STRIKE · SECURITY REVIEW

Strike
Security Review.

A security review of Strike.

CLIENT	PROJECT	AUDITORS	AUDIT WINDOW	FINAL REPORT
Strike	Strike	0xCiphky ., Ferran Celades, Vlad Zotov, Michael Lett	October 16 – October 27, 2026	October 28, 2026

§ 00

Contents.

19 pages: overview, confidence ranking, scope, 5 findings, disclaimer, and about Guardian.

§ 01 Overview P. 03

§ 02 Confidence Ranking P. 04

§ 03 Scope & Classifications P. 05

Files in Scope P. 06

§ 04 Findings P. 07

Main Review · 5 findings P. 08

§ 05 Disclaimer P. 18

§ 06 About Guardian P. 19

§ 01

Overview.

Guardian reviewed Strike across 2 review rounds. The final report includes 2 High, 2 Medium, 1 Low findings.

AUDIT FIRM
Guardian

CLIENT
Strike

PROJECT
Strike

PREPARED BY
0xCiphky ., Ferran Celades, Vlad Zotov, Michael Lett

AUDIT WINDOW
October 16 – October 27, 2026

FINAL REPORT
October 28, 2026

REPOSITORY
https://github.com/rohancherukuri14/strike

METHODOLOGY
Smart contract · Manual · Main Review · Remediation Review

MAIN REVIEW COMMIT
af6bba99f715

REMEDIATION REVIEW COMMIT
dc484855d6b7

1.1 Confidence Ranking

4 / 5

High Confidence

Guardian assigns a Confidence Ranking of 4 out of 5, combining finding severity, remediation outcome, and the reviewer's assessment of latent risk. The full scale is on the next page.

1	2	3	4	5
VERY LOW CONFIDENCE	LOW CONFIDENCE	MODERATE CONFIDENCE	HIGH CONFIDENCE	VERY HIGH CONFIDENCE

1.2 Findings at a Glance

SEVERITY	TOTAL	PENDING	ACK	PARTIAL	RESOLVED	FIX REVIEW
Critical	0	0	0	0	0	0
High	2	0	0	0	2	0
Medium	2	0	0	0	2	0
Low	1	0	0	0	1	0
Informational	0	0	0	0	0	0
All severities	5	0	0	0	5	0

§ 02

Confidence Ranking.

Guardian assigns a project-level confidence score using the scale below. The ranking combines finding severity, remediation outcome, and the reviewer's assessment of latent risk.

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.	0 High/Critical findings and few Low/Medium severity findings.
4 High Confidence ASSIGNED	Code is clean, well-structured, and adheres to best practices. 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 findings.
3 Moderate Confidence	Code is functional but shows gaps in defensive programming or test coverage. Recommendation: remediate the findings and re-review the changes before deployment.	Some High/Critical findings, with various Medium severity findings.
2 Low Confidence	Recurring issues suggest systemic gaps rather than isolated mistakes. Recommendation: remediate, then commission a follow-up review at a frozen commit.	Over 6 High and Critical findings across the engagement.
1 Very Low Confidence	The volume and severity of findings indicate the codebase is not ready for production. Recommendation: substantial rework, then a full re-review.	Over 12 High and Critical findings across the engagement.

§ 03

Scope & Classifications.

Guardian reviewed Strike in `https://github.com/rohancherukuri14/strike`.
The scope includes 579 source lines across 2 files.

3.1 Severity Matrix

	IMP · HIGH	IMP · MED	IMP · LOW
LIK · HIGH	Critical	High	Medium
LIK · MED	High	Medium	Low
LIK · LOW	Medium	Low	Low

IMPACT · HIGH	Significant loss of assets, harm to a group of users, or core protocol functionality is disrupted.
IMPACT · MEDIUM	Small amount of funds can be lost, or ancillary functionality is affected. Users may experience reduced or delayed receipt.
IMPACT · LOW	Unexpected behavior in some functionality that is notable but does not meet the criteria for higher severity.
LIKELIHOOD · HIGH	Possible with reasonable assumptions that mimic on-chain conditions; cost is low compared to amount gained.
LIKELIHOOD · MEDIUM	Possible only in uncommon cases or requires a large amount of capital relative to disruption caused.
LIKELIHOOD · LOW	Unlikely to ever occur in production.

