculates `recoveredRate` for token estimation, then `_updateRateState` recalculates it identically.

Recommendation

Remove unnecessary code.

Feedback: Fixed in `fe9be89`.

7. Centralization risk [Informational]

`Status: Acknowledged`

Description

The privileged function `withdraw` can be used to extract any amount of any token to any address, which may pose a potential centralization risk.

Recommendation

Consider using a multi-sig wallet or timelock to mitigate this centralization risk.

4 Disclaimer

This security audit report does not constitute investment advice or a personal recommendation. It does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Any entity should not rely on this report in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. This security audit report is not an endorsement of any particular project or team, and the report does not guarantee the security of any particular project. This audit does not give any warranties on discovering all security issues of the smart contracts, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues, also cannot make guarantees about any additional code added to the assessed project after the audit version. As one audit-based assessment cannot be considered comprehensive, we always recommend proceeding with independent audits and a public bug bounty program to ensure the security of smart contract(s). Unless explicitly specified, the security of the language itself (e.g., the solidity language), the underlying compiling toolchain and the computing infrastructure are out of the scope.