Token Forge Smart Contract Security Review

Self-assessment, AI-assisted. Not an independent third-party audit.

Subject	Token Forge launchpad contracts on Base mainnet (chain id 8453)
Reviewed source	The contracts as deployed on Base mainnet on 2026-11-13; the source of each is verified on Basescan at the addresses listed in section 2
On-chain configuration	Roles, addresses and parameters read at block 57,063,598 (2026-11-13), after deployment; pool balances, prices and supplies are from the reviewers' chain reads at block 55,355,114 (2026-11-09)
Toolchain	Solidity 0.8.28 (via-IR, optimizer 200 runs, Cancun), OpenZeppelin 5.3.0, Uniswap V3 core and periphery
Review type	Manual code review by independent AI reviewers, one per contract group and one for the system level, with executable proofs of concept, consolidated by the Token Forge core team
Classification	Public
Disclosure	Through the contact published on the website

1. Executive summary

Token Forge is a no-code token launchpad: a creator pays a USD-denominated fee in PBTC, the factory clones a fixed-supply ERC-20, an optional presale (fixed price, stepped price or bonding curve), an optional staking pool and an optional vesting wallet, and at launch opens a full-range Uniswap V3 Token/PBTC position, optionally a Token/WETH position, and locks the main position either in the factory or in Team Finance. Prices come from a 30-minute Uniswap V3 TWAP oracle. PBTC holders stake in a separate upgradeable contract that mints COAL on a ten-year schedule.

The review found no issue that lets an outsider take user funds. Buyers' PBTC moves only into the launch pools or back to the buyers; the factory holds project supply only until launch and burns what the launch does not use; the routers hold no user funds between transactions. The two Medium findings are trust and market-structure assumptions: the owner Safes and the ProxyAdmin Safe can replace the executing code of every upgradeable component without a delay, and the PBTC/USDC pool behind the oracle is thin enough that its 30-minute TWAP can be moved by a determined party for one window at a cost of under 20 USD. Three Low findings describe bounded design properties; nine Informational items record observations, documentation drift and operational duties.

Severity	Count
Critical	0
High	0
Medium	2
Low	3
Informational	9

2. Scope

All contracts listed below were read in full. Proxies use OpenZeppelin `TransparentUpgradeableProxy` behind the shared ProxyAdmin 0xDC203437A0D11aFa20b867433C3C2DeFf7760209.

Contract	Source path	Address	Lines
PBTCForgeFactory	contracts/forge/PBTCForgeFactory.sol	proxy 0x5E0bdAb125dC8771433DdceF169Ced769573a833, implementation 0x0Ff019eB4e1BBEFF7F75B9e7f6ABA7D02613695F	1,026
PBTCForgeStorage	contracts/forge/PBTCForgeStorage.sol	inherited by the factory and both libraries	379
PBTCForgeHelperLib	contracts/forge/libraries/PBTCForgeHelperLib.sol	0xdC52CA7e468c960e668A9cf3Fe30969772338025	921
PBTCForgeLiquidityLib	contracts/forge/libraries/PBTCForgeLiquidityLib.sol	0x31205d17ef9749F20c7A9120240459C7B40E2d20	355
PBTCForgeOracle	contracts/forge/PBTCForgeOracle.sol	proxy 0x44205F6B099B8d6d179235d33eF352c3901d648A, implementation 0xa4E3D08C1082AA963ad664bf3c76500d17625E55	240
PBTCForgePresaleRouter	contracts/forge/PBTCForgePresaleRouter.sol	0x1089cddb4c3C1DdcB30006DBb5C8fddf8bE90f27	1,471
PBTCForgeCreateRouter	contracts/forge/PBTCForgeCreateRouter.sol	0x73aBe958a3d99523eFf1D460f71f3767a2AB77f2 (approved on the factory)	176
PBTCForgePresaleFixedPrice	contracts/forge/PBTCForgePresaleFixedPrice.sol	clone implementation 0x456935C1DD939361a39c93232bd136a075C98F26	730
PBTCForgePresaleBondingCurve	contracts/forge/PBTCForgePresaleBondingCurve.sol	clone implementation 0x1B3215ff2fac087d93094F23356982C83B0F9d60	1,202
PBTCForgePresaleSteppedPrice	contracts/forge/PBTCForgePresaleSteppedPrice.sol	clone implementation 0xEACd542a814eAE50DEc082B94498ffd25FFCb552	1,133
PBTCForgeStaking	contracts/forge/PBTCForgeStaking.sol	clone implementation 0x24100d235d7E1E7BC828D298608eEDC598a4D821	395
PBTCForgeToken	contracts/forge/PBTCForgeToken.sol	clone implementation 0x380acd6da8f9624630c5a0c45AA51F78A4b909Ce	82

