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

ARI

PENETRATION TEST & EXPLOIT ANALYSIS

PENETRATION TEST — STAKING

[redacted] · Track T4 — Validator-node staking & reward distributor

PROJECT	TRACK	DATE
▮▮▮▮▮▮	T4	17 May 2026

EXPLOITS	REFUTED	CLASSIFICATION
6 weaponized	3 defended	PUBLIC SAMPLE

00 ENGAGEMENT SUMMARY

The exploitable surface is owner/validator operational hazards plus a validator fee-siphon, all confirmed. createNode's unbounded fee bricks a node's rewards in an unrescuable deadlock; one inactive node reverts the whole distributeReward batch (a validator can front-run to grief every staker); and setNodeValidator zeroes validator accounting, deactivating a live node and underflow-locking the new validator's own stake. The _unstake CEI violation (principal transferred before delete, no nonReentrant) is a double-principal drain ONLY under a callback-capable token — the deployed [redacted] is a plain ERC20, so it is refuted on-chain and captured as latent. The reward leg is correctly ordered (no reward doubling), there is no first-staker/share-inflation vector (push-based rewards), and no non-owner reward injection.

CRITICAL	HIGH	MEDIUM	LOW	INFO
0	0	3	3	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-T4-01	MEDIUM	PROTO-T4-01	createNode fee ≥ 100% permanently bricks node reward distribution (unrescuable deadlock)
PEN-T4-02	MEDIUM	PROTO-T4-02	One inactive node reverts the entire distributeReward batch — validator griefing DoS on all stakers
PEN-T4-03	MEDIUM	PROTO-T4-03	setNodeValidator zeroes stakedByValidator — deactivates a live node and underflow-locks the new validator's own stake
PEN-T4-04	LOW	reentrancy probe	Missing reentrancy guard + CEI violation in _unstake (principal transferred before delete) — double-principal drain under callback tokens
PEN-T4-05	LOW	PROTO-T4-06	Validator can siphon almost all node rewards via a near-100% fee
PEN-T4-06	LOW	PROTO-T4-05	uint96 reward cap lock + same-timestamp distribution div-by-zero (edge DoS)

MEDIUM PEN-T4-01 createNode fee ≥ 100% permanently bricks node reward distribution (unrescuable deadlock)

WEAPONIZES

PROTO-T4-01

PRECONDITIONS

Owner calls createNode(validator, fee) with fee > DENOMINATOR. createNode has NO fee bound (unlike setFee). Owner error / malicious owner / bad automation input.

ATTACK PATH

1. createNode(V, 2e18) → fee=200%, lastDistributionId=0.
2. Stakers stake; node active.
3. Owner rescue attempt setFee(id, 0.1e18): with lastDistributionId=0 the maturity branch is false, so fee stays 2e18; only nextFee/feeUpdateDistributionId=1 are set.
4. distributeReward([id],[R]): getFee checks feeUpdateDistributionId(1) ≤ lastDistributionId(0) → false → returns OLD fee 2e18; feeAmount=2R; R−2R underflows → REVERT.
5. Because the batch reverts, lastDistributionId never increments to 1, so nextFee can NEVER mature. getFee is frozen on the broken fee forever.

IMPACT

Node permanently unable to distribute rewards; setFee cannot rescue (activation needs a successful distribution, distribution needs a sane fee — circular deadlock). Stakers earn zero forever (principal still withdrawable).

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

```
id=createNode(V,2e18); prank(V); stakeFor(id,MIN); setFee(id,0.1e18);
expectRevert(arithmetic); distributeReward([id],[100e18]); // reverts forever;
getFee(id)==2e18
```

DETECTION & MITIGATION

Add require(fee < DENOMINATOR) to createNode (mirror setFee). Alert on any node whose distributeReward reverts with arithmetic underflow.

Confidence: high (confirmed)

MEDIUM PEN-T4-02

One inactive node reverts the entire distributeReward batch — validator griefing DoS on all stakers

WEAPONIZES

`PROTO-T4-02`

PRECONDITIONS

Non-owner attacker controls one validator in the owner's batch. Owner distributes to multiple nodes in one call (standard). No owner cooperation beyond normal batch distribution.

ATTACK PATH

1. Attacker-validator initially stakes ≥ minimum so their node K is in the rota.
2. _distributeReward hard-requires stakedByValidator ≥ validatorMinimalStake ('NNA'). Attacker calls unstake() on their validator stake, dropping node K below minimum → inactive.
3. Attacker front-runs: watch the mempool for the owner's distributeReward tx and drop below minimum in the same block just before it lands.
4. The batch loop hits node K → reverts 'NNA' → the WHOLE batch reverts atomically. No node receives its distribution. Attacker re-stakes and repeats each cycle.

