



TECHNICAL SECURITY ASSESSMENT

Quillon Vaults

Smart Contract Audit · Yield

REPORT ID QX-TSA-2026-001

QUANTIX RESEARCH ASSESSMENT

Vault core, strategy manager, LP strategy, withdrawal queue and timelock. 1,840 nSLOC of Solidity.

Document status Published	Published 18 August 2026
Review period 27 July 2026 – 7 August 2026	Ecosystems Ethereum
Languages Solidity	Review effort Two reviewers, ten working days, plus a three-day remediation review
Reviewed commit 4be1f0c9d2a7e35b18c64f0e9a2d7713c5b80e41	Remediation commit a91c7e2d04b36f58e1d9c2b740f6a3e85d1c09bb

FINDINGS BY SEVERITY

10 findings · 95% remediated

Critical 1	High 2	Medium 3	Low 2	Informational 2
---------------	-----------	-------------	----------	--------------------

This document is a Quantix research assessment of Quillon Vaults, 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-2026-001
Title	Technical Security Assessment: Quillon Vaults
Version	1.1
Status	Published
Assessment type	Smart Contract Audit
Review period	27 July 2026 – 7 August 2026
Published	18 August 2026
Reviewed commit	4be1f0c9d2a7e35b18c64f0e9a2d7713c5b80e41

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
	QV-01 First depositor can inflate share price and capture subsequent deposits	8
	QV-02 Strategy valuation uses manipulable AMM spot price	9
	QV-03 Fee recipient can be changed by any caller	10
	QV-04 Withdrawal queue updates state after external token transfer	11
	QV-05 previewWithdraw rounds in the user's favour	12
	QV-06 Implementation contract can be initialised by anyone	13
	QV-07 Upgradeable base contracts lack storage gaps	14
	QV-08 Timelock minimum delay can be set to zero	15
	QV-09 Parameter changes do not emit events	16
	QV-10 Floating pragma and unused imports	16
8	Severity and risk analysis	18
9	Remediation status	19
10	Formal verification and invariants	19
11	Recommendations	20
12	Limitations	21
A	Technical appendix	22

1

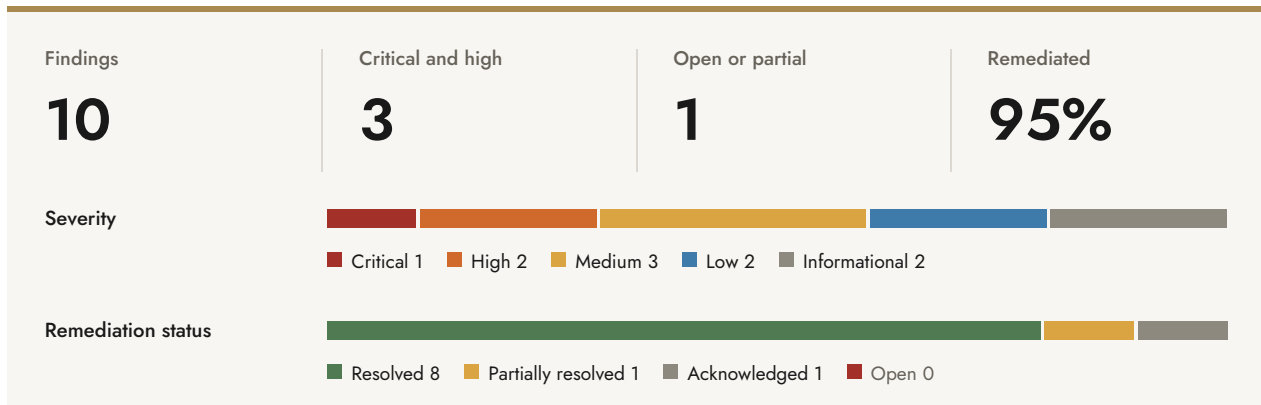
Executive summary

Quillon Vaults lets depositors pool an ERC-20 asset into a vault that allocates capital across yield strategies. The review focused on share accounting, strategy valuation and the privileged paths that can move funds or change code.

Ten findings were reported: one critical, two high, three medium, two low and two informational. The most significant allowed the first depositor to inflate the share price and capture subsequent deposits; another let an attacker inflate strategy valuation within a single transaction using the pool spot price.

All critical, high and medium findings were resolved in the remediation commit and verified, including new invariant tests for all four ERC-4626 conversion paths. One low finding was acknowledged with a documented plan and one was partially resolved. Residual risk is concentrated in strategies added after this review, which were out of scope.