Contract	Source path	Address	Lines
PBTCForgeVestingWallet	contracts/forge/PBTCForgeVestingWallet.sol	clone implementation 0x1939f2984BCC82fD0acf33Ba11cd12B1FDD1dB4e	32
Coal	contracts/coal/Coal.sol	0x25B30C734F6f9359780637120f04103460D9Fa75	45
CoalStaking	contracts/coal/CoalStakingUpgradable.sol	proxy 0x8fd52B375A006436b536768e5c292A3C25AA317C, implementation 0xF4715C1b81780e50eBaDf73501B70eb6cc2A8646	460
PBTC token	contracts/PepeBitcoin.sol	0x31705474c1F2DE7F738E34233c49522CA1E3C53c	16
Interfaces	contracts/forge/interfaces/*.sol (7 files)		424
Total			9,087

2.1 Trusted dependencies

The contracts call and trust the following third-party code, which was not reviewed: OpenZeppelin Contracts 5.3.0 (proxies, Clones, SafeERC20, Initializable, OwnableUpgradeable, ReentrancyGuardUpgradeable, ReentrancyGuardTransient, VestingWalletUpgradeable, ERC20Capped, AccessControl), Uniswap V3 on Base (factory, pools, NonfungiblePositionManager 0x03a520b32C04BF3bEEf7BEb72E919cf822Ed34f1, SwapRouter02 0x2626664c2603336E57B271c5C0b26F421741e481, QuoterV2, OracleLibrary), the Team Finance NFT locker 0x4F0Fd563BE89ec8C3e7D595bf3639128C0a7C33A (itself a proxy under third-party control), USDC and WETH on Base, and the Safe 1.4.1 multisig contracts that hold the privileged roles.

3. Methodology and limitations

Independent AI reviewers (Claude, Anthropic) each audited one contract group with no knowledge of each other's work: factory core (factory, storage, both libraries, token, vesting wallet); presales and routers; oracle, staking and token contracts. A further reviewer covered the system level, fund flows and live state. Each reviewer read every line of their scope, traced the cross-contract calls, ran the repository test suite and wrote executable proofs of concept: 46 in total, 11 of them on a mainnet fork: a local copy of Base mainnet state at block 55,300,000 on which transactions execute against the real Uniswap V3 pools, router and position manager. Live state was read from Base mainnet. The Token Forge core team consolidated the reports, re-verified every claim against the reviewed source and re-made every severity judgment.

The storage layouts of the three proxied contracts (factory, oracle, CoalStaking) were compared against the raw storage of the live proxies and the OpenZeppelin upgrades manifest: every variable sits where the proxy's storage holds it, nothing moved or changed type, and the fields whose names differ from the manifest carry @custom:oz-renamed-from annotations. CoalStaking declares the openzeppelin.storage.Pausable namespace its proxy's storage contains and never reads or writes it, and the OpenZeppelin upgrades plugin validates the reviewed implementation against the live proxy.

Limitations. No independent third-party auditor was engaged. No static analysis tool or formal verification was used. Third-party contracts (OpenZeppelin, Uniswap V3, Team Finance, USDC, WETH, Safe) were not reviewed. Configuration parameters were assessed for safety, not for economic suitability. The team's position is that this document is a self-assessment that records what was examined and what was found; it is not a guarantee of the absence of vulnerabilities.

3.1 Severity scale

Severity	Meaning
Critical	Direct loss or freezing of user funds by an outsider, exploitable with the system as deployed
High	Loss or freezing of user funds under realistic conditions, or an outsider's ability to deny the protocol's core function at low cost
Medium	A trust assumption on privileged parties or market structure that users cannot verify or wait out, or an outsider's ability to cause bounded loss or disruption
Low	A bounded design property with limited impact, or an edge that requires unusual conditions
Informational	Observations, documentation drift, operational duties and recommendations without a direct security impact

