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

ARI

PENETRATION TEST & EXPLOIT ANALYSIS

PENETRATION TEST — [redacted]

[redacted] · Track T3 — Treasury-yield pool + factory

PROJECT	TRACK	DATE
[redacted]	T3	5 May 2026

EXPLOITS	REFUTED	CLASSIFICATION
5 weaponized	5 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

Two HIGH manager-privileged exploits dominate, both confirmed with exact arithmetic. The reward index is per-token and UNDILUTED by totalSupply, and payout is drawn from a SHARED, un-sub-accounted factory reserve, so an unbounded manager APR lets a manager+accomplice drain the cross-pool reserve in a single block, and a large enough rate overflows the accrual under 0.8 checked math to permanently brick a pool with a self-locking reset. Both are manager-gated, not permissionless — with the default rate 0 no attack exists, which was explicitly verified. Also confirmed: KYC only covers deposit + reward-claim, so unvetted/de-whitelisted parties can hold [redacted] and redeem principal; and a timelock-free beacon upgrade re-implements all pools atomically. No first-depositor / share-inflation or reentrancy vector exists. The single highest-leverage fix — a MAX_RATE cap on the rate setters — kills both HIGHs at once.

CRITICAL	HIGH	MEDIUM	LOW	INFO
0	2	2	1	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-T3-01	HIGH	PROTO-T3-01(a)	Malicious/compromised manager drains the SHARED factory reward reserve across pools via unbounded changeAprRate
PEN-T3-02	HIGH	PROTO-T3-01(b)	Manager permanently bricks a pool (principal frozen) via overflow-inducing APR, with a self-locking reset path
PEN-T3-03	MEDIUM	PROTO-T3-02	KYC/compliance bypass — principal exit and token transfer are ungated; withdraw() lacks onlyEligible
PEN-T3-04	MEDIUM	PROTO-T3-03	Timelock-free governor beacon upgrade of ALL pools + arbitrary totalAccumulated overwrite
PEN-T3-05	LOW	PROTO-T3-06	Rate-setter brick via asset==[redacted] index collision at pool creation

HIGH PEN-T3-01

Malicious/compromised manager drains the SHARED factory reward reserve across pools via unbounded changeAprRate

WEAPONIZES

`PROTO-T3-01(a)`

PRECONDITIONS

Attacker controls the manager of ANY pool A. Pool A is public (or uses a pre-whitelisted accomplice for the deposit leg). Factory holds an off-chain-funded [redacted] reserve that also backs other pools. No guard on rate magnitude.

ATTACK PATH

1. Accomplice X deposits D into pool A (D can be 1 [redacted]). The manager itself cannot deposit, so a second address is used.
2. Manager calls pool.changeAprRate(r) — unbounded uint256; the targeted reward asset is the pool's own asset = [redacted].
3. Wait one block (dt≥1s; [redacted] ~1.8s).
4. X calls factory.withdrawReward([redacted], [poolA]) — PERMISSIONLESS. The pool accrues rewardsIndex += dt·r (undiluted by totalSupply), computes withdrawable = D·rewardsIndex/1e18, and the factory pays it from the FACTORY's own balance.
5. Because the reserve is one shared pot with no per-pool sub-accounting, the outsized payout consumes tokens earmarked for pools B/C, whose later withdrawReward calls revert (reserve emptied) — cross-pool theft + DoS.

IMPACT

$reward_{18}$ = D·dt·r; factory pays D·dt·r/1e12 native [redacted]. To steal S [redacted]: r = S·1e18/(D·dt). With D=1, dt=2s, S=1,000,000 → r=5e23 (trivially within uint256). The bound is only the factory's actual balance, so the entire cross-pool reserve is drainable in one block with a single deposited token.

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

```
gov.createPool({      , manager:0xMgr}); // poolA public
prank(X);       .approve(poolA,1e6); poolA.deposit(1e6);
prank(0xMgr); poolA.changeAprRate(5e23); warp(+2);
prank(X); factory.withdrawReward(      ,[poolA]); // pays 1e6*2*5e23/1e12 = 1,000,000
to X
gov; factory.withdrawReward(      ,[poolB]); // legit poolB lender now reverts (reserve
empty)
```

DETECTION & MITIGATION

Enforce a MAX_RATE cap in changeAprRate/[redacted]; sub-account the reserve per pool and cap each pool's cumulative payout to its funding; fund rewards into the pool rather than a shared pot. Alert on rate values far above a sane APR (>~1e16) and on reserve drops faster than the funding schedule.