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.	1	0
■ High	Material loss or protocol disruption is achievable under realistic conditions.	2	0
■ Medium	Limited loss, degraded functionality, or impact that depends on specific conditions.	3	0
■ Low	Minor impact, defence-in-depth gaps, or issues with unlikely preconditions.	2	1
■ Informational	Code quality, documentation or best-practice observations without direct security impact.	2	0

2

Scope

Vault core, strategy manager, LP strategy, withdrawal queue and timelock. 1,840 nSLOC of Solidity.

Repository	github.com/example-org/quillon-vaults
Reviewed commit	4be1f0c9d2a7e35b18c64f0e9a2d7713c5b80e41
Remediation commit	a91c7e2d04b36f58e1d9c2b740f6a3e85d1c09bb
Review period	27 July 2026 – 7 August 2026
Review effort	Two reviewers, ten working days, plus a three-day remediation review

Path	Description	nSLOC
src/QuillonVault.sol	ERC-4626 vault: deposits, withdrawals and share accounting	412
src/StrategyManager.sol	Strategy allocation, harvest and fee routing	386
src/strategies/AmmLpStrategy.sol	Liquidity-provision strategy and position valuation	344
src/WithdrawalQueue.sol	Queued withdrawals for illiquid strategies	268
src/base/VaultBase.sol	Shared upgradeable base: roles, pausing, storage	182
src/governance/QuillonTimelock.sol	Timelocked parameter changes and upgrades	248
Total		1,840

Out of scope

- Strategies other than AmmLpStrategy, including those added after the review commit
- The external AMM and lending protocols the strategies integrate with
- Off-chain keepers that trigger harvests and rebalances
- Front-end application and deployment scripts

3

Protocol architecture

Users deposit the underlying asset into QuillonVault and receive shares. The vault keeps a liquidity buffer and allocates the remainder through StrategyManager to one or more strategies. Total assets reported by the vault are the buffer plus each strategy's self-reported estimate.

AmmLpStrategy provides liquidity to a constant-product pool and values its LP position from pool reserves. Harvests swap reward tokens back to the underlying asset and route a performance fee to a fee recipient.

When the buffer cannot cover a withdrawal, requests enter WithdrawalQueue and are claimed after the strategy unwinds. All parameter changes and implementation upgrades pass through QuillonTimelock; the vault is a UUPS proxy whose upgrade function is restricted to the timelock.

Component	Role	Trust level
QuillonVault (UUPS proxy)	Share accounting, deposits, withdrawals, buffer management	Trusted
StrategyManager	Capital allocation, harvests, fee routing	Trusted
AmmLpStrategy	LP position management and valuation	Semi-trusted
WithdrawalQueue	Deferred withdrawals from illiquid strategies	Trusted
QuillonTimelock	Delays upgrades and parameter changes	Trusted
External AMM pool	Liquidity venue and price source for valuation	Untrusted
Keeper	Triggers harvests and rebalances	Semi-trusted

Trust assumptions

- The timelock is controlled by a multisig whose signers act honestly; users have the timelock delay to exit before changes take effect.
- Strategies report their assets honestly within the bounds of their valuation method; a strategy is not assumed to be adversarial, but its price inputs are.
- The underlying asset is a standard ERC-20 without transfer fees or rebasing.
- Keepers may be delayed or offline but cannot move funds outside configured strategies.

4

Threat model

Assets

- Depositor principal held by the vault and strategies
- Accrued yield and performance fees
- Implementation code and upgrade authority
- Share price integrity used by integrators

Actors

Actor	Capability	Trust
Depositor	Deposits, withdraws, transfers shares	Untrusted
Flash-loan attacker	Moves pool reserves within one transaction	Untrusted
Keeper	Calls harvest and rebalance at chosen times	Semi-trusted
Governance multisig	Queues upgrades and parameter changes through the timelock	Trusted

Threats

Threat	Actor and vector	Mitigation	Findings
Share price manipulation on an empty vault	Depositor: Donation to the vault before others deposit	Virtual shares offset and zero-share revert	QV-01

Threat	Actor and vector	Mitigation	Findings
Strategy valuation inflated within one transaction	Flash-loan attacker: Moving AMM reserves read by the strategy	TWAP valuation with spot deviation guard	QV-02
Fee redirection	Depositor: Unprotected setter	Governance-only modifier	QV-03
Double claim from the withdrawal queue	Depositor: Re-entering claim through token hooks	Checks-effects-interactions and reentrancy guard	QV-04
Value leakage through rounding	Depositor: Repeated small withdrawals	Rounding in the vault's favour on all paths	QV-05
Takeover or bricking of the implementation	Depositor: Initialising the implementation contract directly	Disable initializers in the constructor	QV-06
Upgrades without user notice	Governance multisig: Timelock delay reduced to zero	Minimum delay enforced on-chain	QV-08