IMPACT

One malicious/careless validator denies that distribution round to EVERY staker across ALL nodes in the batch. Repeatable, low-cost (temporary self-unstake), protocol-wide amplification from a single node.

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

```
prank(validatorK); staking.unstake(kValidatorStakeId); // node K < min
expectRevert('NNA'); staking.distributeReward([1..N], rewards);
```

DETECTION & MITIGATION

Distribute per-node with try/catch or skip-inactive logic; require a validator un-stake cooldown/notice; monitor isActive drops just before scheduled distributions.

Confidence: high (confirmed)

MEDIUM PEN-T4-03

setNodeValidator zeroes stakedByValidator — deactivates a live node and underflow-locks the new validator's own stake

WEAPONIZES

`PROTO-T4-03`

PRECONDITIONS

Owner calls setNodeValidator(id, V) on a node that already has stakes, and/or V already holds a normal stake on that node. The NatSpec warns but nothing enforces it. Owner-only trigger causing non-owner fund lock.

ATTACK PATH

1. setNodeValidator sets validator=V and unconditionally stakedByValidator=0, while totalStaked still includes everyone's stakes.
2. Deactivation: stakedByValidator 0 < minimum → node instantly inactive; distributions revert ('NNA', feeds PEN-T4-02) and new stakeFor is rejected.
3. Fund lock: if V previously staked S0 as an ordinary staker (counted in totalStaked, never in stakedByValidator), after promotion V stakes S1 as validator → stakedByValidator=S1.
4. V calls unstake(oldStakeId S0): _unstake runs stakedByValidator −= S0; if S0>S1 the checked subtraction underflows → REVERT. V can never unstake S0 — principal permanently locked.

IMPACT

Live node bricked (deactivated mid-life) and the new validator's pre-existing principal irrecoverable via underflow. Combined with PEN-T4-02 it can also grief the whole batch.

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

```
prank(V); sid=stakeFor(id,S0); owner.setNodeValidator(id,V); // stakedByValidator=0
prank(V); expectRevert(arithmetic); staking.unstake(sid); // locked
```

DETECTION & MITIGATION

setNodeValidator must migrate accounting: forbid when the node has active stakes, or set stakedByValidator to the new validator's actual staked total; alert on setNodeValidator against nodes with totalStaked>0.

Confidence: high (confirmed)

LOW PEN-T4-04 **Missing reentrancy guard + CEI violation in _unstake (principal transferred before delete) — double-principal drain under callback tokens**

WEAPONIZES

`reentrancy probe`

PRECONDITIONS

The staking token must invoke a recipient callback on plain transfer (ERC777/677-style or hooked). Contract holds pooled funds from other stakers. DEPLOYED token is [redacted] (standard ERC20, no hook) → NOT exploitable live; latent for any future/alt-token deployment.

ATTACK PATH

1. _unstake order: (1) _withdrawReward → token.safeTransfer(reward); (2) read amount, decrement totals, token.safeTransfer(amount); (3) delete stakeInfo[stakeId].
2. Reward leg is safe (withdrawnReward incremented before its transfer).
3. Principal leg is NOT: delete happens AFTER the principal transfer. With a callback token, at step (2) the attacker re-enters unstake(sameId); stakeInfo still present → inner call transfers amount a second time and deletes; outer resumes (delete no-op).
4. Net payout = reward + 2·A, drained from other stakers' pooled principal (needs totalStaked ≥ 2A).

IMPACT

On a callback-capable token: theft of other stakers' principal (double-withdraw). On [redacted]: no impact. The defect is a genuine CEI violation; safety rests entirely on the token being hook-free.

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

```
// mock ERC777: attacker.unstake() -> in tokensReceived for the PRINCIPAL transfer, call
staking.unstake(sameId) again -> balance += 2A. (No-op on standard ERC20.)
```

DETECTION & MITIGATION

Add nonReentrant to unstake/batchUnstake/withdrawReward, move delete stakeInfo BEFORE the transfers, and reject/allowlist transfer-hook tokens at initialize.

