PUBLIC SAMPLE — REDACTED · client identity removed; ▮▮ bars mark removed identifiers · methodology & findings intact

ARI

PENETRATION TEST & EXPLOIT ANALYSIS

PENETRATION TEST — [redacted]

[redacted] · Track T1 — [redacted] / BoringVault RWA vault suite

PROJECT	TRACK	DATE
[redacted]	T1	11 April 2026

EXPLOITS	REFUTED	CLASSIFICATION
6 weaponized	6 defended	PUBLIC SAMPLE

Disclaimer. This report describes an authorized, time-boxed penetration-testing / exploit-analysis pass over the materials identified in scope, as they existed at the referenced version during the engagement window. Attack narratives and proof-of-concept sketches are illustrative and were reasoned/derived from source review; no exploit was executed against any live system, and no email, transaction, or request described here was actually sent. It is a point-in-time assessment and does not constitute a guarantee that all exploitable conditions have been found, nor an endorsement of the project. It is not investment, legal, or financial advice. [redacted] may publish or share this report in full, unmodified form, including this disclaimer. Vari accepts no liability for decisions made by third parties on the basis of this report.

00 ENGAGEMENT SUMMARY

The AtomicQueue / AtomicSolverV5 drain surface and the Accountant rate-bound / rate-provider surface are genuinely hardened — no drain was reachable within the shipped role config. Exploitable posture concentrates in three areas the review flagged, now confirmed against the deployment-config: (1) the cross-chain compliance boundary is broken by construction — depositAndBridge / bridge are PUBLIC with no checkAccess and all three receive handlers mint via vault.enter with no KYC / cap / share-lock, so on the MANUAL_WHITELIST vaults ([redacted], [redacted]) a non-whitelisted/sanctioned party can acquire shares and, since AtomicQueue enforces no per-user compliance, redeem them via the keeper; (2) the administered exchange rate is a monotonic ratchet (lower bound 10000 on [redacted]/[redacted]/mainnet) decoupled from backing that is lent out, and on [redacted] strategist == rate-bot == pauser == one EOA — a first-redeemer / insider drain with irreversible, socialized losses; (3) there is still no permissionless emergency exit. Flash-loan manageRoot scoping (PROTO-T1-04) is latent only — unreachable in the current deployment.

CRITICAL	HIGH	MEDIUM	LOW	INFO
0	4	2	0	0

This pass is adversarial follow-up to the security review: each item either **weaponizes** a review finding into a concrete attack with a proof-of-concept sketch, or is a **new** chain discovered while attempting exploitation. The §Refuted section lists attacks that were attempted and defeated by an in-place guard — recorded so the client can see what was tried and why it is safe.

01 WEAPONIZED EXPLOITS

ID	SEVERITY	WEAPONIZES	EXPLOIT
PEN-T1-01	HIGH	PROTO-T1-02 / PROTO-T1-08	Cross-chain receive mints shares to any address — full KYC bypass on [redacted] & [redacted] (entry), and AtomicQueue has no per-user compliance check (exit)
PEN-T1-02	HIGH	PROTO-T1-03 / PROTO-T1-01	Administered ratchet rate + keeper-ordered exits = insider / first-redeemer drain of remaining liquidity, losses socialized (no writedown possible)
PEN-T1-03	HIGH	PROTO-T1-01 / PROTO-T1-03	No permissionless emergency exit — a down/censoring keeper freezes funds indefinitely; pause does not stop interest liability accruing
NEW-T1-02	HIGH	new (key-mgmt)	Role concentration on hot EOAs — a single strategist/rate-bot/pauser key can ratchet the rate up, borrow out backing, and self-solve; operator can mint SOLVER_ROLE
PEN-T1-04	MEDIUM	PROTO-T1-02	depositCap not enforced on the bridge receive path — bridging inflates destination supply past the cap and can DoS local deposits
NEW-T1-01	MEDIUM	new	depositAndBridge (shareLockPeriod hardcoded 0) defeats refundDeposit clawback and the MEV share-lock

HIGH PEN-T1-01

Cross-chain receive mints shares to any address — full KYC bypass on [redacted] & [redacted] (entry), and AtomicQueue has no per-user compliance check (exit)

WEAPONIZES

`PROTO-T1-02 / PROTO-T1-08`

PRECONDITIONS

Target = vault with accessControlMode != DISABLED. Confirmed live: [redacted] and [redacted] are mode 2 (MANUAL_WHITELIST). depositAndBridge/bridge wired as PUBLIC capabilities with no checkAccess. [redacted] & eth-mainnet are mode 0 (KYC disabled) so are not compliance targets.