5

Methodology and severity classification

1. Line-by-line manual review of all in-scope contracts by two reviewers, each covering the vault and strategy independently
2. Static analysis with manual triage of every detector result
3. Invariant fuzzing of share accounting: total assets $\geq$ redeemable value, and conversion round-trips never create value
4. Economic review of strategy valuation under single-block price manipulation, with flash-loan sized reserve changes
5. Upgrade-path review: initializer handling, storage layout and timelock authorisation
6. Remediation review of every finding against the fix commit, with re-run of all proofs of concept

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
QV-01	First depositor can inflate share price and capture subsequent deposits	■ Critical	First-depositor attack	Resolved
QV-02	Strategy valuation uses manipulable AMM spot price	■ High	Oracle manipulation	Resolved
QV-03	Fee recipient can be changed by any caller	■ High	Access control	Resolved
QV-04	Withdrawal queue updates state after external token transfer	■ Medium	Reentrancy	Resolved
QV-05	previewWithdraw rounds in the user's favour	■ Medium	Precision and rounding	Resolved
QV-06	Implementation contract can be initialised by anyone	■ Medium	Unsafe proxy upgrade	Resolved
QV-07	Upgradeable base contracts lack storage gaps	■ Low	Unsafe proxy upgrade	Acknowledged
QV-08	Timelock minimum delay can be set to zero	■ Low	Governance manipulation	Partially resolved
QV-09	Parameter changes do not emit events	■ Informational	Code quality	Resolved
QV-10	Floating pragma and unused imports	■ Informational	Code quality	Resolved

7

Detailed findings

QV-01 ■ Critical Resolved

First depositor can inflate share price and capture subsequent deposits

src/QuillonVault.sol:L188-L214

Category	CWE	Likelihood	Impact
First-depositor attack	CWE-682	High	High

DESCRIPTION

Share minting uses `assets * totalSupply / totalAssets()`, rounding down, with no virtual offset. When the vault is empty, the exchange rate is defined entirely by whatever the first depositor does.

An attacker deposits 1 wei and receives one share. They then transfer a large amount of the asset directly to the vault without calling `deposit`. `totalAssets()` now includes the donation while `totalSupply` is still 1, so a single share is worth the entire balance.

Any later deposit smaller than the donation computes `assets * 1 / (donation + 1)`, which rounds down to zero shares. The deposit is accepted, the depositor receives nothing, and the assets accrue to the attacker's single share.

PRECONDITIONS

- The vault has zero or near-zero total supply (new deployment or fully drained vault)
- The attacker can front-run or precede the first legitimate deposit
- The vault does not revert on zero-share mints

EXPLOIT SCENARIO

1. Attacker observes a pending first deposit of 100,000 USDC in the mempool.
2. Attacker front-runs it: `deposit(1 wei)` mints 1 share.
3. Attacker transfers 100,001 USDC directly to the vault, making `totalAssets = 100,001 USDC + 1 wei`.
4. The victim's deposit computes $100,000e6 \times 1 / 100,001e6 = 0$ shares and succeeds.
5. Attacker redeems 1 share for ~200,001 USDC, netting the victim's deposit minus gas.

IMPACT

Total loss of any deposit made while the attacker controls the inflated share. Losses are bounded only by the size of victim deposits, and the pattern can be repeated whenever the vault returns to an empty state.

RECOMMENDATION

Introduce a virtual share and asset offset in the conversion functions (for example a decimals offset as in OpenZeppelin's ERC4626 implementation), or mint dead shares to an unrecoverable address at initialisation. In addition, revert any deposit that would mint zero shares and any mint that would require zero assets.

FIX

src/QuillonVault.sol	DIFF
1	<code>function _convertToShares(uint256 assets, Math.Rounding r) internal view returns (uint256) {</code>
2	<code>- uint256 supply = totalSupply();</code>
3	<code>- return supply == 0 ? assets : assets.mulDiv(supply, totalAssets(), r);</code>
4	<code>+ return assets.mulDiv(totalSupply() + 10 ** _decimalsOffset(), totalAssets() + 1, r);</code>
5	<code>}</code>
6	
7	<code>function deposit(uint256 assets, address receiver) public override returns (uint256 shares) {</code>

```
8     shares = previewDeposit/assets);
9 +   if (shares == 0) revert ZeroShares();
10    _deposit(msg.sender, receiver, assets, shares);
11 }
```

RESOLUTION

Fixed with a decimals offset of 6 and a zero-share revert on deposit and mint. Verified in the remediation commit; INV-03 added to the invariant suite.