4. System description

4.1 Architecture and trust boundaries

PBTCForgeFactory is an upgradeable proxy whose storage is declared in PBTCForgeStorage. The factory holds routing, validation, fee collection, upkeep and the delegatecall plumbing; the heavy logic lives in two separately deployed contracts that inherit the same storage layout and run under delegatecall: PBTCForgeHelperLib (project evaluation, presale and staking deployment, token deployment, launch distribution, Team Finance locking, LP claims) and PBTCForgeLiquidityLib (pool opening, token replacement, secondary leg). _lib re-raises revert data unchanged, and every caller checks the length of the returned data. Tokens, presales, staking pools and vesting wallets are EIP-1167 clones created and initialized in the same transaction; presale and staking clones record msg.sender as their factory in initialize (token and vesting clones have no factory role: the token mints its supply to the factory once and records the project creator as its CCIP admin, the vesting wallet belongs to its beneficiary), all implementations disable their own initializers in the constructor, every lifecycle hook is onlyFactory, and the presale state transitions emit events.

Trust boundaries: the factory owner Safe configures parameters, implementations, libraries and external addresses; the ProxyAdmin Safe can replace the implementations of the factory, oracle and CoalStaking proxies; the oracle owner Safe sets pools and window; the COAL admin Safe grants minting. The owner withdraws fee PBTC with withdrawFees, which refuses to touch what affiliates are owed. A project's creator alone can launch its sold-out presale early (creatorEarlyLaunch), take the LP position after the lock and its fees (claimLpNft, claimLpFees) and move the lock to Team Finance (convertToTeamFinance). Everything else is permissionless: project creation, buying, selling, claiming, refunds, upkeep and burnUnallocated. The routers have no owner and no privileged function. A token's CCIP admin can register the token with Chainlink CCIP and hand that role on or clear it; it has no power over the token's supply, balances, transfers or approvals (I-09).

4.2 Project lifecycle

Creation (_createProject) validates supply (10,000 to 100,000,000,000 whole tokens), name and symbol, staking allocation (at most 25% of supply, 30 to 720 days, lock at most 7 days), vesting allocation (1% to 20%, 180 to 720 days) and LP lock (180 to 720 days); reads the PBTC price; charges baseFeeUsd plus add-ons, applies the COAL discount when the payer burns coalDiscountUsd of COAL, adds the Team Finance budget when chosen, and converts the sum to PBTC with ceiling division. The whole fee is pulled into the factory and stays there as the buffer that pays affiliate commissions; the Team Finance budget is swapped to USDC at once (swapPbtcForUsdc, bounded by slippageBps) and held per project; the owner withdraws fee PBTC above the open commissions with withdrawFees. The token clone is deployed at a CREATE2 address derived from the project id and the previous block's hash, with its whole supply held by the factory. A project without presale launches in the creation transaction with creator-supplied PBTC worth at least 1,900 USD.

Projects with a presale wait for upkeep, which anyone can call. It scans projects from a cursor, asks the helper library whether a project is ready or must fail, hands every launch attempt a fixed stipend of 14,500,000 gas inside a self-call protected by try/catch, pauses a project whose attempt failed for one hour (finalizeRetryAfter), and records the first failed attempt in launchFailingSince; a project that is evaluated as not ready clears that record, so the clock measures an uninterrupted run of failed attempts. A launch that keeps failing 14 days after its first failed attempt is abandoned: the presale is marked failed and buyers take refunds. Two revert reasons are treated as transient and are neither recorded nor paused: TokenAddressPreempted and SecondaryRouteOffPrice. A readiness view that reverts makes upkeep skip that project and emit ProjectEvaluationSkipped. The creator may launch early with creatorEarlyLaunch when the presale is ready; bonding curves have no early launch.

4.3 Launch liquidity mechanism

mintLp first sizes the secondary leg, then checks the project token's PBTC pool, and its WETH pool when the secondary leg produced WETH, with _poolUsable: a pool that does not exist, or whose balances of both assets are zero, is used; a pool that holds anything was prepared by an outsider and causes _replaceToken. Replacement burns the old supply held by the factory, deploys a fresh token, which takes over the replaced token's CCIP admin, at an address derived from the previous block's hash and an attempt counter (up to 64 candidates, skipping any candidate that already has a PBTC or WETH pool at the 0.3% tier), and repoints the presale (setToken, allowed only before claiming is enabled) and the presale-staking pool (setStakingToken, allowed only in presale mode). Buyers' entitlements are amounts of tokens, so nothing else changes. If every candidate has a pool the attempt reverts TokenAddressPreempted.