ATTACK PATH

1. Attacker A is not on the [redacted]/[redacted] whitelist (or is sanctioned).
2. On the sibling chain (Base) A calls depositAndBridge(asset, amount, minMint, BridgeData{dest=ETH, receiver=A}). depositAndBridge is public and has NO checkAccess — the whitelist is never consulted; _erc20Deposit mints shares then bridge() burns and emits the message.
3. On Ethereum the relayer delivers to _lzReceive / handle / receiveBridgeMessage, each calling vault.enter(..., receiver=A, shares) with NO checkAccess, NO depositCap, NO shareUnlockTime. A now holds freely-transferable ETH-vault shares.
4. A submits AtomicQueue.updateAtomicRequest(offer=share, want=USDC, deadline). A keeper (SOLVER_ROLE) runs redeemSolve → queue.solve → teller.bulkWithdraw(_to=solver). bulkWithdraw only checks checkAccess(solver) — the end-user A is never access-checked in the queue/solve/bulkWithdraw path.
5. A receives USDC — full whitelist bypass on entry and exit.

IMPACT

Complete defeat of the compliance gate for [redacted] and [redacted]: a sanctioned or non-KYC'd party can acquire and redeem vault shares. Regulatory/AML exposure for a gated [redacted] product. Also lets a whitelisted holder off-board shares to an unlisted address via the public bridge().

PROOF-OF-CONCEPT SKETCH (ILLUSTRATIVE — NOT EXECUTED)

```
// Base:
teller.depositAndBridge{value:fee}(USDC, 1_000_000e6, 0, BridgeData({chainSelector:ETH_EID,
destinationChainReceiver:attacker, ...}));
// ETH (relayer): _lzReceive -> vault.enter(0,0,attacker,shares);  // no KYC/cap/lock
// ETH: queue.updateAtomicRequest(share, USDC, deadline, shares);
// keeper: solver.redeemSolve(queue, share, USDC, [attacker], minOut, maxAssets, teller);
//   -> teller.bulkWithdraw(USDC, shares, address(solver));  // only solver access-checked
```

DETECTION & MITIGATION

Add checkAccess(receiver) in every receive handler and checkAccess(destinationChainReceiver) in _beforeBridge; enforce per-user compliance in AtomicQueue._finalizeSolve (already a deferred follow-up). Monitor Enter events with from==0 whose receiver is not whitelisted.

Confidence: high

HIGH PEN-T1-02 **Administered ratchet rate + keeper-ordered exits = insider / first-redeemer drain of remaining liquidity, losses socialized (no writedown possible)**

WEAPONIZES

`PROTO-T1-03 / PROTO-T1-01`

PRECONDITIONS

[redacted]/[redacted]/mainnet set allowedExchangeRateChangeLower = 10000 (rate can never be lowered — a monotonic ratchet). Backing is lent out to the borrower via the [redacted] merkle leaves. Redemptions serviced only by SOLVER_ROLE. On [redacted] strategist == exchangeRateBot == pauser == one EOA ([redacted]), which also calls the solver.

ATTACK PATH

1. Vault has public LPs; base assets are lent out, so on-chain backing << share liabilities valued at ratchet rate R.
2. A credit event (borrower default/loss) occurs. Because the lower bound is 10000, updateExchangeRate cannot write the price down and there is no other writedown mechanism — R stays put (and climbs if lendingRate>0).
3. Whoever redeems first extracts full value at R from the limited liquidity still in the vault (bulkWithdraw → vault.exit). Later redeemers' exit reverts on insufficient balance — stuck holding shares priced at R against an empty vault.
4. On [redacted] the insider controls exit ordering directly (strategist==rate-bot==the solver caller), so the insider or a colluding LP is solved first, ahead of the public queue; claimFees can additionally pull base out without lowering R.

