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

ARI

PENETRATION TEST & EXPLOIT ANALYSIS

PENETRATION TEST — [redacted] PROTOCOL

[redacted] · Track T2 — Membership-gated fixed-term lending

PROJECT	TRACK	DATE
[redacted]	T2	23 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

[redacted] pools custody no funds (all peer-to-peer transfers) and are guarded by nonReentrant, so theft/drain vectors are refuted. The live risk is accounting-correctness and griefing. The monthly overdue-interest double-charge (PROTO-T2-01) is weaponized and numerically confirmed, and two NEW chains were found: a multi-lend aggregate-proration overcharge (~43% on the second position) and an origination-refund underflow that reverts repay() after a roll+callback (a self-resolving repayment DoS). PROTO-T2-02 is a hard griefing primitive — a single lender can permanently brick on-chain settlement at maturity+3d, and there is NO post-default claim function at all. All accounting divergences favor the lender/treasury (borrower overpaid); no lender-underpaid path was found. One PoolBeacon upgrades all pools per chain across 8 chains, so a single beacon upgrade can remediate every pool at once.

CRITICAL	HIGH	MEDIUM	LOW	INFO
0	0	2	4	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-T2-01	MEDIUM	PROTO-T2-01	Monthly overdue interest double-charged (pay-through-to-now vs deduct-one-month mismatch)
PEN-T2-02	MEDIUM	new (PROTO-T2-01/04 root cause)	Multi-lend + callback early-repay overcharge — aggregate totalInterest proration ignores per-position start
PEN-T2-03	LOW	new (PROTO-T2-03 + roll)	Origination-fee refund underflow reverts repay() after a roll while a callback is active (repayment DoS)
PEN-T2-04	LOW	PROTO-T2-03	Early full repay without a callback overcharges origination fee to treasury (refund gate inverted vs comment)
PEN-T2-05	LOW	PROTO-T2-02	Terminal lender-triggered default bricks all on-chain settlement (grief/extortion, no cure, no post-default claim)
PEN-T2-06	LOW	PROTO-T2-07	KYC-revoked lender's stale callback permanently locks the pool out of rolls

MEDIUM PEN-T2-01

Monthly overdue interest double-charged (pay-through-to-now vs deduct-one-month mismatch)

WEAPONIZES

`PROTO-T2-01`

PRECONDITIONS

Monthly pool (isBulletLoan=false); borrower calls repayInterest() past the next scheduled payment (lastPaidTimestamp+30d).

ATTACK PATH

1. Pool active at t0, principal P, rate R; totalInterest = P·R·tenor/YEAR, accrualTs=t0, lastPaidTimestamp=t0.
2. Borrower lets month 1 lapse and calls repayInterest() at bt = t0+30d+overdue.
3. _repayInterest sets newPaidTimestamp = nextPaymentTimestamp = t0+30d (advances only one month).
4. dueInterestOf monthly branch computes endDate = max(bt, nextPaymentTs) = bt, so interest is paid over [t0, bt] (30d+overdue paid now).
5. But totalInterest is reduced and accrualTs advanced only to t0+30d — the overdue slice [t0+30d, bt] was paid yet never deducted; it is charged again next round and at final repay().

IMPACT

Borrower overpays one extra copy of the overdue interval per late payment: excess = P·R·overdue/YEAR (gross). Verified: P=1e6, R=10%, overdue=10d → paid 19,444 vs correct 16,667 → 2,778 excess. If N months late in one call, ~(N-1) months remain double-booked. Split lender/treasury by spreadRate — a lender profits from borrower lateness on top of the penalty. No funds stuck (P2P).

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

```
whitelist(lender); lender.lend(P); warp(t0+40d); prank(borrower); pool.repayInterest();
warp(t0+70d); pool.repayInterest(); assert(paidInterest > 2*monthInterest by P*R*10d/YEAR);
pool.repayAll(); assert(final due still carries inflated totalInterest);
```

DETECTION & MITIGATION

Set accrualTs=endDate and deduct totalInterest by the same interval actually paid; or cap endDate at nextPaymentTs and let penalty (not interest) cover lateness. Reconcile cumulative RepayedInterest vs P·R·elapsed/YEAR.

Confidence: high — code trace + numeric sim

MEDIUM PEN-T2-02

Multi-lend + callback early-repay overcharge — aggregate totalInterest proration ignores per-position start

WEAPONIZES

`new (PROTO-T2-01/04 root cause)`

PRECONDITIONS

Lender lends TWICE at different timestamps (accrualTs fixed at FIRST lend; later Positions start later), holds a callback, and borrower repays before maturity.