Files in Scope.

FILE	NSLOC	TOTAL
MAIN REVIEW		
contracts/arbitrum/src/CctpDepositRelay.sol	108	237
contracts/arbitrum/src/EpochVault.sol	453	792
REMEDIATION REVIEW		
contracts/arbitrum/src/CctpDepositRelay.sol	0	237
contracts/arbitrum/src/EpochVault.sol	18	832

§ 04

Findings.

Grouped by round. Visible IDs reset by severity within each round, so a finding is named the same way in the report as it was in the channel it was written in. Status is where each finding ended up; its own page says how.

ID	TITLE	CATEGORY	SEVERITY	STATUS	PAGE
MAIN REVIEW					
H-01	Attesters accept gaps in v2 history, allowing inflated EpochVault withdrawals	accounting	HIGH	RESOLVED	p. 8
H-02	Paginated forced exits can commit an underbacked EpochVault root and stall subsequent publication	accounting	HIGH	RESOLVED	p. 10
M-01	Oversized epoch-data events prevent the emergency client from building exits	DoS	MEDIUM	RESOLVED	p. 13
M-02	An uninitialized attester quorum can re-sign consumed v2 spans	economic/accounting	MEDIUM	RESOLVED	p. 15
L-01	Venue genesis includes a zero-balance maker account that does not match EpochVault.EMPTY_ROOT	liveness/accounting	LOW	RESOLVED	p. 17

HIGH **H-01** RESOLVED

Attesters accept gaps in v2 history, allowing inflated EpochVault withdrawals

CATEGORY	accounting
LOCATION	contracts/arbitrum/src/EpochVault.sol:publishEpoch:L359,_checkAttestations:L386-L400; zk/proverd/src/epoch_attest.rs:attest_epoch:L331-L373,L427-L460; zk/transition-v3/src/lib.rs:EpochPublic:L843-L870
REVIEW	Main Review

DESCRIPTION

EpochVault relies on the attester quorum to confirm that each epoch's balance changes match the public v2 transition log. Each attester records where the last span it signed ended, but `attest_epoch` rejects the next span only when it starts before that offset:

```
if !exact_retry && span_start < h.end_offset
```

A span starting after the previous end is accepted. To reach the caller-supplied `v2From`, the attester advances its tracker across the intervening frames without including their balance changes in the signed delta list. Neither `EpochPublic` nor the attestation signed by the quorum records the v2 interval, so `EpochVault` cannot detect the missing history.

An epoch producer can use this to skip a frame in which an attacker-controlled account loses funds, then start the next attested span at a later frame that credits the account. Honest attesters derive and sign the deltas from that later span. The v3 proof applies those deltas to the previous v3 root, where the skipped loss was never recorded.

For example, an account with a 1,000 balance loses 1,000 in the skipped frame and receives 1,000 in a later frame. The complete history leaves it with 1,000, but the accepted epoch applies only the later credit and gives it a 2,000 balance. The account can sign a 2,000 withdrawal, `publishEpoch` accepts the proof and quorum signatures, and the account claims its own deposit plus another user's collateral. The attack requires control of the epoch-producing path. The normal keeper then relays the request to the private attesters; no attester key or quorum member must be compromised.

RECOMMENDATION

Require every non-exact request to start exactly at the attester's high-water end offset. If an attester missed a published epoch, catch it up by replaying the missed epoch spans in order and matching their delta commitments before returning it to the quorum. Preserve exact retries for requests that were signed but not published.

