



TECHNICAL SECURITY ASSESSMENT

Fenwick Lending

Formal Verification · Lending

REPORT ID QX-TSA-2026-002

QUANTIX RESEARCH ASSESSMENT

Core accounting, interest accrual and liquidation engine, verified against 12 specified properties.

Document status	Published
Published	30 September 2026
Review period	Ecosystems
31 August 2026 – 18 September 2026	Arbitrum
Languages	Review effort
Solidity	Two verification engineers, fifteen working days
Reviewed commit	Remediation commit
0ce97e1de9ab9b08986888c883206cffe98ee76f	4e9804ddfe0e800b1dc64b0b4f1f166e5b2d904c

FINDINGS BY SEVERITY

5 findings · 100% remediated

Critical	High	Medium	Low	Informational
0	1	2	1	1

This document is a Quantix research assessment of Fenwick Lending, a protocol design from Quantix Protocol Research. It is not a client engagement, it does not describe a deployed system, and it is not a certification.

DOCUMENT CONTROL

Document ID	QX-TSA-2026-002
Title	Technical Security Assessment: Fenwick Lending
Version	1.1
Status	Published
Assessment type	Formal Verification
Review period	31 August 2026 – 18 September 2026
Published	30 September 2026
Reviewed commit	0ce97e1de9ab9b08986888c883206cffe98ee76f

CONTENTS

1	Executive summary	3
2	Scope	4
3	Protocol architecture	4
4	Threat model	5
5	Methodology and severity classification	6
6	Findings summary	7
7	Detailed findings	8
	FL-01 Partial liquidation can leave debt with no collateral and no write-off	8
	FL-02 Reserve withdrawal is bounded by cash, not by accumulated reserves	9
	FL-03 Repaying on behalf of another account can strand surplus tokens	10
	FL-04 Borrow rate decreases by one unit at the kink due to integer division	11
	FL-05 Verification relies on fresh prices and sequencer availability	11
8	Severity and risk analysis	13
9	Remediation status	13
10	Formal verification and invariants	14
11	Recommendations	15
12	Limitations	15
A	Technical appendix	16

1

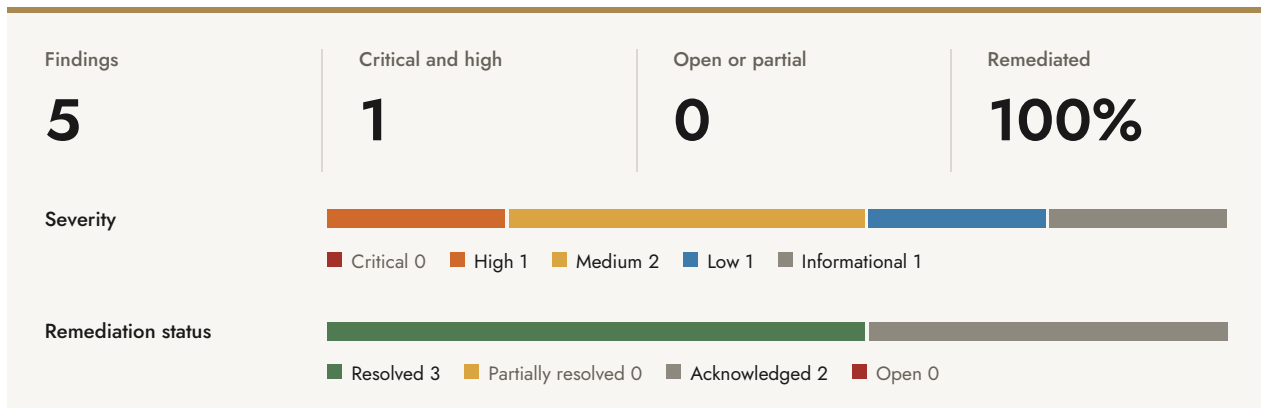
Executive summary

Fenwick Lending is an isolated-market, over-collateralised lending protocol on Arbitrum. This engagement verified properties whose violation would create bad debt or move value between suppliers and borrowers without a corresponding action.

Twelve properties were specified with the protocol team and checked with an SMT-based prover. Eight were proved on the original commit. Three were violated, each with a concrete counterexample that became a finding (FL-01 to FL-03); all three were fixed and re-proved on the remediation commit. One property holds only for bounded executions and one is an explicit assumption about oracle freshness.