Confidence: high (manager-privileged, not permissionless)

HIGH PEN-T3-02

Manager permanently bricks a pool (principal frozen) via overflow-inducing APR, with a self-locking reset path

WEAPONIZES

`PROTO-T3-01(b)`

PRECONDITIONS

Manager of pool A; pool has any non-zero totalSupply; Solidity 0.8 checked arithmetic.

ATTACK PATH

1. Pool has a deposit → totalSupply_wei = D·1e18.
2. Manager calls changeAprRate(r_brick): this first accrues with the OLD rate (succeeds), then stores r_brick. updateTotalAccumulated computes totalSupply_wei·(dt·r) before dividing by 1e18 → overflow when that product > $2^{256}-1$ (~1.16e77). With D=1, dt=1 any r > 1.16e59 overflows; use r_brick=2^{255}.
3. From the next block, EVERY accrual path reverts: deposit, withdraw, any transfer (_update), factory.withdrawReward, and changeAprRate itself.
4. Reset deadlock: to lower the rate the setter must _accrueRewardsIndex() FIRST, which overflows on totalSupply_wei·(dt·r_brick) before the new rate is written — the rate can never be reduced. Permanent.

IMPACT

Total, irreversible DoS on pool A: all deposited principal frozen forever, transfers frozen, rewards uncomputable. No recovery short of a beacon upgrade (itself an unguarded governor power).

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

```
prank(X); poolA.deposit(1e6);
prank(0xMgr); poolA.changeAprRate(2**255); // succeeds (accrues with old rate 0)
warp(+2); prank(X); expectRevert(); poolA.withdraw(1e6); // overflow
prank(0xMgr); expectRevert(); poolA.changeAprRate(0); // reset path overflows on its own
accrue -> deadlock
```

DETECTION & MITIGATION

A MAX_RATE cap well below 1e59 makes overflow impossible and fixes this and PEN-T3-01 together; additionally make the reset path skip/saturate accrual.

Confidence: high (manager-privileged griefing)

MEDIUM PEN-T3-03

KYC/compliance bypass — principal exit and token transfer are ungated; withdraw() lacks onlyEligible

WEAPONIZES

`PROTO-T3-02`

PRECONDITIONS

A KYC pool (kycRequired=true). One whitelisted holder H acquires [redacted] legitimately; recipient N is not whitelisted.

ATTACK PATH

1. Whitelisted H deposits and receives [redacted] (1:1).
2. H calls transfer(N, amount). _update has NO eligibility guard, so [redacted] transfers to any address including non-whitelisted N.
3. N calls withdraw(amount): withdraw() has nonReentrant + nonZeroValue but NO onlyEligible — N burns [redacted] and receives underlying [redacted] principal despite never passing KYC.
4. Variant: a holder de-whitelisted via setLenderStatus(H,false) can still withdraw() and still transfer [redacted].

IMPACT

Defeats the KYC/AML control for the principal leg: unvetted/de-listed/sanctioned parties can hold and redeem pool tokens for [redacted]. Reward withdrawal IS gated (onlyEligible), so only principal leaks — but principal is the entire deposit.

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

```
gov.setLenderStatus(H,true); prank(H); pool.deposit(100e6);
prank(H); pool.transfer(N,100e6); // N never whitelisted
prank(N); pool.withdraw(100e6); assert(      .balanceOf(N)==100e6);
```

DETECTION & MITIGATION

Add onlyEligible(msg.sender) to withdraw() and eligibility checks for both from and to in _update when kycRequired; or make [redacted] non-transferable in KYC pools.

Confidence: high

MEDIUM PEN-T3-04

Timelock-free governor beacon upgrade of ALL pools + arbitrary totalAccumulated overwrite

WEAPONIZES

`PROTO-T3-03`

PRECONDITIONS

Governor (factory owner) key malicious/compromised; single-sig / no timelock in front of factory ownership.