_openPool creates the pool at the launch price or, for an empty pool an outsider initialized at another price, moves it there with a swap of one wei bounded by the target price (a swap slides through empty ranges without taking anything; the callback

verifies the pool against the Uniswap factory), then mints a full-range position with a 1% amount bound to the factory. distributeLaunch then pays the treasury 1% of supply, the vesting wallet, the staking pool (rewards plus whatever was staked during the presale), hands the presale exactly totalTokensSold minus presale stakes, enables claiming, switches a presale-staking pool to normal mode and burns everything left in the factory.

The secondary leg (_secondaryLeg) takes secondaryLpBps of the raised PBTC, capped at secondaryLpCapUsd by the oracle price, and sells it for USDC with a Uniswap price limit slippageBps below the oracle price, so the swap fills only what the pool takes down to that limit and the rest stays in the main position. If the bounded sale yields less than 25 USDC, a minimum sale is made with a 10% bound, sized to reach 25 USDC after the 0.3% pool fee. The USDC is swapped to WETH and checked against the oracle's ETH price within slippageBps plus 5 bps. A market more than 10% below its own TWAP sells nothing and the attempt reverts SecondaryRouteOffPrice, which upkeep skips without recording. The Token/WETH position receives project tokens in proportion to the PBTC sold.

4.4 Failure and refund handling

Fixed and stepped presales that end without meeting a soft cap are failed on the first evaluation after presaleEnd; the creator's held USDC budget is returned (or recorded in usdcRefundsOwed when the transfer fails) and buyers call claimRefund for their full contribution. A bonding curve has no deadline and fails only through abandonment; markFailed snapshots the PBTC it holds, zeroes unclaimed creator fees and pays each buyer refundPoolPbtc * refundBasis / totalRefundBasis, where basis is the net PBTC paid for the tokens the buyer holds. launched and failed are mutually exclusive, withdrawRaised is one-shot and requires launched, so raised PBTC is never both sent to the pool and refunded. A Team Finance lock that fails (fee above the budget, locker reverting) falls back to the internal lock and refunds the budget without failing the launch.

4.5 Presales and routers

Fixed price: one price, allocation at most half the remaining supply plus one token, optional per-wallet cap with overflow into linear vesting, PBTC soft cap at least 9,000 USD at creation, optional USD soft cap with a buyer-count condition, COAL burn for early access, optional presale staking. Sold out means at most 1/10,000 of the allocation is left. Exact-input buys revert when they would yield zero token wei; exact-output buys charge the ceiling.

Stepped price: 2 to 20 equal buckets on an arithmetic price ladder from startPrice to endPrice, both derived from the creator's USD hard cap and multiplier (1.5x to 4x); the active step is the later of the time-based step and the first unsold step, and unsold tokens of the steps already passed roll into the active step at its price. The creator gives both caps in USD. The factory requires hardCapUsd * M / (M + 10) <= usdSoftCap <= hardCapUsd, where M is the multiplier scaled by ten, and usdSoftCap >= 9,000 USD, then converts both caps to PBTC at the same oracle price, so the range holds whatever the price does before the creation is mined. It stores hardCapUsd as the presale's usdSoftCap: a raise below the PBTC soft cap launches only when its USD value reaches the USD value the hard cap had at creation, with one buyer per 1,000 USD of it and 25% of the allocation sold.

Bonding curve: linear price in tokens sold with tier hard caps of 5,000, 10,000 or 25,000 USD converted to PBTC at creation; buys round the curve integral up and sells round it down, so the curve always covers any sell and the last holder can sell everything; a 5% sell fee is split between the creator and a PBTC-to-COAL burn (falling back to the treasury when the oracle or the swap fails); sellExactOut reverts when the curve cannot pay the requested amount; standing sell-operator approvals let a router sell on a holder's behalf and receive the proceeds.

PBTCForgePresaleRouter swaps any token or ETH to PBTC through SwapRouter02, buys for a recipient with the presale allowance set to the swap output and reset afterwards, measures the PBTC spent as a balance delta, refunds leftovers to the caller, enforces the caller's deadline on every trade and hands bonding presales a five-minute deadline of their own, and sells only the caller's own bonding position. PBTCForgeCreateRouter swaps USDC to PBTC along a caller path that must be made of whole hops, start with USDC and end with PBTC, calls createProjectFor(msg.sender, p) with the factory allowance set to the swap output, and refuses the COAL discount.