Results hold under the stated assumptions, most importantly that oracle prices are positive, fresh and not served during sequencer downtime. Those assumptions are part of the result and must be monitored in production.

AT A GLANCE



SEVERITY SUMMARY

Severity	Definition	Count	Open
■ Critical	Direct loss of funds or permanent loss of control is achievable with little or no precondition.	0	0
■ High	Material loss or protocol disruption is achievable under realistic conditions.	1	0
■ Medium	Limited loss, degraded functionality, or impact that depends on specific conditions.	2	0
■ Low	Minor impact, defence-in-depth gaps, or issues with unlikely preconditions.	1	0
■ Informational	Code quality, documentation or best-practice observations without direct security impact.	1	0

2

Scope

Core accounting, interest accrual and liquidation engine, verified against 12 specified properties.

Repository	github.com/example-org/fenwick-lending
Reviewed commit	0ce97e1de9ab9b08986888c883206cffe98ee76f
Remediation commit	4e9804ddfe0e800b1dc64b0b4f1f166e5b2d904c
Review period	31 August 2026 – 18 September 2026
Review effort	Two verification engineers, fifteen working days

Path	Description	nSLOC
src/MarketCore.sol	Supply, borrow, repay and withdraw; exchange-rate accounting	538
src/InterestRateModel.sol	Utilisation-based borrow and supply rates with a kink	96
src/LiquidationEngine.sol	Health checks, close factor, incentive and collateral seizure	274
src/ReserveManager.sol	Protocol reserves and reserve withdrawals	88
spec/	Formal specification, harnesses and rule files (handed over to the protocol team)	–
Total		996

Out of scope

- OracleRouter (sequencer-uptime and staleness checks are an explicit assumption, P-12)
- Governance and timelock contracts
- Off-chain liquidation bots and keepers
- Market listing and risk-parameter selection

3

Protocol architecture

Each market pairs one collateral asset with one debt asset. MarketCore tracks cash, total borrows, reserves and per-account balances; suppliers hold an exchange-rate-based claim on the pool, borrowers hold debt indexed by a global borrow index.

InterestRateModel returns borrow and supply rates from utilisation with a kink: rates rise slowly to the kink, then steeply. Interest accrues lazily on every state-changing call.

LiquidationEngine checks account health against the market's liquidation threshold and lets anyone repay part of an unhealthy account's debt in exchange for collateral plus an incentive, capped by a close factor. OracleRouter provides prices and rejects stale or sequencer-down rounds.

Component	Role	Trust level
MarketCore	Balances, exchange rate, borrow index, accrual	Trusted
InterestRateModel	Kinked utilisation-based rates	Trusted
LiquidationEngine	Health checks and partial liquidations	Trusted
ReserveManager	Reserve accounting and withdrawals	Trusted
OracleRouter	Price feeds with staleness and sequencer checks	Semi-trusted
Arbitrum sequencer	Transaction ordering on L2	Semi-trusted
Liquidators	Permissionless liquidation callers	Untrusted

Trust assumptions

- Oracle prices are positive, no older than the feed heartbeat, and not consumed while the L2 sequencer is down or within its grace period (P-12).
- The reserve admin is a timelocked multisig; its key is not compromised.
- Collateral and debt assets are standard ERC-20 tokens without fees on transfer or rebasing.

4

Threat model

Assets

- Supplier deposits and accrued interest
- Borrower collateral
- Protocol reserves
- Market solvency (no unbacked debt)

Actors

Actor	Capability	Trust
Borrower	Supplies collateral, borrows, repays	Untrusted
Liquidator	Liquidates unhealthy accounts for an incentive	Untrusted
Reserve admin	Withdraws protocol reserves	Trusted
Oracle reporters	Publish prices consumed by the market	Semi-trusted

Threats

Threat	Actor and vector	Mitigation	Findings
Unbacked debt inflates supplier exchange rate	Liquidator: Liquidation that seizes all collateral but leaves debt	Write off residual debt in the same call	FL-01
Reserve admin withdraws supplier funds	Reserve admin: Withdrawal bounded only by cash	Bound withdrawals by accumulated reserves	FL-02