IMPACT

First-mover / insider LP redeems at par during insolvency; honest LPs absorb the entire shortfall. Because exits are keeper-gated and ordering is keeper-chosen, the key-holder chooses who gets paid — an economic loss becomes a discretionary self-preferential payout. On [redacted] one hot key decides it.

PROOF-OF-CONCEPT SKETCH (ILLUSTRATIVE — NOT EXECUTED)

```
// after undisclosed loss, rate still R (ratchet floor blocks decrease):
queue.updateAtomicRequest(share, USDC, dl, insiderShares);
solver.redeemSolve(queue, share, USDC, [insider], minOut, maxAssets, teller); // insider
solved first
// public LPs later: solve -> bulkWithdraw -> vault.exit reverts (no USDC left)
```

DETECTION & MITIGATION

Off-chain solvency oracle comparing on-chain backing to totalSupply*rate; introduce a permissioned rate-DOWN / NAV-linked path; diversify solver/rate-bot/strategist/pauser keys onto separate multisigs; enforce pro-rata solve ordering.

Confidence: med

HIGH PEN-T1-03 **No permissionless emergency exit — a down/censoring keeper freezes funds indefinitely; pause does not stop interest liability accruing**

WEAPONIZES

`PROTO-T1-01 / PROTO-T1-03`

PRECONDITIONS

AtomicQueue.solve and teller.bulkWithdraw are SOLVER_ROLE-gated; users can only submit AtomicRequests, never self-execute. No user-callable redeem on teller or vault. refundDeposit is strategist-only and only inside the lock window.

ATTACK PATH

1. User holds shares and submits an AtomicRequest.
2. The keeper simply never solves that user — outage, key loss, sanctions screening, or deliberate censorship. No on-chain fallback forces redemption at any deadline.
3. Independently, if governance pauses the accountant, getRateSafe reverts (deposits/withdraws halt) but calculateExchangeRateWithInterest keeps accruing over the pause window; on unpause the full elapsed interest is booked — the liability grew through the freeze.

IMPACT

User funds freezable for unbounded time with no trustless exit — the core custody risk of a keeper-gated vault. Combined with PEN-T1-02 the keeper both gates exits and chooses ordering. Pausing to contain a loss does not stop share liabilities growing.

PROOF-OF-CONCEPT SKETCH (ILLUSTRATIVE — NOT EXECUTED)

```
queue.updateAtomicRequest(share, USDC, now+7d, shares);
// keeper never calls redeemSolve([user]) -> request expires, funds stuck; no
teller.redeem() fallback
```

DETECTION & MITIGATION

Add a permissionless, time-delayed emergency redeem (after deadline+grace, anyone triggers a pro-rata bulkWithdraw at the last-checkpointed rate). Freeze lendingRate accrual while the accountant is paused (checkpoint at pause, stop the clock).

Confidence: high

HIGH NEW-T1-02 **Role concentration on hot EOAs — a single strategist/rate-bot/pauser key can ratchet the rate up, borrow out backing, and self-solve; operator can mint SOLVER_ROLE**

WEAPONIZES

`new (key-mgmt)`

PRECONDITIONS

[redacted]: strategist == exchangeRateBot == pauser == [redacted] (one EOA). All vaults: operator [redacted]… holds OPERATOR_ROLE (setUserRole) + UPDATE_EXCHANGE_RATE + PAUSER. UPDATE_EXCHANGE_RATE can setLendingRate up to maxLendingRate 5000 (50%). The canonical AtomicQueue is already approved on the solver.

ATTACK PATH

1. Compromise of the single [redacted] EOA yields STRATEGIST (borrow vault base out via [redacted] leaves + call queue.solve), UPDATE_EXCHANGE_RATE (raise lendingRate / ratchet rate up +0.1%/hr), and PAUSER (freeze the world) simultaneously.
2. Attacker inflates share value via lendingRate / repeated hourly ratchet-ups; the ratchet lower bound guarantees honest governance can never correct it down afterward.
3. Attacker self-solves their own AtomicRequest through the already-approved canonical queue, extracting assets at the inflated rate, then pauses to block competing exits.
4. Separately the OPERATOR key alone can setUserRole to grant itself SOLVER_ROLE and redeemSolve through the approved queue — approvedQueues does not help because the canonical queue is legitimately approved.