ATTACK PATH

1. Lend p1 at t0 → accrualTs=t0, Position1 over [t0,M].
2. Lend p2 (large) at t1>t0 → principal!=0 so accrualTs stays t0; Position2 over [t1,M]; totalInterest=I1+I2.
3. requestCallBack() (before maturity).
4. Borrower repay(lender) at T (t1<T<M): due = totalInterest·(T-accrualTs)/(M-accrualTs) = (I1+I2)·(T-t0)/(M-t0).
5. Correct = I1·(T-t0)/(M-t0) + I2·(T-t1)/(M-t1). Since (T-t0)/(M-t0) > (T-t1)/(M-t1), position 2 is over-attributed early interest.

IMPACT

Borrower overcharged by I2·[(T-t0)/(M-t0) − (T-t1)/(M-t1)]. Verified p1=100k@t0, p2=900k@120d, tenor=180d, T=150d, R=10%: charged 16,667 vs correct 11,667 → 5,000 (~43%) excess. A lender maximizes it by anchoring accrualTs with a tiny early lend, a large late lend, then forcing early repayment via a callback.

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

```
lender.lend(1); warp(t0+120d); lender.lend(1e6); pool.requestCallBack();
warp(t0+150d); prank(borrower); pool.repay(lender);
assert(interestPaid == totalInterest*(150-0)/180 > perPositionSum);
```

DETECTION & MITIGATION

Prorate each Position over its own [startAt,endAt] in _dueInterestOf instead of a single aggregate accrualTs; on repay, cross-check dueOf against the per-position sum used by balanceOf.

Confidence: high — code trace + sim

LOW PEN-T2-03 **Origination-fee refund underflow reverts repay() after a roll while a callback is active (repayment DoS)**

WEAPONIZES

`new (PROTO-T2-03 + roll)`

PRECONDITIONS

originationRate>0; single-lender pool that accepted ≥1 roll (maturity extended to M2=M1+tenor); lender holds an active callback; borrower attempts repay before the ORIGINAL maturity M1.

ATTACK PATH

1. Lend at activation → totalOriginationFee = P·orig·tenor/YEAR; original maturity M1.
2. Roll accepted → maturityDate=M2=M1+tenor; totalOriginationFee NOT increased by the roll.
3. requestCallBack() (allowed, before M2).
4. repay(lender)/repayAll() at T<M1: refund branch computes unusedTime = M2−T; refund = P·orig·(M2−T)/YEAR.
5. Since M2−T > tenor, refund > totalOriginationFee → checked subtraction underflows (panic [redacted]) → whole repay reverts.

IMPACT

Borrower cannot repay (or repayAll) until block.timestamp reaches M1, after which the refund branch is skipped and repay succeeds. Self-resolving, bounded griefing/DoS window, no theft — but it chains with PEN-T2-05 (block repay pre-M1, then default at M1+3d).

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

```
activate+lend; requestRoll+acceptRoll; pool.requestCallBack(); warp(M1-2d);
expectRevert(arithmetic); prank(borrower); pool.repay(lender);
```

DETECTION & MITIGATION

Clamp refund to member.totalOriginationFee (min) and base unusedTime on the original tenor window, not the roll-extended maturity.

Confidence: high — code trace + sim

LOW PEN-T2-04

Early full repay without a callback overcharges origination fee to treasury (refund gate inverted vs comment)

WEAPONIZES

`PROTO-T2-03`

PRECONDITIONS

originationRate>0; borrower repays in full before maturity; the lender has NOT created a callback.

ATTACK PATH

1. Lender lends; totalOriginationFee accrues for the full tenor.
2. Borrower repay(lender) early: the refund branch requires _poolCallbacks[lender].isCreated==true.
3. No callback → refund skipped → full-tenor origination fee charged though the loan lived only part of the tenor. The comment says the refund is for the no-callback case; the code does the opposite.

IMPACT

Borrower overpays origination fee = P·orig·unusedTime/YEAR to treasury on every no-callback early repayment; a callback-holding lender instead gets the intended refund. Fee-misallocation, not theft.

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

```
lend at t0; warp(t0+tenor/2); prank(borrower); pool.repay(lender);
assert(originationFee includes full-tenor amount, no proration);
```

DETECTION & MITIGATION

Refund unused origination on early repay regardless of callback (align code with comment), guarding the underflow of PEN-T2-03.

Confidence: high — direct contradiction with the docstring

LOW PEN-T2-05

Terminal lender-triggered default bricks all on-chain settlement (grief/extortion, no cure, no post-default claim)

WEAPONIZES

`PROTO-T2-02`

PRECONDITIONS

Bullet: now > maturity+3d grace. Monthly: now > lastPaidTimestamp+30d+3d. Caller = any lender with non-zero principal (or borrower / factory owner).