Threat	Actor and vector	Mitigation	Findings
User funds stranded by over-repayment	Borrower: Repay on behalf with amount above debt	Cap transfer at outstanding debt	FL-03
Solvency checks using stale prices	Oracle reporters: Stale round or sequencer downtime	OracleRouter checks (assumed, out of scope)	FL-05

5

Methodology and severity classification

1. Requirements workshop with the protocol team to agree properties in plain language before formalisation
2. Formal specification of 12 properties covering solvency, conservation, monotonicity and access control
3. Verification with an SMT-based prover over unbounded state; bounded model checking where the prover timed out
4. Mutation testing of the specification: each rule was shown to fail against a deliberately broken implementation
5. Counterexample analysis and minimisation for every violated property
6. Re-verification of all properties against the remediation commit

Severity classification

Each finding is rated on the impact it would have if exploited and the likelihood that its preconditions occur. The rating reflects the code at the reviewed commit.

Severity	Meaning
■ Critical	Direct loss of funds or permanent loss of control is achievable with little or no precondition.
■ High	Material loss or protocol disruption is achievable under realistic conditions.
■ Medium	Limited loss, degraded functionality, or impact that depends on specific conditions.
■ Low	Minor impact, defence-in-depth gaps, or issues with unlikely preconditions.
■ Informational	Code quality, documentation or best-practice observations without direct security impact.

Remediation status

Status	Meaning
Resolved	Fixed and verified in the remediation review.
Partially resolved	Mitigated; the residual risk is documented.
Acknowledged	Accepted by the protocol team with a stated rationale.
Open	Not yet addressed at the reviewed remediation commit.

6

Findings summary

ID	Title	Severity	Category	Status
FL-01	Partial liquidation can leave debt with no collateral and no write-off	High	Business logic	Resolved
FL-02	Reserve withdrawal is bounded by cash, not by accumulated reserves	Medium	Access control	Resolved
FL-03	Repaying on behalf of another account can strand surplus tokens	Medium	Business logic	Resolved
FL-04	Borrow rate decreases by one unit at the kink due to integer division	Low	Precision and rounding	Acknowledged
FL-05	Verification relies on fresh prices and sequencer availability	Informational	Configuration	Acknowledged

7

Detailed findings

FL-01 ■ High Resolved

Partial liquidation can leave debt with no collateral and no write-off

src/LiquidationEngine.sol:L118-L164

Category	CWE	Likelihood	Impact
Business logic	CWE-682	Medium	High

DESCRIPTION

Property P-05 failed. When an account's collateral value is close to its debt, the engine computes collateral to seize as $\text{repay} \times (1 + \text{incentive}) / \text{price}$, rounds it up, and only afterwards applies the close-factor cap to the repay amount.

The prover produced a sequence in which the rounded seizure equals the account's entire collateral while the capped repayment covers only part of the debt. The account ends with zero collateral and positive debt. Nothing marks the remainder as bad debt, so it stays in `totalBorrows` and keeps accruing interest.

Because the supplier exchange rate is computed from $\text{cash} + \text{totalBorrows} - \text{totalReserves}$, phantom debt inflates the rate that suppliers redeem at.

PRECONDITIONS

- An account becomes undercollateralised close to the liquidation boundary
- A liquidator repays the maximum allowed by the close factor

EXPLOIT SCENARIO

1. Collateral price drops so that an account's collateral is worth slightly more than its debt.
2. A liquidator repays the close-factor maximum and seizes collateral including the incentive.
3. Rounding makes the seizure equal to all remaining collateral; residual debt remains.
4. Residual debt accrues interest; suppliers who withdraw early are paid from value that does not exist.

IMPACT

Unbacked debt inflates the exchange rate. Early withdrawers are overpaid and the shortfall falls on the last suppliers to exit. Each occurrence is small, but they accumulate in volatile markets.

RECOMMENDATION

When a liquidation reduces collateral to zero, write off the remaining debt against reserves (or socialise it explicitly) in the same call and emit an event. Compute the close-factor cap before the incentive.

FIX

src/LiquidationEngine.sol	DIFF
1 uint256 seize = _collateralToSeize(repayCapped);	
2 _seize(borrower, liquidator, seize);	
3 + if (collateralOf(borrower) == 0 && debtOf(borrower) > 0) {	
4 + uint256 bad = debtOf(borrower);	
5 + _writeOffAgainstReserves(borrower, bad);	
6 + emit BadDebtRealised(borrower, bad);	
7 + }	

RESOLUTION

Residual debt is written off against reserves in the same transaction and BadDebtRealised is emitted. P-05 re-proved.

