GA GUARDIAN

Bounce

USDC Updates Review

Security Assessment

January 18th, 2026

Summary

Audit Firm Guardian

Prepared By Felipe Gomez, Ali Shebab

Client Firm Bounce

Final Report Date January 18, 2026

Audit Summary

Bounce engaged Guardian to review the security of their USDC Updates. From the 7th of January to the 9th of January, a team of 2 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Confidence Ranking

Given the lack of High and Critical issues detected during the main review, Guardian assigns a Confidence Ranking of 5 to the protocol. For detailed understanding of the Guardian Confidence Ranking, please see the rubric on the following page.

✅ Verify the authenticity of this report on Guardian's GitHub: https://github.com/guardianaudits

Guardian Confidence Ranking

Confidence Ranking	Definition and Recommendation	Risk Profile
5: Very High Confidence	Codebase is mature, clean, and secure. No High or Critical vulnerabilities were found. Follows modern best practices with high test coverage and thoughtful design. **Recommendation:** Code is highly secure at time of audit. Low risk of latent critical issues.	0 High/Critical findings and few Low/Medium severity findings.
4: High Confidence	Code is clean, well-structured, and adheres to best practices. Only 1 Significant issue was uncovered per week. Design patterns are sound, and test coverage is strong. **Recommendation:** Suitable for deployment after remediations; consider periodic review with changes.	0-1 High/Critical findings per engagement week and little to no Medium severity issues. Varied Low severity findings.
3: Moderate Confidence	Medium-severity and occasional High-severity issues found. Code is functional, but there are concerning areas (e.g., weak modularity, risky patterns). No critical design flaws, though some patterns could lead to issues in edge cases. **Recommendation:** Address issues thoroughly and consider a targeted follow-up audit depending on code changes.	1-2 High/Critical findings per engagement week.
2: Low Confidence	Code shows frequent emergence of Critical/High vulnerabilities. Audit revealed recurring anti-patterns, weak test coverage, or unclear logic. These characteristics suggest a high likelihood of latent issues. **Recommendation:** Post-audit development and a second audit cycle are strongly advised.	2-4 High/Critical findings per engagement week. Or additional High/Critical findings uncovered in remediation review which have not been resolved and confirmed by Guardian.
1: Very Low Confidence	Code has systemic issues. Multiple High/Critical findings (≥5/week), poor security posture, and design flaws that introduce compounding risks. Safety cannot be assured. **Recommendation:** Halt deployment and seek a comprehensive re-audit after substantial refactoring.	≥5 High/Critical findings and overall systemic flaws.

Table of Contents

Project Overview

Project Summary

Project Name	Bounce
Language	Solidity
Codebase	https://github.com/bounce-tech/bounce-contracts
Commit(s)	Main Review commit: 2199398287c87c5ad63cf4d8f3464f03309ba72d Remediation Review commit: 39d0ee246812da00be0ad6bb9707aeb1c7b306b3 Remediation V2 Review commit: 43446a4042035de1c7e84bd338c21b15dc2da770

Audit Summary

Delivery Date	January 18, 2026
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
High	0	0	0	0	0	0
Medium	0	0	0	0	0	0
Low	4	0	0	0	0	4
Info	10	0	0	1	0	9

Audit Scope & Methodology

The following diff is in scope:
https://github.com/bounce-tech/bounce-contracts/compare/648c37fb830d6a760810ea7f1c92f85bf037ee4d...2199398287c87c5ad63cf4d8f3464f03309ba72d

Audit Scope & Methodology

Vulnerability Classifications

Severity	Impact: *High*	Impact: *Medium*	Impact: *Low*
Likelihood: *High*	🔴 Critical	🟠 High	🟡 Medium
Likelihood: *Medium*	🟠 High	🟡 Medium	🟢 Low
Likelihood: *Low*	🟡 Medium	🟢 Low	🟢 Low

Impact

High Significant loss of assets in the protocol, significant harm to a group of users, or a core functionality of the protocol is disrupted.

Medium A small amount of funds can be lost or ancillary functionality of the protocol is affected. The user or protocol may experience reduced or delayed receipt of intended funds.

Low Can lead to any unexpected behavior with some of the protocol's functionalities that is notable but does not meet the criteria for a higher severity.