4.6 Oracle

PBTCForgeOracle quotes the arithmetic-mean tick of the PBTC/USDC 0.3% pool over twapWindow for 10^12 PBTC (1e30 wei), which yields the price of one PBTC in USD with 18 decimals; the COAL price multiplies the COAL/PBTC 0.3% pool's PBTC-per-COAL quote by that price; the ETH price is a 6-decimal quote of one WETH from the WETH/USDC 0.05% pool set by setEthReference. Every pool setter verifies the pool's pair; setTokens re-verifies every configured pool; setWindow accepts 5 minutes to 12 hours and requires every configured pool to keep more observation slots than the window has seconds, the property ringCoversWindow reports. The PBTC and COAL prices are exposed with 18 decimals (pbtcPriceUsd18, coalPriceUsd18), which every on-chain consumer divides by; the 6-decimal views pbtcPriceUsdc6 and coalPriceUsdc6 are the same quotes scaled down, and ethPriceUsdc6 is used only by the launch's WETH slippage check.

4.7 Staking

PBTCForgeStaking pays rewards in the staked project token from a fixed totalRewards over 30 to 730 days (the factory passes 30 to 720), linear or front-loaded (50/30/15/5 per quartile), with a MasterChef-style accumulator and a withdraw lock of at most 7 days. With presale staking it starts in presale mode: the presale records stakes without moving tokens; at launch the factory transfers the staked amount plus the rewards and switches the pool to normal mode. Emission while nothing is staked accrues to unallocatedRewards, which anyone may burn. A presale-staking pool emits from project creation, so presale buyers who stake early earn the larger share of a front-loaded pool; the creator chooses the pool's duration and mode, and the factory only requires the pool to outlive the presale by a week.

CoalStaking holds staked PBTC and mints COAL on a ten-year schedule (20M, 16M, 12M, 8M, 6.4M, 5.6M, 4M, 3.2M, 2.4M, 2.4M; 80M in total, which with the 20M pre-mint equals COAL's 100M cap). Rewards accrue continuously per unit of weight; a stake's weight grows from 1.0x to 2.0x in six 15-day steps scheduled as weight deltas at day-epoch boundaries and credited retroactively. The contract has no pause. withdraw mints nothing, so principal is withdrawable even at the cap. Any caller can close the ledger's missed day-epochs in bounded steps (rollEpochs), so the ledger can always be brought up to date after a long gap without interaction.

4.8 Affiliate commission

A wallet named as affiliate must have been approved by the owner (setAffiliateApproval); a creation naming any other wallet reverts, and the owner can revoke an approval at any time without touching what was already credited. The commission (affiliateCommissionUsd, an owner-set USD amount) is credited in PBTC to affiliateOwed and to the running total affiliateOwedTotal. Every creation fee stays in the factory, which holds no other PBTC at rest, so the fees of all creations back the open commissions together: claimAffiliateCommission pays as far as the factory's PBTC covers, anyone can add to the buffer with a plain transfer, and withdrawFees lets the owner take only what exceeds affiliateOwedTotal.

4.9 Privileged accounts and configuration

Factory owner, oracle owner, CoalStaking owner and COAL DEFAULT_ADMIN_ROLE: Safe 0x1482A1772c0Bdd1f61ddaF44DF8f6133EaA6223b. ProxyAdmin owner: Safe 0xB20eCD9FFF474460d41f6d5d97Ce674736243D90. Treasury: Safe 0x501292c577F55Af8393A36fEa82E2b51cdB49fBa. Each of the three is a Safe 1.4.1 with a single owner, the parent Safe 0x745b45913789382591f2450FC555E4B267949451 (Safe 1.4.1, 3 owners, threshold 2), and has no module and no guard. Every action of any of the three therefore needs two of the same three signers. COAL MINTER_ROLE is held only by the CoalStaking proxy. The approved create router is 0x73aBe958a3d99523eFf1D460f71f3767a2AB77f2. Each project token has a CCIP admin, the project creator unless handed on or cleared (I-09). There is no timelock in front of any role.