REFERENCES

- EIP-4626: Tokenized Vaults
<https://eips.ethereum.org/EIPS/eip-4626>
- OpenZeppelin ERC-4626: inflation attack and virtual offset
<https://docs.openzeppelin.com/contracts/5.x/erc4626>
- CWE-682: Incorrect Calculation
<https://cwe.mitre.org/data/definitions/682.html>

QV-02 ■ High Resolved

Strategy valuation uses manipulable AMM spot price

src/strategies/AmmLpStrategy.sol:L97-L131

Category	CWE	Likelihood	Impact
Oracle manipulation	CWE-807	Medium	High

DESCRIPTION

`estimatedTotalAssets()` values the LP position by reading the pool's current reserves and computing the position's share of each side at the spot ratio. The vault includes this estimate in `totalAssets()`, which drives both minting and redemption.

Reserves can be moved arbitrarily within one transaction using borrowed capital. Because the valuation reads reserves at the moment of the call, an attacker can skew the ratio, interact with the vault at the distorted price, and restore the pool before the transaction ends.

The LP token itself is not priced naively (the strategy decomposes it into underlying amounts), but both amounts are derived from the same manipulable reserves, so the decomposition does not help.

PRECONDITIONS

- Flash-loan liquidity in the pooled assets is available
- The vault allows deposit and redeem in the same transaction
- The strategy holds a material share of vault assets

EXPLOIT SCENARIO

1. Attacker flash-borrows a large amount of token B.
2. Attacker swaps B for A in the pool, skewing reserves so that the strategy's LP position is valued above its fair value.
3. Attacker redeems shares from the vault at the inflated `totalAssets()`, receiving more underlying than their shares are worth.
4. Attacker swaps back to restore the pool and repays the flash loan.
5. The loss is borne by remaining depositors through a lower share price.

IMPACT

Value extraction from depositors in a single transaction, repeatable while the strategy is active. Profitability depends on pool depth and the strategy's share of vault assets; at the reviewed configuration the strategy held about 60% of assets.

RECOMMENDATION

Value positions with a manipulation-resistant reference price: a time-weighted average price over a window long enough that sustaining a distortion is uneconomic, or an external oracle with deviation checks. Additionally compare spot to the reference and revert deposits and withdrawals when the deviation exceeds a bound.

FIX

src/strategies/AmmlpStrategy.sol	DIFF
1	function estimatedTotalAssets() public view returns (uint256) {
2	- (uint112 r0, uint112 r1,) = pool.getReserves();
3	- return _lpValue(uint256(r1) * 1e18 / r0) + idle();
4	+ uint256 twap = oracle.consult(address(asset), TWAP_WINDOW); // 30 minutes
5	+ uint256 spot = _spotPrice();
6	+ if (_deviationBps(spot, twap) > MAX_DEVIATION_BPS) revert PriceDeviation(spot, twap);
7	+ return _lpValue(twap) + idle();
8	}

RESOLUTION

Valuation now uses a 30-minute TWAP with a 2% spot deviation guard applied to deposits and withdrawals. INV-02 holds under fuzzing with flash-loan sized reserve changes.

REFERENCES

- Uniswap v2 oracles: time-weighted average prices
<https://developers.uniswap.org/docs/protocols/v2/concepts/oracles>
- CWE-807: Reliance on Untrusted Inputs in a Security Decision
<https://cwe.mitre.org/data/definitions/807.html>

QV-03 ■ High Resolved

Fee recipient can be changed by any caller

src/StrategyManager.sol:L64

Category	CWE	Likelihood	Impact
Access control	CWE-862	High	Medium

DESCRIPTION

`setFeeRecipient(address)` is declared `external` without the `onlyGovernance` modifier applied to every other setter in the contract. The function writes the address that receives performance and management fees on each harvest.

The omission appears to have been introduced when the function was moved from the vault into StrategyManager: the vault version was restricted, the moved version was not. No test covered calling the setter from an unprivileged account.

PRECONDITIONS

- None beyond knowing the function exists

EXPLOIT SCENARIO

1. Attacker calls `setFeeRecipient(attacker)` from any account.

- The next keeper harvest transfers the performance fee to the attacker.
- Fees continue to flow to the attacker until governance notices and resets the recipient through the timelock.

IMPACT

Loss of all protocol fees from the moment of the change until it is reverted. Depositor principal is not directly affected, but the timelock delay means the protocol cannot react quickly.

RECOMMENDATION

Restrict the function to the timelocked governance role, emit an event on change, and add a negative test for every privileged function.

FIX

src/StrategyManager.sol	DIFF
1 - function setFeeRecipient(address recipient) external {	
2 + function setFeeRecipient(address recipient) external onlyGovernance {	
3 + if (recipient == address(0)) revert ZeroAddress();	
4 + emit FeeRecipientUpdated(feeRecipient, recipient);	
5 feeRecipient = recipient;	
6 }	

RESOLUTION

Modifier added; FeeRecipientUpdated emitted; negative tests added for all governance setters.