Likelihood

High The attack is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount gained or the disruption to the protocol.

Medium An attack vector that is only possible in uncommon cases or requires a large amount of capital to exercise relative to the amount gained or the disruption to the protocol.

Low Unlikely to ever occur in production.

Audit Scope & Methodology

Methodology

Guardian is the ultimate standard for Smart Contract security. An engagement with Guardian entails the following:

- Two competing teams of Guardian security researchers performing an independent review.
- A dedicated fuzzing engineer to construct a comprehensive stateful fuzzing suite for the project.
- An engagement lead security researcher coordinating the 2 teams, performing their own analysis, relaying findings to the client, and orchestrating the testing/verification efforts.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts. Comprehensive written tests as a part of a code coverage testing suite.
- Contract fuzzing for increased attack resilience.

Findings & Resolutions

ID	Title	Category	Severity	Status
L-01	Incorrect Decimals	Logical Error	Low	Resolved
L-02	bridgeFromPerp May Bridge	Best Practices	Low	Resolved
L-03	Total Assets Reduced When Bridging	Logical Error	Low	Resolved
L-04	Exchange Rate Drop When Bridging	Logical Error	Low	Resolved
I-01	Agent Activation Not Verified In setAgent	Validation	Info	Resolved
I-02	Unsafe Integer Casting	Best Practices	Info	Resolved
I-03	Complex Position Notional Value Calculations	Informational	Info	Resolved
I-04	Decimal Mismatch	Math	Info	Resolved
I-05	Multiple Documented Functions That Are Removed	Documentation	Info	Resolved
I-06	Unused Imports	Superfluous Code	Info	Resolved

L-01 | Incorrect Decimals

Category	Severity	Location	Status
Logical Error	Low	LeveragedTokenHelper.sol: 96-97	Resolved

Description

In `LeveragedTokenHelper`, the logic used to determine whether a leveraged token is considered held multiplies `balanceOf` (18 decimals) by `exchangeRate` (18 decimals) and directly compares the result against `_MIN_ASSET_VALUE_THRESHOLD = 1e6`.

This produces a value with 36 decimals, while the threshold is expressed in 6-decimal USDC units, making the comparison invalid.

As a result, the `onlyHeld_` filter can behave incorrectly (false positives or negatives) and the raw multiplication risks overflow and revert in view calls for large balances.

Recommendation

Normalize units before comparison by scaling the computed value to USDC decimals, following the same pattern used in `LeveragedToken.ltToBaseAmount`.

Resolution

Bounce Team: The issue was resolved in PR#171.

L-02 | bridgeFromPerp May Bridge

Category	Severity	Location	Status
Best Practices	Low	LeveragedToken.sol: 286-287	Resolved

Description

To bridge USDC from perps to EVM, `bridgeFromPerp` has to be called, which:

1. Transfers USDC from perps to spot

2. Bridges USDC from spot to EVM

However, if there is not enough USDC in perps, the transfer step fails, but the bridging happens anyway.

This could result in an accidental spot-to-EVM bridge when the intent was to bridge from perps, which could happen more often if the protocol has low TVL.

Recommendation

Consider acknowledging and documenting this behavior, so agents are aware of it.

Resolution

Bounce Team: Resolved.

L-03 | Total Assets Reduced When Bridging

Category	Severity	Location	Status
Logical Error	Low	LeveragedToken.sol: 349	Resolved

Description

According to the Hyperliquid docs: A transfer from HyperCore to HyperEVM costs 200k gas at the base gas price of the next HyperEVM block. If the account is only funded with USDC, these bridge fees will be charged in USDC tokens (around 0.0005 USDC per bridge according to internal testing).

Consequently, two issues arise:

• if automation script requires that 100% of the Core funds should be bridged to EVM, the bridge transaction will revert due to insufficient funds to pay for fees.

• the bridgeFromPerp and bridgeFromSpot functions do not execute _checkpoint(), so the next action will calculate fees based on a slightly lower value of totalAssets.

Recommendation

Fund the leveraged token address in HyperCore with HYPE, so that bridge fees are paid with HYPE instead of USDC.

Resolution

Bounce Team: Resolved.