Configuration: base fee 25 USD, staking add-on 10 USD, vesting add-on 10 USD, Team Finance budget 140 USD, affiliate commission 10 USD, COAL discount burn 5 USD of COAL for 25% off internal fees and 10% off the Team Finance fee, COAL early access burn 10 USD, early window 1,800 s, slippage guard 200 bps, secondary LP 10% of the raise capped at 2,000 USD, oracle window 1,800 s. Oracle pools: PBTC/USDC 0.3% 0x3aD6F07ef4f663d9687B14d29DAa9B9aFF15A781 (observation ring 2,000 slots, liquidity 5.63e17, about 23.06M PBTC and 13,798 USDC in one full-range position, PBTC price 0.000598 USD); COAL/PBTC 0.3% (ring 2,056 slots, about 17.5M COAL and 2.87M PBTC, COAL price 0.000097 USD); WETH/USDC 0.05% 0xd0b53D9277642d899DF5C87A3966A349A798F224 (ring 6,700 slots, about 7.5M USD of liquidity) as the ETH reference. COAL supply 40.74M, 1.29M burned, cap 100M; 28.40M PBTC staked in CoalStaking. Team Finance NFT lock fee 0.0704 ETH. Base: 200 ms blocks, 40M block gas limit, 16,777,216 gas per transaction, base fee about 0.005 gwei.

5. Findings

M-01 Privileged roles can replace executing code and oracle inputs without delay

Severity	Medium
Location	PBTCForgeFactory.sol setLibraries, setImplementations, setOracle, setRouter, setPositionManager, setUsdc, setTfLocker; PBTCForgeOracle.sol setPools, setEthReference; Coal.sol DEFAULT_ADMIN_ROLE; ProxyAdmin 0xDC203437A0D11aFa20b867433C3C2DeFf7760209

Description. The factory owner can point helperLib and liquidityLib at any contract; the factory then executes that code under delegatecall on its own storage and balances. The same owner can replace the clone implementations used for future projects

and every external address the factory trades through. The ProxyAdmin owner can replace the implementation of the factory, the oracle and CoalStaking. The oracle owner can repoint either price pool to any pool of the same pair, including a freshly created one with negligible liquidity. The COAL admin can grant MINTER_ROLE to any address and mint up to the cap (about 59M COAL of headroom). Each role is a Safe owned by one parent Safe with three owners and a threshold of two, so the same three signers stand behind every role, and no timelock sits in front of any of them: a compromised or colluding signer set acts in one transaction.

Impact. The factory custodies locked LP positions, the unlaunched supply of every live project, USDC held for Team Finance locks and the fee buffer that backs affiliate commissions; a launch-ready presale's raised PBTC can be pulled through finalizeLaunch and withdrawRaised. CoalStaking custodies 28.40M PBTC. Buyers' PBTC in a presale that is not launch-ready is reachable only through markFailed, which refunds it to the buyers.

Evidence. Reviewer proof of concept, not kept in the repository: one setLibraries call installing a contract whose claimLpNft and claimLpFees selectors transfer assets takes a locked LP NFT, a live presale's whole token supply, the held USDC and the fee buffer. Role holders were read on chain.

Recommendation. Put a timelock (24 to 48 hours) in front of the ProxyAdmin, the factory owner, the oracle owner and the COAL admin, publish the signer set, and monitor LibrariesUpdated, ImplementationsUpdated, OracleUpdated, proxy Upgraded and COAL RoleGranted events.

M-02 The thin PBTC/USDC pool makes the 30-minute TWAP movable for one window at low cost

Severity	Medium
Location	PBTCForgeOracle.sol _quote; consumers PBTCForgeFactory.sol, PBTCForgeHelperLib.sol, PBTCForgeLiquidityLib.sol, PBTCForgePresaleFixedPrice.sol, PBTCForgePresaleSteppedPrice.sol, PBTCForgePresaleBondingCurve.sol

Description. The oracle's quality is that of the pool behind it. With 5.63e17 of liquidity in one full-range position and 13,798 USDC on the quote side, lifting the spot price by a factor k costs about 13,798 * (sqrt(k) - 1) USDC of input, and with no counter-trade during the window the TWAP follows. Measured on the mainnet fork (window 1,800 s, no other trader):

Target TWAP move	USDC in	TWAP after 1,800 s	Round-trip cost	25 USD creation fee at the moved price
+2%	137.56	598 to 610	0.82 USDC	40,984 PBTC (41,806 at the true price)
+10%	674.78	598 to 658	3.95 USDC	37,994 PBTC
+50%	3,107.07	598 to 897	16.91 USDC	27,871 PBTC

A downward move needs the same value in PBTC. A single-block move does not shift a 30-minute TWAP materially; the exposure is multi-block manipulation of a thin pool, and arbitrage during the window is bounded by the pool's own depth.