REFERENCES

- CWE-682: Incorrect Calculation
<https://cwe.mitre.org/data/definitions/682.html>

FL-02 ■ Medium Resolved

Reserve withdrawal is bounded by cash, not by accumulated reserves

src/ReserveManager.sol:L57

Category	CWE	Likelihood	Impact
Access control	CWE-284	Low	High

DESCRIPTION

Property P-02 failed. `withdrawReserves(amount)` checks `amount <= cash` but not `amount <= totalReserves`. The prover produced a counterexample in which the reserve admin withdraws more than the protocol has accumulated, taking supplier cash and reducing the exchange rate.

The function is restricted to the reserve admin (P-08 proved), so this is not exploitable by arbitrary users. It does mean that the reserve role is effectively able to withdraw all market liquidity, which is a larger privilege than documented.

PRECONDITIONS

- The reserve admin key is compromised or makes an error

EXPLOIT SCENARIO

1. Reserve admin calls `withdrawReserves(cash)`.
2. Cash is transferred out; `totalReserves` becomes negative in effect; the exchange rate drops.

IMPACT

A compromised or mistaken reserve admin can transfer supplier funds out of the market.

RECOMMENDATION

Require `amount <= totalReserves` and decrement reserves before transferring.

FIX

src/ReserveManager.sol	DIFF
1 function withdrawReserves(uint256 amount) external onlyReserveAdmin {	
2 - require(amount <= cash(), "cash");	
3 + if (amount > totalReserves) revert ExceedsReserves(amount, totalReserves);	
4 + totalReserves -= amount;	
5 asset.safeTransfer(reserveAdmin, amount);	
6 }	

RESOLUTION

Bound added and reserves decremented before transfer. P-02 re-proved.

REFERENCES

- CWE-284: Improper Access Control
<https://cwe.mitre.org/data/definitions/284.html>

FL-03 ■ Medium Resolved

Repaying on behalf of another account can strand surplus tokens

src/MarketCore.sol:L301-L327

Category	CWE	Likelihood	Impact
Business logic	CWE-682	Medium	Medium

DESCRIPTION

Property P-06 failed. `repayBorrowBehalf(borrower, amount)` transfers `amount` from the caller in full but reduces the borrower's debt by at most the outstanding balance. Any surplus stays in the contract as unaccounted cash.

Integrations commonly pass a slightly larger amount than the quoted debt to cover interest that accrues before execution, so the surplus case is routine rather than exotic.

PRECONDITIONS

- Caller passes an amount above the outstanding debt

EXPLOIT SCENARIO

1. A repayment helper sends debt + 1% buffer.
2. The buffer is transferred in but not credited; the caller loses it.

IMPACT

Users who over-repay lose the surplus. Unaccounted cash also breaks P-01 until it is swept.

RECOMMENDATION

Cap the transferred amount at the outstanding debt and support `type(uint256).max` as "repay all".

FIX

src/MarketCore.sol	DIFF
1 function repayBorrowBehalf(address borrower, uint256 amount) external {	
2 _accrue();	
3 - asset.safeTransferFrom(msg.sender, address(this), amount);	
4 - _reduceDebt(borrower, Math.min(amount, debtOf(borrower)));	
5 + uint256 owed = debtOf(borrower);	
6 + uint256 pay = amount == type(uint256).max ? owed : Math.min(amount, owed);	
7 + asset.safeTransferFrom(msg.sender, address(this), pay);	
8 + _reduceDebt(borrower, pay);	
9 }	

RESOLUTION

Amount capped before transfer. P-06 re-proved.

REFERENCES

- CWE-682: Incorrect Calculation
<https://cwe.mitre.org/data/definitions/682.html>

FL-04 ■ Low Acknowledged

Borrow rate decreases by one unit at the kink due to integer division

src/InterestRateModel.sol:L44-L61

Category	CWE	Likelihood	Impact
Precision and rounding	CWE-682	Low	Low

DESCRIPTION

P-07 was provable only after excluding a one-unit discontinuity at the kink. The two branches of the rate formula divide in a different order, so the rate immediately above the kink can be one unit lower than immediately below it.