L-04 | Exchange Rate Drop When Bridging

Category	Severity	Location	Status
Logical Error	Low	LeveragedToken.sol: 284	Resolved

Description

When bridging from Core to EVM, the `bridgeFromPerp` transaction will first reduce the Core balance, but the USDC EVM tokens will only be minted in the next EVM block.

Therefore, when the RPC fetches `totalAssets` in the same block as the `bridgeFromPerp` transaction, the `hyperliquidUsdc` call will not detect the Core assets, and the USDC tokens are not yet minted to the LT contract, causing an unexpected exchange rate drop.

Keep in mind that USDC is transferred/minted to the leveraged token contract one block after the `bridgeFromPerp` transaction is confirmed, but these transfers take place in the first position of the block, preventing invalid state if frontrunned.

Recommendation

Consider adding a short delay after bridging funds to EVM and allow `totalAssets` to normalize before proceeding with the automation flow.

Alternatively, listen to the `BridgeFromPerp` event and create a `no actions` delay to prevent executing redeems before state settles.

Resolution

Bounce Team: Resolved.

I-01 | Agent Activation Not Verified In setAgent

Category	Severity	Location	Status
Validation	Info	LeveragedToken.sol	Resolved

Description

`setAgent` does not validate whether the provided `agent_` is eligible to be added as an API wallet on Hyperliquid Core (e.g., whether the address is already activated on Core).

This is intentional behavior: enforcing `coreUserExists(agent_) == false` would prevent setting/removing an agent if the address gets activated in Core outside of this contract.

However, the lack of validation means the owner may set an address that cannot be added as an API wallet, causing `addApiWallet(agent_)` to revert/fail at the Core level and requiring manual operational checks.

Recommendation

Document this operational requirement: before calling `setAgent`, the admin should manually verify that `agent_` is eligible for API wallet addition on Hyperliquid Core.

Resolution

Bounce Team: Resolved.

I-02 | Unsafe Integer Casting

Category	Severity	Location	Status
Best Practices	Info	HyperliquidHandler.sol: 43	Resolved

Description

Unsafe casting is used in `LeveragedToken.sol` `bridgeFromPerp()` `(uint64(amount_))` and in `HyperliquidHandler.sol` `notionalUsdc()` `(uint16(marketId_))`. These casts can silently truncate on overflow.

Recommendation

Consider using `OpenZeppelin`'s `SafeCast` library to ensure safe conversions.

Resolution

Bounce Team: The issue was resolved in PR#172.

I-03 | Complex Position Notional Value Calculations

Category	Severity	Location	Status
Informational	Info	HyperliquidHandler.sol: 39	Resolved

Description

The current `notionalUsdc` function calculates the LT's position notional value based on the position's size, mark price and performs some scaling based on token, size and price decimals.

However, if every leveraged token only manages one hyperliquid position, the `AccountMarginSummary` struct has the `ntlPos` which yields to the same value.

Recommendation

Consider using the `AccountMarginSummary.ntlPos` to avoid rounding errors when scaling the results.

Resolution

Bounce Team: The issue was resolved in PR#170.

I-04 | Decimal Mismatch

Category	Severity	Location	Status
Math	Info	LeveragedTokenHelper.sol: 176	Resolved

Description

The function `getLeveragedTokenPositionData` calculates `netValue_` by subtracting credit value from total assets. However, there is a decimal mismatch between the two operands:

```
uint256 netValue_ = lt_.totalAssets() - lt_.credit().mul(lt_.exchangeRate());
```

• `lt_.totalAssets()`: Returns USDC amount with 6 decimals

• `lt_.credit()`: Returns LT token amount with 18 decimals

• `lt_.exchangeRate()`: Returns exchange rate with 18 decimals

• `lt_.credit().mul(lt_.exchangeRate())`: Results in 18 decimals

Subtracting an 18-decimal value from a 6-decimal value causes an underflow. This will revert and break off-chain monitoring and automation.

Recommendation

Consider scaling the credit value down to 6 decimals before subtraction:

```
uint256 creditValue_ = lt_.credit().mul(lt_.exchangeRate()).scaleTo(baseAssetDecimals_);
uint256 netValue_ = lt_.totalAssets() - creditValue_;
```

This ensures both operands use the same decimal precision (6 decimals) before arithmetic operations.

Resolution

Bounce Team: The issue was resolved in PR#169.