H-01 Attesters accept gaps in v2 history, allowing inflated EpochVault withdrawals

CONTINUED

PROOF OF CONCEPT https://github.com/GuardianOrg/SentryRepo-Strike--Round-1-01a14630ce507ea88b8dfd994a9ca715/blob/refinery/poc-01a14630-e4e5-7789-a2c4-28a2a24da52d/test/refinery-poc/honest-attesters-sign-a-forward-v2-span-gap-erasing-losses-a-3e6f1fcd.rs

RESOLUTION

Status: Resolved.

https://github.com/rohancherukuri14/strike/commit/cf17e559f60527b8f6996b4b5d5d3e5dcb05169a

HIGH **H-02** RESOLVED

Paginated forced exits can commit an underbacked EpochVault root and stall subsequent publication

CATEGORY	accounting
LOCATION	`contracts/arbitrum/src/EpochVault.sol:publishEpoch:L320-L375,_payBatch:L426-L454,claim/_claim:L575-L602,exit:L607-L653,_openExodus:L655-L665,refundUnfoldedDeposit:L754-L769; venue/src/epoch.rs:cut_pending_budget:L666-L743,cut_with:L745-L897; venue/src/venue.rs:epoch_merge:L1702-L1707,pay_serviced_exits:L1712-L1746; venue/src/main.rs:L1914-L1938; zk/transition-v3/src/lib.rs:L1434-L1483; zk/proverd/src/epoch_loop.rs:L474-L624`
REVIEW	Main Review

DESCRIPTION

`EpochVault` can accept a valid paginated epoch whose forced-exit entitlements consume funds needed to back the other positive balances committed by the same root. The next page then reverts after the Venue has already advanced to it, stalling further publication. If exodus opens while publication is stalled, users are paid from an underfunded pool and controlled accounts can profit.

When `cut_pending_budget()` limits an epoch to 1,000 payouts, it carries the remaining deposit-log entries into a later page. It subtracts those carried deposits from the prospective proven account view, but the negative-view guard only runs when signed withdrawals are selected and only clears those withdrawals. `cut_with()` can therefore derive and adopt a root containing a negative account while the transition separately pays each folded forced-exit requester its entire positive cash balance.

`EpochVault._payBatch()` checks that the vault can cover existing entitlements, the new batch, and unfolded deposits. It does not account for the positive balances committed by the accepted root. This permits the first page to create claimable entitlements using funds still needed to back those balances.

With 2,001 exit requests and the production-default 1,000-payout limit:

H-02 Paginated forced exits can commit an underbacked EpochVault root and stall subsequent publication CONTINUED

1. The vault holds 62,001 USDC.
2. Page one creates 40,448.400000 USDC of entitlements while 21,000 USDC remains unfolded. The required reserve is 61,448.400000 USDC, so `EpochVault` accepts the page with only 552.600000 USDC left for the root's 20,153 USDC of positive balances.
3. Page two contains another 11,000 USDC of payouts while 20,000 USDC remains unfolded. Together with the existing entitlements, publication requires 71,448.400000 USDC and reverts with `overcommitted`.
4. The Venue has already adopted and debited page two, while the prover continues retrying that rejected page. Later Venue roots also descend from page two and cannot publish against the page-one root held by `EpochVault`.

After the exodus delay, attacker-controlled accounts can claim 40,448.400000 USDC of accepted entitlements and refund 21,000 USDC of unfolded deposits. The attacker controls 10,001 USDC of the remaining positive leaves and receives 274.229772 USDC from exodus. The remaining 152 USDC belongs to the configured treasury, which receives 4.167875 USDC from exodus. The attacker's total recovery is 61,722.629772 USDC after contributing 52,001 USDC, producing 9,721.629772 USDC of profit before gas if the deficit is not recapitalized before exodus; the trading and settlement fees are already reflected in these balances.

An unrelated user whose accepted leaf contains 10,000 USDC receives only 274.202352 USDC from the 552.600000 USDC exodus pool, losing 9,725.797648 USDC. The attacker can open long positions before requesting exit. The requests freeze voluntary trading but leave the positions outstanding, allowing settlement to move the still-unfolded debtor deposit into the frozen requesters before the first page is cut.

RECOMMENDATION

Do not adopt a cut when subtracting deposits that remain unfolded and refundable creates a negative prospective balance. Apply this check even when the page contains only forced exits. Because withholding those exits would stall the ordered deposit log, change the pagination or deposit-crediting design so that the backing deposit can be folded before its value is paid or committed elsewhere. Keep local adoption provisional until publication succeeds, or support rollback to the last accepted epoch.

RESOLUTION

Status: Resolved.