IMPACT

A governance-key compromise leads to full drain, and the rate ratchet makes the damage irreversible by honest governance. The [redacted] single-EOA topology means one hot key is a complete pivot — the highest-probability real-world loss path given the deployed key layout.

PROOF-OF-CONCEPT SKETCH (ILLUSTRATIVE — NOT EXECUTED)

```
accountant.setLendingRate(5000); // or repeated updateExchangeRate +0.1%/hr
queue.updateAtomicRequest(share, USDC, dl, myShares);
solver.redeemSolve(queue, share, USDC, [me], minOut, maxAssets, teller); // approved queue
teller.pause(); accountant.pause(); // block competing exits
```

DETECTION & MITIGATION

Move UPDATE_EXCHANGE_RATE / STRATEGIST / PAUSER / OPERATOR onto separate multisigs; never collocate rate + solve + pause on one EOA; timelock rate increases; reconsider OPERATOR holding setUserRole together with solve capability.

Confidence: med

MEDIUM PEN-T1-04 **depositCap not enforced on the bridge receive path — bridging inflates destination supply past the cap and can DoS local deposits**

WEAPONIZES

`PROTO-T1-02`

PRECONDITIONS

Any deployment with a finite depositCap ([redacted] 50M, [redacted]/[redacted] 10M) and an active inbound bridge route. depositAndBridge enforces the cap on the source side only.

ATTACK PATH

1. Attacker mints/acquires shares on the high-cap/cheap source chain and bridges a large tranche to the destination (or splits across many source deposits each under the source cap).
2. Destination receive handler calls vault.enter with no cap check, pushing destination totalSupply*rate above depositCap.
3. Legitimate destination depositors now hit 'Deposit cap exceeded' and are blocked while bridged supply sits above the cap.
4. Attacker can hold the small (10M) caps saturated cheaply relative to their capital, which is recoverable by bridging back out.

IMPACT

Per-chain risk-cap invariant broken, plus a griefing DoS of local deposits on a small-cap vault.

PROOF-OF-CONCEPT SKETCH (ILLUSTRATIVE — NOT EXECUTED)

```
// repeatedly on source, each under source cap:
teller.depositAndBridge{value:fee}(USDC, capSized, 0, BridgeData{dest, receiver:attacker});
// dest receive mints with no cap check -> totalSupply*rate > depositCap; victim deposit
reverts
```

DETECTION & MITIGATION

Check the cap inside the receive handlers, or track a global cross-chain supply accountant. Alert when totalSupply*rate on any chain exceeds that chain's cap.

Confidence: high

MEDIUM NEW-T1-01

depositAndBridge (shareLockPeriod hardcoded 0) defeats refundDeposit clawback and the MEV share-lock

WEAPONIZES

`new`

PRECONDITIONS

A deployment relying on shareLockPeriod for deposit refundability / MEV protection. depositAndBridge calls _afterPublicDeposit(...,0) so shareUnlockTime is never set; the receive handler also omits it.

ATTACK PATH

1. User routes through depositAndBridge to a sibling chain with receiver=self. No shareUnlockTime is written on source (shares burned in-tx) or destination.
2. The source publicDepositHistory entry has lockPeriod 0, so refundDeposit's (now - depositTs) > 0 treats it as already unlocked one block later — a DEPOSIT_REFUNDER cannot claw it back.
3. Both the anti-sandwich share-lock and the deposit clawback are bypassed by choosing depositAndBridge over deposit().

IMPACT

Loss of the deposit-clawback control and the MEV share-lock for any user routing through the bridge entrypoint — relevant if refundDeposit is used as a compliance/error-remediation tool.

PROOF-OF-CONCEPT SKETCH (ILLUSTRATIVE — NOT EXECUTED)

```
teller.depositAndBridge{value:fee}(USDC, amt, 0, BridgeData{sibling, receiver:self});
// source: shareUnlockTime unset, history lockPeriod=0; refundDeposit(...) -> reverts
SharesAreUnLocked
```

DETECTION & MITIGATION

Set shareUnlockTime on the depositAndBridge source and every receive handler, or document bridged deposits as non-refundable and enforce compliance before mint.