REFERENCES

- CWE-862: Missing Authorization
<https://cwe.mitre.org/data/definitions/862.html>
- Solidity security considerations
<https://docs.soliditylang.org/en/latest/security-considerations.html>

QV-04 ■ Medium Resolved

Withdrawal queue updates state after external token transfer

src/WithdrawalQueue.sol:L142-L160

Category	CWE	Likelihood	Impact
Reentrancy	CWE-841	Low	High

DESCRIPTION

`claim(requestId)` checks that the request is ready, transfers the assets to the owner, and only then sets `requests[requestId].claimed = true`. The function has no reentrancy guard.

For a standard ERC-20 the transfer does not call back into the recipient, so the ordering is harmless. If the vault is deployed with an asset that invokes hooks on transfer (ERC-777 style or tokens with receive callbacks), the recipient regains control before the claimed flag is written and can call `claim` again for the same request.

PRECONDITIONS

- The vault's underlying asset invokes a recipient hook during transfer
- The attacker owns a ready withdrawal request

EXPLOIT SCENARIO

- Attacker queues a withdrawal and waits until it is claimable.
- Attacker calls `claim(id)` from a contract that implements the token's receive hook.
- Inside the hook, the contract calls `claim(id)` again; the request is still unclaimed, so it pays out a second time.

4. The recursion repeats until the queue's balance is exhausted or gas runs out.

IMPACT

Multiple payouts for one request, draining assets reserved for other queued withdrawals. Standard ERC-20 deployments are not affected, which keeps likelihood low, but the vault is designed to support arbitrary assets.

RECOMMENDATION

Follow checks-effects-interactions: mark the request claimed before transferring, and add a reentrancy guard to claim and claimMany.

FIX

src/WithdrawalQueue.sol	DIFF
1 - function claim(uint256 id) external {	
2 + function claim(uint256 id) external nonReentrant {	
3 Request storage r = requests[id];	
4 if (r.claimed block.timestamp < r.readyAt) revert NotClaimable(id);	
5 - asset.safeTransfer(r.owner, r.assets);	
6 - r.claimed = true;	
7 + r.claimed = true;	
8 + asset.safeTransfer(r.owner, r.assets);	
9 }	

RESOLUTION

State update moved before the transfer; nonReentrant added to claim and claimMany. INV-04 holds under fuzzing with a hook-enabled test token.

REFERENCES

- SWC-107: Reentrancy
<https://swcregistry.io/docs/SWC-107/>
- CWE-841: Improper Enforcement of Behavioral Workflow
<https://cwe.mitre.org/data/definitions/841.html>

QV-05 ■ Medium Resolved

previewWithdraw rounds in the user's favour

src/QuillonVault.sol:L240

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

DESCRIPTION

EIP-4626 requires that rounding always favours the vault: `previewWithdraw` and `previewMint` must round up, `previewDeposit` and `previewRedeem` must round down. The implementation rounds down in `previewWithdraw`, so a withdrawal of a given asset amount burns fewer shares than it should.

The difference is at most one share unit per call, but it is systematic and in one direction. An account that repeatedly withdraws small amounts extracts slightly more value than its shares represent.

PRECONDITIONS

- Gas cost per withdrawal is lower than the rounding gain, which is plausible on low-fee networks or for high-value share units

EXPLOIT SCENARIO

1. Attacker holds shares and calls withdraw with an asset amount chosen so the exact share cost has a large fractional part.
2. The vault burns the rounded-down share count.
3. Repeating many times accumulates the fractional difference as extra assets.

IMPACT

Gradual value leakage from remaining depositors. Negligible per call at current gas prices, but it violates the standard integrators rely on.

RECOMMENDATION

Use `Math.mulDiv` with `Rounding.Ceil` for share calculations on withdraw and mint paths, and add invariant tests for all four conversion paths.

FIX

src/QuillonVault.sol	DIFF
1 function previewWithdraw(uint256 assets) public view returns (uint256) {	
2 - return _convertToShares(assets, Math.Rounding.Floor);	
3 + return _convertToShares(assets, Math.Rounding.Ceil);	
4 }	

RESOLUTION

Rounding direction corrected; INV-01 added for all four conversion paths.

REFERENCES

- EIP-4626: rounding requirements
<https://eips.ethereum.org/EIPS/eip-4626>

QV-06 ■ Medium Resolved

Implementation contract can be initialised by anyone

src/QuillonVault.sol:L72

Category	CWE	Likelihood	Impact
Unsafe proxy upgrade	CWE-665	Medium	Medium

DESCRIPTION

The vault implementation behind the UUPS proxy does not call `_disableInitializers()` in its constructor. The proxy was initialised correctly at deployment, but the implementation contract itself still has uninitialised storage and a callable `initialize`.