I-05 | Multiple Documented Functions That Are Removed

Category	Severity	Location	Status
Documentation	Info	README.md	Resolved

Description

There are multiple functions that were removed from the code but are still referenced in the

README:

1. `marginUsedAssets`
2. `perpUsdcAssets`
3. `spotBaseAssets`
4. `spotUsdcAssets`

Recommendation

Update the README to reflect the new function names.

Resolution

Bounce Team: The issue was resolved in PR#168.

I-06 | Unused Imports

Category	Severity	Location	Status
Superfluous Code	Info	LeveragedToken.sol: 10-11	Resolved

Description

The `LeveragedToken` imports `HLConversions` and `PrecompileLib` but these are never used.

Recommendation

Remove unused imports.

Resolution

Bounce Team: The issue was resolved in PR#167.

Remediation Findings & Resolutions

ID	Title	Category	Severity	Status
I-01	Missing Event In State Changing Function	Events	Info	Resolved
I-02	Redemption Fee Invariant Can Be Broken	Logical Error	Info	Acknowledged
I-03	Inconsistent Base-Asset Balance Retrieval	Best Practices	Info	Resolved
I-04	Incorrect Note About Account Activation	Documentation	Info	Resolved

I-01 | Missing Event In State Changing Function

Category	Severity	Location	Status
Events	Info	LiquidationPoints.sol: 26	Resolved

Description

The contract mutates state in `claimLiquidationPoints()` by marking claims and appending to `_users` without emitting an event. This makes it harder to monitor, audit, or reconstruct claim history reliably, especially if `users()` becomes unusable due to size.

Recommendation

Emit an event (e.g., `event Claimed(address indexed user)`) inside `claimLiquidationPoints` and consider relying on events instead of keeping an ever-growing on-chain array.

Resolution

Bounce Team: The issue was resolved in PR#177.

I-02 | Redemption Fee Invariant Can Be Broken

Category	Severity	Location	Status
Logical Error	Info	GlobalStorage.sol: 160	Acknowledged

Description

The `executeRedemptionFee` is bounded by `minTransactionSize * 0.5` at the time of setting, but lowering `minTransactionSize` later can make the existing `executeRedemptionFee` exceed this intended bound. No continuous invariant enforcement exists.

Recommendation

Either enforce the invariant on both setters (revalidate and adjust or reject updates that violate the constraint) or compute the max allowed dynamically at use-time instead of set-time.

Resolution

Bounce Team: Acknowledged.

I-03 | Inconsistent Base-Asset Balance Retrieval

Category	Severity	Location	Status
Best Practices	Info	LeveragedTokenHelper.sol: 249	Resolved

Description

`getLeveragedTokenPositionData` returns the `baseAssetContractBalance` as `baseAsset_.balanceOf(leveragedTokenAddress_)`. However, the LT contract exposes a helper function `baseAssetBalance()` that returns the actual base asset balance held by the LT contract.

Recommendation

Consider using `lt_.baseAssetBalance()` instead of `baseAsset_.balanceOf(leveragedTokenAddress_)` to ensure consistency with other functions in the contract that already rely on `baseAssetBalance()` for retrieving the LT's actual base asset balance.

Resolution

Bounce Team: The issue was resolved in PR#175.

I-04 | Incorrect Note About Account Activation

Category	Severity	Location	Status
Documentation	Info	README.md	Resolved

Description

The `README` currently mentions that the Smart Contract must be 'activated', which requires sending a small amount of USDC to that contract on HyperCore. \$5 should be plenty.

However, tha activation fee is 1 stablecoin unit, no matter how much is sent. An account can be activated in Hypercore by transferring 0.01 USDC, but will pay a 1 USDC fee to the network.

Recommendation

Lower the amount sent to activate the contract or fix the `README` comment if a lower amount is already used.

Resolution

Bounce Team: The issue was resolved in PR#174.

Disclaimer

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Guardian to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Guardian’s position is that each company and individual are responsible for their own due diligence and continuous security. Guardian’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Guardian is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

About Guardian

Founded in 2022 by DeFi experts, Guardian is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian upholds best-in-class security while achieving our mission to relentlessly secure DeFi.

To learn more, visit https://guardianaudits.com

To view our audit portfolio, visit https://github.com/guardianaudits

To book an audit, message https://t.me/guardianaudits