ATTACK PATH

1. Solvent borrower intends to repay shortly after maturity but misses the 3-day grace by any margin.
2. A single lender (even holding dust principal) calls markPoolDefaulted() → defaultedAt=now.
3. repay/repayAll/repayInterest/lend/roll all carry nonDefaulted and now revert permanently. There is NO claim/withdraw/settle/redeem function in Pool.sol, so post-default there is no on-chain path for anyone to be paid.
4. Default is irreversible (defaultedAt never cleared).

IMPACT

One adversarial (or merely impatient) lender permanently blocks a solvent borrower from repaying on-chain and freezes every other lender's on-chain recovery. No direct profit — value is extortion (threaten default to extract off-chain payment) or reputational sabotage. Recovery becomes purely off-chain/legal.

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

```
activate bullet pool; warp(maturity+3d+1); prank(minorityLender); pool.markPoolDefaulted();
prank(borrower); expectRevert('PDD'); pool.repayAll();
```

DETECTION & MITIGATION

Restrict lender-triggered default to governor/factory, or add a borrower cure window / a repay path that stays open post-default, or require a quorum of principal to default.

Confidence: high

LOW PEN-T2-06 **KYC-revoked lender's stale callback permanently locks the pool out of rolls**

WEAPONIZES

`PROTO-T2-07`

PRECONDITIONS

Single-lender pool; lender creates a callback while [redacted], then is blacklisted in [redacted].

ATTACK PATH

1. requestCallBack() ([redacted], before maturity) → activeCallbacksCount=1.
2. Governor blacklistMember(lender) → isMember=false.
3. cancelCallBack() is [redacted] → the revoked lender can no longer cancel; no other actor can clear it.
4. requestRoll() forever reverts 'RCR' (needs zero callbacks). Borrower loses the roll/extension feature for the life of the loan.

IMPACT

Griefing / feature-lock: rolls permanently disabled until full repayment (which clears the callback). Any lender can also block rolls simply by refusing to cancel; KYC revocation makes it involuntary and unclearable. No fund loss.

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

```
lend; pool.requestCallBack();       .blacklistMember(lender);
prank(borrower); expectRevert('RCR'); pool.requestRoll();
prank(lender); expectRevert('NPM'); pool.cancelCallBack();
```

DETECTION & MITIGATION

Allow borrower/governor to clear callbacks of non-member lenders, or skip non-member callbacks in the requestRoll count.

Confidence: high

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
Direct theft / pool draining	Pools never custody funds — lend and every repay use safeTransferFrom peer-to-peer. There is no pool balance to steal; every external mutator is nonReentrant.
Cross-pool / token-callback reentrancy	Each pool has its own guard and no shared mutable state; assets are governance-whitelisted stablecoins, so no attacker-controlled reentrant token. markPoolDefaulted lacks a guard but performs no transfers.
Borrower == lender self-deal	_initLender requires borrower != member on both the public and whitelist paths.
lend → blacklist → funds locked	_repayTo has no membership check and short-circuits only on principal==0, so a blacklisted lender can still be fully repaid — principal is never trapped.
Lender underpaid by interest math	Every identified proration divergence rounds in the LENDER's favor (borrower overpaid); full repay at/after maturity is exact. No lender-underpaid sequence found.
Treasury mis-routing / fee theft	Fees always go to [redacted]().treasury() from the borrower; excess over-collections go to lender/treasury, never an attacker-chosen address.

03 REMEDIATION PRIORITY

PRIORITY	ID	FIX
P1	PEN-T2-01	Set accrualTs=endDate and deduct totalInterest by the same interval actually paid; or cap endDate at nextPaymentTs and let penalty (not interest) cover lateness. Reconcile cumulative RepayedInterest vs P·R·elapsed/YEAR.
P1	PEN-T2-02	Prorate each Position over its own [startAt,endAt] in _dueInterestOf instead of a single aggregate accrualTs; on repay, cross-check dueOf against the per-position sum used by balanceOf.
P2	PEN-T2-03	Clamp refund to member.totalOriginationFee (min) and base unusedTime on the original tenor window, not the roll-extended maturity.
P2	PEN-T2-04	Refund unused origination on early repay regardless of callback (align code with comment), guarding the underflow of PEN-T2-03.
P2	PEN-T2-05	Restrict lender-triggered default to governor/factory, or add a borrower cure window / a repay path that stays open post-default, or require a quorum of principal to default.
P2	PEN-T2-06	Allow borrower/governor to clear callbacks of non-member lenders, or skip non-member callbacks in the requestRoll count.