H-02 Paginated forced exits can commit an underbacked EpochVault root and stall subsequent publication CONTINUED

https://github.com/rohancherukuri14/strike/commit/5c51f6f6516ad15bedf2813a8b02f39e0799da62 https://github.com/rohancherukuri14/strike/commit/aaa2df74b0510975852075829302 8a03ec583699 https://github.com/rohancherukuri14/strike/commit/3d5d607ad7ee5fe56d9b6b2309500ee3d7e09de6

MEDIUM **M-01** RESOLVED

Oversized epoch-data events prevent the emergency client from building exits

CATEGORY	DoS
LOCATION	`contracts/arbitrum/src/EpochVault.sol:302-306; supporting impact in zk/exit/src/main.rs:176-235`
REVIEW	Main Review

DESCRIPTION

EpochVault emits each epoch's encrypted account rows in on-chain logs so that users can reconstruct an exit without data held by Strike. The public custody guide instructs users to run `strike-exit`, and the terminal's downloaded exit kit saves that same command.

However, `postEpochData` is permissionless and only requires `epoch > lastEpoch` and a nonempty chunk. It accepts values outside the `uint64` epoch domain used by the v3 transition, including `type(uint256).max`, and remains callable after `exodusOpen` has latched.

`strike-exit` first retrieves every `EpochDataPosted` event from the requested history. It then converts each event's `uint256` epoch with `to::<u64>()` while building `chunk_logs`, before selecting the `(epoch, publisher)` pairs referenced by `EpochPublished` events. The pinned `ruint` conversion panics when the value does not fit in `u64`. Consequently, a single call to `postEpochData(type(uint256).max, hex"01")` causes every unpatched scan containing that event to terminate before it produces exit calldata.

An attacker can submit the event after exodus opens. At that point no later epoch can be published, while choosing a `--from-block` after the hostile event also removes earlier account rows required to rebuild the frozen `currentRoot`. The generated exit kit does not contain a Merkle proof; it directs users to generate one with the affected client.

Impact: Positive-balance users cannot generate the Merkle proof needed to exit with the provided recovery tooling until a compatible patched client is available.

RECOMMENDATION

Constrain `postEpochData` to the epoch domain used by the transition and recovery client by requiring `epoch <= type(uint64).max`.

RESOLUTION

Status: Resolved.

M-01 Oversized epoch-data events prevent the emergency client from building exits CONTINUED

https://github.com/rohancherukuri14/strike/commit/8ac90b8bb7cd51ee4ca5a78ab2e899f04c01c13e

MEDIUM M-02

RESOLVED

An uninitialized attester quorum can re-sign consumed v2 spans

CATEGORY	economic/accounting
LOCATION	`contracts/arbitrum/src/EpochVault.sol:publishEpoch:L320-L375,proposeAttesters/activateAttesters:L459-L488; zk/proverd/src/epoch_attest.rs:HighWater:L189-L226,attest_epoch:L296-L463`
REVIEW	Main Review

DESCRIPTION

`EpochVault` does not record the v2 log position behind its current v3 state. Each attester keeps its last signed position in a local `HighWater` file. If that file is missing, `HighWater::load()` returns `None`, and `attest_epoch()` skips both the last-epoch check and the overlapping-span check.

If enough active attesters lack this checkpoint to meet the threshold, a compromised venue can build the next v3 epoch from the current v3 root while giving the attesters a previously consumed `v2From` / `v2To` span. The normal keeper forwards the venue's epoch request to the private attester endpoints, so the attacker does not need direct access to them. Each uninitialized attester re-derives the historical deltas and signs because it has no record that the span was already used.

The normal prover aborts if any contacted attester returns an `INCIDENT`, even after collecting a threshold of signatures. The venue-only scenario therefore requires that no contacted signer report an overlap incident.

The guest applies those deltas to the current v3 state before processing withdrawals. `EpochVault` checks the proof, v3 root, epoch number, deposit accumulator and quorum, but it never receives the v2 span. It therefore cannot distinguish this epoch from one built from the next unused span.