Confidence: med

02 ATTEMPTED & REFUTED

Attacks tried during this pass that a guard defeated. Documenting them shows the defended surface and prevents these from being re-raised as open issues.

TARGET / IDEA	WHY IT IS SAFE
PROTO-T1-04 flash-loan continuation scoped to manageRoot[manager]	Real in code, but unreachable in the shipped config: the Manager is granted only MANAGER_ROLE (never STRATEGIST_ROLE) so the nested manage call reverts on requiresAuth, and manageRoot[manager] is never set by any script and contains no flashLoan leaf. Latent design flaw only — becomes exploitable only if governance both grants the Manager STRATEGIST_ROLE and authors a broad manageRoot[manager].
AtomicQueue griefing (no min-price / dust-zero reverts batch, PROTO-T1-05/06)	Hardened: updateAtomicRequest validates deadline/balance/allowance at submission and solve() pins solver==msg.sender. The dust zero-want case reverts only the griefer's own batch inclusion; viewSolveMetaData flags such users and the solver excludes them — self-DoS, not a lever against others.
Rogue-queue → finishSolve drain (C-1 class)	AtomicSolverV5 defends in depth: finishSolve requires in-solve context + msg.sender==expectedQueue + approvedQueues[queue]; approvedQueues is owner-only and not public, and OPERATOR lacks setRoleCapability, so a compromised OPERATOR minting QUEUE_ROLE on a rogue queue still cannot approve it. rescue() is blocked mid-solve and hard-codes recipient=owner.
Single-chain same-block mint-and-exit by a sanctioned address	Not atomic: deposit/bulkDeposit carry checkAccess; depositAndBridge mints then immediately burns and ships off-chain; every exit is SOLVER_ROLE-gated and asynchronous. The compliance bypass (PEN-T1-01) is a multi-step, keeper-serviced cross-chain flow, not a one-block mint+redeem.
setRateProviderData inflation / one-step unbounded updateExchangeRate drift	5% deviation cap on provider replacement (pegged-bypass closed), reject-and-pause without commit on out-of-bounds rate, and hard ±10% change caps. A single malicious update cannot inflate share value in one step; sustained drift needs the multi-update key-compromise path (NEW-T1-02).
Destination receive black-holing bridged shares on pause	Deliberately mitigated (N-2): inbound receive gated by isReceivePaused decoupled from deposit-side isPaused, and the LZ path removed accountant.checkpoint() so a paused accountant no longer reverts delivery.

03 REMEDIATION PRIORITY

PRIORITY	ID	FIX
P0	PEN-T1-01	Add checkAccess(receiver) in every receive handler and checkAccess(destinationChainReceiver) in _beforeBridge; enforce per-user compliance in AtomicQueue._finalizeSolve (already a deferred follow-up). Monitor Enter events with from==0 whose receiver is not whitelisted.
P0	PEN-T1-02	Off-chain solvency oracle comparing on-chain backing to totalSupply*rate; introduce a permissioned rate-DOWN / NAV-linked path; diversify solver/rate-bot/strategist/pauser keys onto separate multisigs; enforce pro-rata solve ordering.
P0	PEN-T1-03	Add a permissionless, time-delayed emergency redeem (after deadline+grace, anyone triggers a pro-rata bulkWithdraw at the last-checkpointed rate). Freeze lendingRate accrual while the accountant is paused (checkpoint at pause, stop the clock).
P0	NEW-T1-02	Move UPDATE_EXCHANGE_RATE / STRATEGIST / PAUSER / OPERATOR onto separate multisigs; never collocate rate + solve + pause on one EOA; timelock rate increases; reconsider OPERATOR holding setUserRole together with solve capability.
P1	PEN-T1-04	Check the cap inside the receive handlers, or track a global cross-chain supply accountant. Alert when totalSupply*rate on any chain exceeds that chain's cap.
P1	NEW-T1-01	Set shareUnlockTime on the depositAndBridge source and every receive handler, or document bridged deposits as non-refundable and enforce compliance before mint.