ATTACK PATH

1. Governor calls factory.changePoolBeacon(evilImpl) — onlyOwner, only nonZero/nonSame checks, no timelock. The beacon backs every BeaconProxy pool, so all pools are re-implemented in one tx.
2. evilImpl adds a backdoor withdraw transferring all pool cash to the governor. All pool TVL captured atomically with zero warning window.
3. Independently, updateTotalAccumulated(asset, arbitrary) overwrites the reported accrual (feeds the totalRewards() view) to mask a prior reserve drain.

IMPACT

Full custodial control of all pools with no user exit window; updateTotalAccumulated lets the governor falsify reported accruals (reporting/forensics-evasion). A single compromised owner key becomes an instant, deniable rug of every pool.

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

```
gov; factory.changePoolBeacon(address(new EvilPool()));
gov; EvilPool(poolA).sweep(gov); // drains poolA cash
gov; poolA.updateTotalAccumulated(      , 0); // mask the drain
```

DETECTION & MITIGATION

48h timelock + multisig behind factory ownership; emit and delay beacon changes so users can exit; remove/derive updateTotalAccumulated; monitor beacon implementation() changes.

Confidence: high (governor trust/centralization)

LOW PEN-T3-05 **Rate-setter brick via asset==[redacted] index collision at pool creation**

WEAPONIZES

PROTO-T3-06

PRECONDITIONS

A pool is created (governor-only) whose asset equals the [redacted] reward token. Configuration error, not attacker-triggerable.

ATTACK PATH

1. __init inserts [redacted] at index 0, then asset at index 1. If asset==[redacted], the second insert sees the flag already set and just updates the index-0 rate to 0 instead of pushing — rewardAssetData length is 1.
2. changeAprRate reads rewards[1].asset → index out of bounds → revert. The manager APR setter is permanently unusable; the [redacted] rate was clobbered to 0.

IMPACT

That pool can never have a manager APR set and its [redacted] rate is silently reset. Denial of the reward feature for one misconfigured pool. (Silver lining: such a pool is immune to PEN-T3-01/02.)

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

```
createPool({asset:      }); pool.changeAprRate(1e12); // reverts out-of-bounds
```

DETECTION & MITIGATION

Validate asset != [redacted] (and against existing reward assets) in createPool/__init; look up reward assets by address, not positional index.

Confidence: high (config-time)

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
Permissionless (non-manager) reward drain	Rewards accrue only when a rate is non-zero; the pool-asset rate defaults to 0 and is settable only by the manager, the [redacted] rate only by the governor. With rate 0, withdrawReward returns 0 — a random attacker with no manager/governor cooperation cannot mint or drain anything.
Non-manager triggering the overflow brick	Setting an astronomically large rate is gated by onlyManager/onlyGovernor; external users can only make accrual RUN, and with a sane rate that arithmetic never overflows.
Reward theft by a non-whitelisted holder in a KYC pool	pool.withdrawReward carries onlyEligible(account), so a non-whitelisted address cannot claim REWARDS — the KYC gap is limited to principal + transfer.
Reentrancy on deposit/withdraw/withdrawReward	deposit/withdraw carry nonReentrant and burn precedes transfer; factory.withdrawReward records state before the trailing safeTransfer. No reentrancy lever.
First-depositor share-inflation / donation	1:1 mint/burn with no share-price ratio and a per-token reward index; a direct donation changes no user's principal or entitlement.

03 REMEDIATION PRIORITY

PRIORITY	ID	FIX
P0	PEN-T3-01	Enforce a MAX_RATE cap in changeAprRate/[redacted]; sub-account the reserve per pool and cap each pool's cumulative payout to its funding; fund rewards into the pool rather than a shared pot. Alert on rate values far above a sane APR (>~1e16) and on reserve drops faster than the funding schedule.
P0	PEN-T3-02	A MAX_RATE cap well below 1e59 makes overflow impossible and fixes this and PEN-T3-01 together; additionally make the reset path skip/saturate accrual.
P1	PEN-T3-03	Add onlyEligible(msg.sender) to withdraw() and eligibility checks for both from and to in _update when kycRequired; or make [redacted] non-transferable in KYC pools.
P1	PEN-T3-04	48h timelock + multisig behind factory ownership; emit and delay beacon changes so users can exit; remove/derive updateTotalAccumulated; monitor beacon implementation() changes.
P2	PEN-T3-05	Validate asset != [redacted] (and against existing reward assets) in createPool/__init; look up reward assets by address, not positional index.