The attack requires a signing threshold without initialized checkpoints and control of the venue's epoch output, or control of the keeper/prover path together with the current witness data. It must happen before those attesters process a legitimate request. Replacing one signer in a 2-of-3 set is not enough while the other two retain their checkpoints. The same gap exists at deployment: `lastEpoch == 0` does not authenticate the v2 position where a new vault should begin.

This violates the attester quorum's intended role of preventing a compromised venue or prover from assigning unauthorized deltas: under these conditions, the compromised component can obtain quorum signatures from honest but uninitialized attesters.

M-02 An uninitialized attester quorum can re-sign consumed v2 spans CONTINUED

Alice starts with 1,000 and Bob with zero. A legitimate 500-unit transfer leaves each with 500. Reapplying that span leaves Alice with zero and Bob with 1,000. If Bob is attacker-controlled, he can sign a 1,000-unit withdrawal, which is 500 more than his legitimate balance and reduces the shared custody backing Alice by 500. Loss is limited by the duplicated credit and available vault custody.

RECOMMENDATION

Fail closed if the high-water checkpoint is missing. For a new vault, initialize attesters with an authenticated genesis v2 root and log offset. During rotation or rebuild, use the outgoing quorum's authenticated checkpoint for the last published epoch. Verify that the local log reaches the checkpoint and keep it current through handover, accounting for intervening publications and outstanding signed requests. Complete initialization before activating enough replacement attesters to meet the threshold.

PROOF OF CONCEPT https://github.com/GuardianOrg/SentryRepo-Strike--Round-1-01a14630ce507ea88b8dfd994a9ca715/blob/refinery/poc-01a14630-e4e5-7789-a2c4-28a2a24da52d/test/refinery-poc/fresh-markless-attesters-can-re-apply-an-earlier-v2-delta-an-aedc63c1.rs

RESOLUTION

Status: Resolved.

https://github.com/rohancherukuri14/strike/commit/79f9f118c440ce26e4a3590ab73e191527570fbe

LOW **L-01** RESOLVED

Venue genesis includes a zero-balance maker account that does not match EpochVault.EMPTY_ROOT

CATEGORY	liveness/accounting
LOCATION	packages/strike/venue/src/venue.rs:epoch_accounts:L1474-L1488,epoch_genesis:L1530-L1558; packages/strike/contracts/arbitrum/src/EpochVault.sol:constructor:L230-L234,publishEpoch:L330-L334
REVIEW	Main Review

DESCRIPTION

`Venue::with_config` always creates the maker account, and `epoch_accounts` includes it even when its cash balance is zero. As a result, `epoch_genesis` constructs a non-empty ledger root.

`EpochVault` initializes `currentRoot` to `EMPTY_ROOT`, while the first venue epoch uses its generated genesis root as `prevRoot`. If the current components are deployed together, `publishEpoch` rejects the first epoch with `wrong prior root`.

The genesis mismatch prevents the venue from publishing epochs, blocking normal withdrawals. Depositors can reclaim their deposits directly from the vault after the exodus delay.

RECOMMENDATION

Exclude the zero-balance maker account from the venue's epoch genesis, or otherwise ensure the venue begins from `EpochVault.EMPTY_ROOT`. Verify the venue genesis root against `EpochVault.currentRoot` before enabling deposits.

PROOF OF CONCEPT https://github.com/GuardianOrg/SentryRepo-Strike--Round-1-01a14630ce507ea88b8dfd994a9ca715/blob/refinery/poc-01a14630-e4e5-7789-a2c4-28a2a24da52d/test/refinery-poc/zero-cash-maker-seat-makes-the-venue-genesis-root-differ-fro-10a202a0.sol

RESOLUTION

Status: Resolved.

https://github.com/rohancherukuri14/strike/commit/ec1081cc4e6bda78a8b3036e01eaedc8900a55e3

§ 05

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.

§ 06

About Guardian.

Guardian is a smart contract security firm. Every engagement is worked by two independent teams competing on the same scope, each with its own tooling and methods, so no single reviewer's blind spot survives.

$45B+
DIGITAL ASSETS UNDER REVIEW

200+
ENGAGEMENTS COMPLETED

500+
CRITICAL & HIGH FINDINGS REPORTED

Reach us at audits@guardianaudits.com.