An attacker can call `initialize` on the implementation address and become its owner and upgrader. In UUPS, the upgrade logic lives in the implementation, so the attacker can then call `upgradeToAndCall` on the implementation itself and execute arbitrary code in its context.

PRECONDITIONS

- The implementation is deployed without disabled initializers

EXPLOIT SCENARIO

1. Attacker reads the implementation address from the EIP-1967 slot of the proxy.
2. Attacker calls `initialize` on the implementation, setting themselves as owner.
3. Attacker calls `upgradeToAndCall` on the implementation with a contract that self-destructs or otherwise corrupts it.

IMPACT

Proxy storage is unaffected, but corrupting or destroying the implementation can make the proxy unusable until governance deploys and upgrades to a new implementation. Since EIP-6780 limits SELFDESTRUCT, the practical impact today is lower, which is why the finding is rated medium.

RECOMMENDATION

Call `_disableInitializers()` in the implementation constructor and include an initializer test in the deployment script.

FIX

src/QuillonVault.sol	DIFF
1 + <code>/// @custom:oz-upgrades-unsafe-allow constructor</code>	
2 + <code>constructor() {</code>	
3 + <code> _disableInitializers();</code>	
4 + <code>}</code>	

RESOLUTION

Constructor added; deployment script asserts the implementation reports initialised.

REFERENCES

- OpenZeppelin: writing upgradeable contracts
<https://docs.openzeppelin.com/upgrades-plugins/writing-upgradeable>
- EIP-1822: Universal Upgradeable Proxy Standard
<https://eips.ethereum.org/EIPS/eip-1822>

QV-07 ■ Low Acknowledged

Upgradeable base contracts lack storage gaps

src/base/VaultBase.sol

Category	CWE	Likelihood	Impact
Unsafe proxy upgrade	—	Low	Medium

DESCRIPTION

VaultBase declares state variables and is inherited by QuillonVault, which declares its own variables afterwards. VaultBase reserves no storage gap and does not use namespaced storage.

If a future upgrade adds a variable to VaultBase, every variable in QuillonVault shifts by one slot and reads values that belonged to a different variable. Storage-layout tooling would catch this only if it is run on every upgrade.

PRECONDITIONS

- A future upgrade adds state to an inherited base contract

EXPLOIT SCENARIO

1. Governance upgrades to a version that adds a variable to VaultBase.
2. Child-contract variables now read shifted storage, corrupting accounting.

IMPACT

No current impact. Raises the risk of storage collisions in future upgrades, which could corrupt accounting irrecoverably.

RECOMMENDATION

Reserve a `__gap` array in each upgradeable base, or migrate to ERC-7201 namespaced storage before the next upgrade, and run a storage-layout diff in CI.

RESOLUTION

Acknowledged. The team will migrate to ERC-7201 namespaced storage in the next major version and has added a storage-layout check to CI.

REFERENCES

- ERC-7201: Namespaced Storage Layout
<https://eips.ethereum.org/EIPS/eip-7201>

QV-08 ■ Low Partially resolved

Timelock minimum delay can be set to zero

src/governance/QuillonTimelock.sol:L51

Category	CWE	Likelihood	Impact
Governance manipulation	CWE-284	Low	Medium

DESCRIPTION

`updateDelay(uint256)` accepts any value, including zero. The function is itself timelocked, so a reduction takes one full delay to apply, but once applied every later change, including implementation upgrades, executes immediately.

The timelock is the only control that gives depositors time to exit before a harmful upgrade. Allowing it to reach zero removes that guarantee for the remainder of the protocol's life.

PRECONDITIONS

- Governance keys are compromised or act carelessly

EXPLOIT SCENARIO

- A compromised multisig queues `updateDelay(0)`; users who are not monitoring the queue do not notice.
- After the delay, the change executes; subsequent upgrades are immediate.
- The attacker upgrades the vault to an implementation that transfers all assets.

IMPACT

Removes users' ability to react to governance actions. Combined with key compromise, enables an immediate drain.

RECOMMENDATION

Enforce a minimum delay in the contract (we recommended 48 hours) and alert on any queued delay change.

FIX

src/governance/QuillonTimelock.sol	DIFF
1 function updateDelay(uint256 newDelay) external onlySelf {	
2 + if (newDelay < MIN_DELAY) revert DelayTooShort(newDelay, MIN_DELAY); // MIN_DELAY = 24 ho	
urs	
3 emit DelayUpdated(delay, newDelay);	
4 delay = newDelay;	
5 }	

RESOLUTION

A 24-hour floor was added. The team considers 24 hours sufficient for its user base; our recommendation of 48 hours stands and the residual risk is documented.