The effect on interest paid is negligible, but it contradicts the documented monotonicity of the model, which integrators may rely on when computing rate bounds.

PRECONDITIONS

- Utilisation exactly at the kink boundary

EXPLOIT SCENARIO

1. Utilisation moves across the kink; the rate drops by one unit instead of rising.

IMPACT

Negligible economic effect. Recorded because it contradicts the documented model.

RECOMMENDATION

Compute both branches with the same operation order, or document the tolerance in the specification.

RESOLUTION

Acknowledged. The tolerance is documented in the specification and P-07 is stated with it.

REFERENCES

- Solidity: integer division semantics
<https://docs.soliditylang.org/en/latest/types.html>

FL-05 ■ Informational Acknowledged

Verification relies on fresh prices and sequencer availability

spec/assumptions.md

Category	CWE	Likelihood	Impact
Configuration	—	Low	Medium

DESCRIPTION

Solvency properties are proved under assumption P-12: oracle prices are positive, updated within the heartbeat, and not consumed while the Arbitrum sequencer is down or inside the grace period after it restarts. MarketCore does not enforce these conditions itself; it relies on OracleRouter, which was out of scope.

During sequencer downtime, prices on L2 can lag L1 markets significantly, and liquidations or borrows executed immediately after restart may use stale values.

PRECONDITIONS

- OracleRouter misconfigured or bypassed

— Sequencer outage

EXPLOIT SCENARIO

1. Sequencer goes down while collateral prices fall on L1.
2. On restart, borrows execute against stale, higher prices before feeds update.

IMPACT

If the assumption fails in production, the solvency results do not apply.

RECOMMENDATION

Enforce staleness, positivity and sequencer-uptime checks at the market level as defence in depth, and monitor feed heartbeats and the sequencer uptime feed.

RESOLUTION

Acknowledged. An on-chain staleness check in MarketCore is planned for the next release; monitoring is in place.

REFERENCES

- Chainlink: L2 sequencer uptime feeds
<https://docs.chain.link/data-feeds/l2-sequencer-feeds>

8

Severity and risk analysis

The matrix places each finding by the likelihood of its preconditions and the impact of exploitation. Identifiers in each cell refer to the detailed findings in section 7.

Impact

	Low	Medium	High
High	1 FL-02	1 FL-01	0
Medium	1 FL-05	1 FL-03	0
Low	1 FL-04	0	0
	Low	Medium	High

Likelihood

Distribution by vulnerability class

Business logic	2
Access control	1
Configuration	1
Precision and rounding	1

Severity distribution



9

Remediation status

3 of 5 findings are resolved, 2 acknowledged, 0 partially resolved and 0 open, as reviewed at remediation commit 4e9804ddfe0e.

Severity	Resolved	Partially resolved	Acknowledged	Open	Total
Critical	0	0	0	0	0
High	1	0	0	0	1

Severity	Resolved	Partially resolved	Acknowledged	Open	Total
Medium	2	0	0	0	2
Low	0	0	1	0	1
Informational	0	0	1	0	1

Remediation progress

100%

Remediation history



Residual items

ID	Status	Notes
FL-04	Acknowledged	Acknowledged. The tolerance is documented in the specification and P-07 is stated with it.
FL-05	Acknowledged	Acknowledged. An on-chain staleness check in MarketCore is planned for the next release; monitoring is in place.

10

Formal verification and invariants

Twelve properties were specified. Results below are final, on the remediation commit. Properties violated on the original commit are marked in their notes with the finding they produced.

Tool: SMT-based prover with bounded model checking fallback

ID	Property	Result	Notes
P-01	cash + totalBorrows – totalReserves = ? supplierAssets	Proved	–
P-02	¬badDebtEvent => exchangeRate? >= exchangeRate	Proved	Violated on the original commit (FL-02); re-proved after the fix.
P-03	borrowIndex? >= borrowIndex	Proved	–
P-04	health(a) >= 1 => liquidate(a) reverts	Proved	–
P-05	debt(a) > 0 ? collateral(a) = 0 => isBadDebt(a)	Proved	Violated on the original commit (FL-01); re-proved after the fix.
P-06	transferIn(repay) <= debt(a)	Proved	Violated on the original commit (FL-03); re-proved after the fix.