Confidence: plausible (latent)

LOW PEN-T4-05 **Validator can siphon almost all node rewards via a near-100% fee**

WEAPONIZES

`PROTO-T4-06`

PRECONDITIONS

Attacker is a node validator. setFee is validator-callable and only bounds fee < DENOMINATOR, allowing 0.9999e18.

ATTACK PATH

1. Validator setFee(id, 0.9999e18) → nextFee, effective one distribution later.
2. On the subsequent owner distribution getFee returns 99.99%: feeAmount = reward·0.9999 to collectedFee; stakers split ~0.01%.
3. withdrawFee(id) → validator pockets nearly the entire node reward.

IMPACT

Validator captures up to 99.99% of their node's stakers' rewards. Mitigated by the one-distribution delay + FeeUpdated event giving attentive stakers a window to exit — abuse-within-design, not silent theft.

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

```
prank(V); setFee(id,0.9999e18); owner.distributeReward([id],[R]);
owner.distributeReward([id],[R2]);
prank(V); withdrawFee(id); // ~0.9999*R2
```

DETECTION & MITIGATION

Cap validator-settable fee to a protocol maximum (e.g. 20–30%); timelock/notice on increases beyond a threshold.

Confidence: high (confirmed)

LOW PEN-T4-06 **uint96 reward cap lock + same-timestamp distribution div-by-zero (edge DoS)**

WEAPONIZES

`PROTO-T4-05`

PRECONDITIONS

A stake's accumulated reward / a single distribution reward / collectedFee exceeds uint96 max (~7.9e28); or two distributions land in the same block.timestamp with partial stakes present.

ATTACK PATH

1. uint96 cap: rewardOf returns .toUint96(); if accrued > uint96 max it reverts, and since _unstake calls _withdrawReward first, unstake ALSO reverts → principal locked until the reward is reduced (which the staker cannot do).
2. Same-timestamp div0: maxTotalPowerXTime = stakedIn·(now − prevDistTimestamp); if a distribution lands at the same timestamp as the prior one with stakedIn>0, the denominator is 0 → revert.

IMPACT

Localized DoS: oversized-reward stake cannot be unstaked (principal temporarily locked); owner cannot run two distributions in one block with partial stakes. Hard to reach at [redacted] supply (~1e27 < uint96 max); mostly self-inflicted / owner-timing.

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

```
// same block twice with a fresh partial stake -> expectRevert(arithmetic). // uint96: push
accumulated > type(uint96).max then unstake reverts.
```

DETECTION & MITIGATION

Use uint256 for reward math (or saturate); require block.timestamp > prevTimestamp for a distribution.

Confidence: high (confirmed)

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
Staking reentrancy on the DEPLOYED [redacted] token	[redacted] is a standard ERC20 with no transfer callback, so the _unstake CEI gap cannot be reentered live. The reward leg is correctly ordered regardless of token — reward can never be double-withdrawn. Only the principal leg is latently unsafe, only under a callback token (PEN-T4-04).
Staking first-depositor / share-inflation	Rewards are PUSH-based, allocated by time-weighted stake power; there is no shares=deposit/totalAssets accounting, so the ERC4626-style first-staker vector does not exist. Rounding leaves unwithdrawable dust — unfavorable to an attacker.
Non-owner reward injection / distribution manipulation	distributeReward/createNode/setNodeValidator/upgrade are onlyOwner; the only non-owner levers are griefing (PEN-T4-02) and the validator fee (PEN-T4-05), neither of which lets a staker steal another's principal on [redacted].

03 REMEDIATION PRIORITY

PRIORITY	ID	FIX
P1	PEN-T4-01	Add require(fee < DENOMINATOR) to createNode (mirror setFee). Alert on any node whose distributeReward reverts with arithmetic underflow.
P1	PEN-T4-02	Distribute per-node with try/catch or skip-inactive logic; require a validator un-stake cooldown/notice; monitor isActive drops just before scheduled distributions.
P1	PEN-T4-03	setNodeValidator must migrate accounting: forbid when the node has active stakes, or set stakedByValidator to the new validator's actual staked total; alert on setNodeValidator against nodes with totalStaked>0.
P2	PEN-T4-04	Add nonReentrant to unstake/batchUnstake/withdrawReward, move delete stakeInfo BEFORE the transfers, and reject/allowlist transfer-hook tokens at initialize.
P2	PEN-T4-05	Cap validator-settable fee to a protocol maximum (e.g. 20–30%); timelock/notice on increases beyond a threshold.
P2	PEN-T4-06	Use uint256 for reward math (or saturate); require block.timestamp > prevTimestamp for a distribution.