REFERENCES

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

QV-09 ■ Informational Resolved

Parameter changes do not emit events

src/StrategyManager.sol (multiple)

Category	CWE	Likelihood	Impact
Code quality	CWE-778	Low	Low

DESCRIPTION

Several setters change risk parameters (strategy debt ratios, deviation bounds, keeper address) without emitting events. Off-chain monitoring has to diff storage to notice changes.

PRECONDITIONS

- Not applicable

EXPLOIT SCENARIO

1. A parameter is changed; monitoring that relies on events does not detect it.

IMPACT

No direct impact. Reduces the effectiveness of monitoring and incident response.

RECOMMENDATION

Emit an event with old and new values for every privileged state change.

RESOLUTION

Events added for all setters.

REFERENCES

- CWE-778: Insufficient Logging
<https://cwe.mitre.org/data/definitions/778.html>

QV-10 ■ Informational Resolved

Floating pragma and unused imports

src/* (multiple)

Category	CWE	Likelihood	Impact
Code quality	CWE-664	Low	Low

DESCRIPTION

Contracts declare `pragma solidity ^0.8.20` and import libraries that are no longer used. The tested compiler version may differ from the deployed one.

PRECONDITIONS

- Not applicable

EXPLOIT SCENARIO

1. A newer compiler with different optimiser behaviour is used at deployment than in testing.

IMPACT

No direct impact. Compiler drift between testing and deployment.

RECOMMENDATION

Pin the compiler version and remove unused imports.

RESOLUTION

Pinned to 0.8.24; unused imports removed.

REFERENCES

- SWC-103: Floating Pragma
<https://swcregistry.io/docs/SWC-103/>

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 QV-04	1 QV-02	1 QV-01
Medium	2 QV-07 QV-08	2 QV-05 QV-06	1 QV-03
Low	2 QV-09 QV-10	0	0
	Low	Medium	High

Likelihood

Distribution by vulnerability class

Code quality	2
Unsafe proxy upgrade	2
Access control	1
First-depositor attack	1
Governance manipulation	1
Oracle manipulation	1
Precision and rounding	1
Reentrancy	1

Severity distribution



9

Remediation status

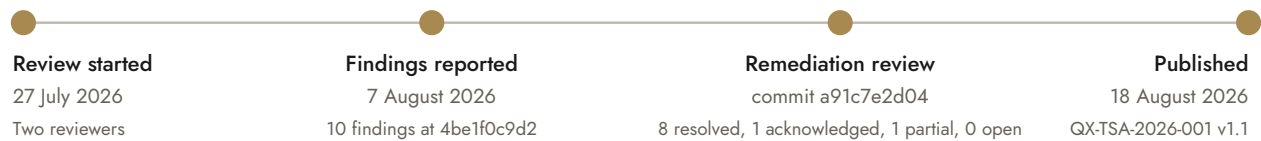
8 of 10 findings are resolved, 1 acknowledged, 1 partially resolved and 0 open, as reviewed at remediation commit a91c7e2d04b3.

Severity	Resolved	Partially resolved	Acknowledged	Open	Total
■ Critical	1	0	0	0	1
■ High	2	0	0	0	2
■ Medium	3	0	0	0	3
■ Low	0	1	1	0	2
■ Informational	2	0	0	0	2

Remediation progress

95%

Remediation history



Residual items

ID	Status	Notes
QV-07	Acknowledged	Acknowledged. The team will migrate to ERC-7201 namespaced storage in the next major version and has added a storage-layout check to CI.
QV-08	Partially resolved	A 24-hour floor was added. The team considers 24 hours sufficient for its user base; our recommendation of 48 hours stands and the residual risk is documented.

10

Formal verification and invariants

Formal verification was not part of this engagement. Four accounting properties were instead encoded as invariants and checked by fuzzing for 50 million calls against the remediation commit. They are listed for traceability; fuzzing results are evidence, not proofs.

Tool	Invariant fuzzing (Foundry)
------	-----------------------------

ID	Property	Result	Notes
INV-01	<code>convertToAssets(convertToShares(x)) <= x</code> for all <code>x</code>	■ Inconclusive	No violation in 50M calls after the QV-05 fix; not formally proved.
INV-02	<code>totalAssets()</code> $\geq$ sum of assets redeemable by all holders	■ Inconclusive	Held under fuzzing with TWAP valuation; violated on the original commit (QV-02).
INV-03	A deposit that mints zero shares reverts	■ Inconclusive	Introduced with the QV-01 fix; held under fuzzing.
INV-04	Each withdrawal request is paid at most once	■ Inconclusive	Violated on the original commit with hook-enabled tokens (QV-04); held after the fix.