ID	Property	Result	Notes
P-07	$u1 \leq u2 \Rightarrow \text{rate}(u1) \leq \text{rate}(u2)$, within one unit at the kink	■ Proved	Proved with a documented one-unit tolerance (FL-04).
P-08	$\text{withdrawReserves} \Rightarrow \text{msg.sender} = \text{reserveAdmin}$	■ Proved	—
P-09	$\text{withdraw}(a, x) \Rightarrow x \leq \text{redeemable}(a)$	■ Proved	—
P-10	$\text{accrue}(t0 \rightarrow t2) ? \text{accrue}(t0 \rightarrow t1 \rightarrow t2)$ within 1 wei per step	■ Inconclusive	Holds for sequences up to 8 accruals; the prover timed out for unbounded sequences.
P-11	$\text{paused} \Rightarrow \text{supply, borrow and withdraw revert}$	■ Proved	—
P-12	$\text{price} > 0 ? \text{now} - \text{updatedAt} \leq \text{heartbeat} ? \text{sequencerUp}$	■ Not in scope	Assumption about OracleRouter, not proved (FL-05).

11

Recommendations

Run the specification in CI

The rule files were handed over with the report. Re-running them on every change to MarketCore or LiquidationEngine keeps the proofs valid as the code evolves.

Close the P-10 gap

Path-independence of accrual was proved only for bounded sequences. Either restructure accrual to make it provable, or add a fuzzing campaign targeting long accrual chains.

Verify OracleRouter next

Solvency depends on P-12. Extending verification to OracleRouter turns the largest assumption into a result.

Document the reserve admin's power

Even with FL-02 fixed, the reserve admin controls protocol revenue. Publish the role, its holder and its timelock.

12

Limitations

- Results are proofs about a model of the contracts listed in scope, under the stated assumptions. They are not proofs about the deployed system as a whole.
- Properties not written in the specification were not verified. The absence of a property is not evidence of safety.
- P-10 was verified only up to bounded sequences of accruals and is reported as inconclusive for unbounded executions.
- OracleRouter, governance contracts and off-chain liquidation bots were out of scope.

No assessment can show that a system is free of vulnerabilities. Conclusions apply to the code at the commits listed and the conditions stated in this document.

Appendix

Technical appendix

A.1 Tools

Tool	Purpose
SMT-based prover	Unbounded verification of the 12 properties
Bounded model checker	P-10 within bounded accrual sequences
Specification mutation testing	Confirming each rule can fail
Foundry	Reproducing counterexamples as concrete tests

A.2 Reviewed files (SHA-256)

File	SHA-256
src/MarketCore.sol	c54cb07c0aba4a3ea6ba67b9633c0fd63a7724c3340b947b36acfa66197504e6
src/InterestRateModel.sol	7f898e092957f93aef5ab1f1affe65c1543d40187ff7e52cf0f69f6ec595384d
src/LiquidationEngine.sol	f4751a07029d64694957c9b48d8a575d598b14d725ec7017e58a16b90e40c60e
src/ReserveManager.sol	a70d4065728b0143fd035ead5e47bf2da9e15cde6d01b63fa28bebe6fe254352

A.3 Notes

- Every counterexample was replayed as a Foundry test that fails on the original commit and passes on the remediation commit.
- The specification uses the notation: $x?$ for post-state, $\sum$ for summation over all accounts.

A.4 Methodology and technical references

Standards, specifications and weakness classifications cited in the findings. Each entry links to the official source.

- CWE-682: Incorrect Calculation
<https://cwe.mitre.org/data/definitions/682.html>
- CWE-284: Improper Access Control
<https://cwe.mitre.org/data/definitions/284.html>
- Solidity: integer division semantics
<https://docs.soliditylang.org/en/latest/types.html>
- Chainlink: L2 sequencer uptime feeds
<https://docs.chain.link/data-feeds/l2-sequencer-feeds>

A.5 Document information

Document	Quantix Technical Security Assessment QX-TSA-2026-002, version 1.1
Subject	Fenwick Lending, Quantix Protocol Research
Status	Published 30 September 2026
Disclosure	This document is a Quantix research assessment of Fenwick Lending, a protocol design from Quantix Protocol Research. It is not a client engagement, it does not describe a deployed system, and it is not a certification.