11

Recommendations

Assess every new strategy before activation

Strategies define most of the vault's risk. Treat each new strategy as a separate review scope with its own valuation analysis, and require a timelocked activation so depositors can exit.

Keep the invariant suite in CI

The four invariants written during this engagement caught three of the findings. Run them on every pull request and extend them when new conversion paths or strategies are added.

Monitor price deviation and timelock queues

Alert when the TWAP deviation guard trips repeatedly, when any delay change is queued, and when an upgrade is queued. These are the earliest signals of the attack paths documented here.

Raise the timelock floor

Increase the minimum delay to 48 hours as originally recommended, and publish the delay so integrators can plan around it.

Restrict supported assets

Document that the vault supports standard ERC-20 tokens only, or keep the reentrancy guard on every path that transfers assets.

12

Limitations

- Only the files and commit listed in scope were reviewed. Strategies added later are not covered.
- External protocols integrated by the strategy were treated as behaving as documented.
- The review does not cover off-chain keepers, front ends or deployment infrastructure.
- Economic analysis assumed pool liquidity at the time of review; materially shallower pools change the cost of manipulation.

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
Foundry	Unit, fork and invariant tests; proofs of concept
Static analyser with custom detectors	Detector sweep with manual triage
Storage layout diff	Upgrade-safety comparison between commits
Manual review	Primary method for all findings

A.2 Reviewed files (SHA-256)

File	SHA-256
src/QuillonVault.sol	055aef8bbd18134315fadc2ccbf94f2aff3ac33394c8320391f58782975680b
src/StrategyManager.sol	9c95775339b455502cde3e94750c250c0a9d2c938477c3751c88e23e64656f5d
src/strategies/AmmLpStrategy.sol	42340ac94bf877813bea4e163ed9c73c67c6d9b36ef6559ae19e0f1e718dd47d
src/WithdrawalQueue.sol	71b171b444951d8b58778af20211d42ca304bde5b8faac7d404fdfa8e584913c
src/base/VaultBase.sol	acaeac0e97572e9efbf349faba3e53e4889d54ee7328f73015f8130a23cb22d7
src/governance/QuillonTimelock.sol	e840bba7545f1f1f3bcafec369d8cc001ccbbd900afcaee6b6d4b2d980c72aa9

A.3 Notes

- Hashes are SHA-256 digests of the reviewed files at the review commit.
- Severity follows the Quantix impact × likelihood matrix published in the methodology.

A.4 Methodology and technical references

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

1. EIP-4626: Tokenized Vaults
<https://eips.ethereum.org/EIPS/eip-4626>
2. OpenZeppelin ERC-4626: inflation attack and virtual offset
<https://docs.openzeppelin.com/contracts/5.x/erc4626>
3. CWE-682: Incorrect Calculation
<https://cwe.mitre.org/data/definitions/682.html>
4. Uniswap v2 oracles: time-weighted average prices
<https://developers.uniswap.org/docs/protocols/v2/concepts/oracles>
5. CWE-807: Reliance on Untrusted Inputs in a Security Decision
<https://cwe.mitre.org/data/definitions/807.html>
6. CWE-862: Missing Authorization
<https://cwe.mitre.org/data/definitions/862.html>
7. Solidity security considerations
<https://docs.soliditylang.org/en/latest/security-considerations.html>

8. SWC-107: Reentrancy
<https://swcregistry.io/docs/SWC-107/>
9. CWE-841: Improper Enforcement of Behavioral Workflow
<https://cwe.mitre.org/data/definitions/841.html>
10. OpenZeppelin: writing upgradeable contracts
<https://docs.openzeppelin.com/upgrades-plugins/writing-upgradeable>
11. EIP-1822: Universal Upgradeable Proxy Standard
<https://eips.ethereum.org/EIPS/eip-1822>
12. ERC-7201: Namespaced Storage Layout
<https://eips.ethereum.org/EIPS/eip-7201>
13. CWE-284: Improper Access Control
<https://cwe.mitre.org/data/definitions/284.html>
14. CWE-778: Insufficient Logging
<https://cwe.mitre.org/data/definitions/778.html>
15. SWC-103: Floating Pragma
<https://swcregistry.io/docs/SWC-103/>

A.5 Document information

Document	Quantix Technical Security Assessment QX-TSA-2026-001, version 1.1
Subject	Quillon Vaults, Quantix Protocol Research
Status	Published 18 August 2026
Disclosure	This document is a Quantix research assessment of Quillon Vaults, 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.