Impact. Creation fees, the Team Finance budget and the affiliate commission convert at the price (a +50% move cuts the PBTC due by a third, less than the round-trip cost unless many projects are created in one window). The 9,000 USD soft-cap floors and the bonding and stepped hard caps are fixed in PBTC at creation from the price. The USD soft-cap readiness of fixed and stepped presales is evaluated at the price (see L-03). The secondary leg's sale floor is 2% and 10% below the TWAP, so a depressed TWAP lets the launch sell PBTC deeper, bounded by the secondary share (at most 2,000 USD) times the discount; a spot more than 10% below the TWAP defers launch attempts without recording them. The COAL price inherits any move; the COAL/PBTC pool is thinner (about 3,400 USD).

Evidence. Reviewer test on the mainnet fork, not kept in the repository (three moves, outputs above); consumer paths traced in the source.

Recommendation. Deepen PBTC/USDC liquidity (protocol-owned liquidity scales the manipulation cost linearly with depth), consider a longer window for launch-gating reads, and bound consumers: evaluate USD soft caps against the smaller of TWAP and spot or over a grace window, and take the secondary floor as the higher of the TWAP-derived and a spot-derived bound.

L-01 Anyone can change a project's token address at launch for a pool creation and one wei

Severity	Low
Location	PBTCForgeLiquidityLib.sol _poolUsable, _replaceToken; PBTCForgeHelperLib.sol deployToken

Description. A project's token address is public from creation. An outsider who creates its Token/PBTC pool and transfers one wei of PBTC to it (about 4,613,465 gas plus the wei) makes _poolUsable false, and the launch gives the project a fresh token. The launch itself is unaffected: the old supply is burned, the presale and staking pool are repointed, and both pools open on the fresh token.

Impact. The address shown during the presale differs from the launched one; off-chain listings, favorites and integrations that cached the first address point at a burned token. Buyers' claims, stakes and vesting are unchanged. The ProjectTokenReplaced event records the change.

Evidence. FactoryLaunch.js (foreign assets in the pool give the project a fresh token; a presale-staking pool follows; a secondary pool prepared by an outsider replaces the token as well) and the mainnet fork LaunchGas.js (pool prepared by an outsider: 4,613,465 gas; fresh-token launch 11,566,582 gas).

Recommendation. Present the token address as provisional until LaunchFinalized, index ProjectTokenReplaced, and consider deploying the token only at launch for presale projects.

L-02 LP positions and fee collection are bound to the immutable creator address

Severity	Low
Location	PBTCForgeHelperLib.sol claimLpNft, claimLpFees, _lockLpInTf

Description. claimLpNft requires msg.sender == prj.creator and transfers the position to msg.sender with safeTransferFrom; claimLpFees collects to prj.creator; the Team Finance lock names prj.creator as withdrawal address. creator has no setter.

Impact. A creator that is a contract without onERC721Received can collect the position's fees but can never take the position; a creator whose key is lost can do neither. The liquidity then stays in the pool permanently. For launches through the create router the creator is the router's caller, so this applies equally there.

Evidence. Code reading; FactoryLaunch.js covers the claim paths for indexes 0 and 1.

Recommendation. Accept a to parameter on claimLpNft and claimLpFees, or add a creator-signed transfer of the creator role.

L-03 USD soft-cap readiness is evaluated at a caller-chosen moment against the live TWAP

Severity	Low
Location	PBTCForgePresaleFixedPrice.sol isLaunchReady; PBTCForgePresaleSteppedPrice.sol isLaunchReady; PBTCForgeHelperLib.sol evaluateProject

Description. A presale whose PBTC raise is below softCap but whose USD value reaches usdSoftCap is ready only while pbtcPriceUsd18() says so. upkeep is permissionless and fails such a presale the first time it is evaluated after presaleEnd while not ready; markFailed is irreversible.

Impact. A presale within the TWAP's short-term variance of its USD soft cap can be failed by calling upkeep during a dip, or launched during a spike, including a dip or spike produced as in M-02. No funds are lost; a launch can be denied.

Evidence. Code reading; the buyer-count condition on this path (I-05) is satisfiable by one party.

Recommendation. Snapshot the readiness result or the price at the first evaluation after presaleEnd, or require the USD condition to fail across a grace window before failing the presale.

I-01 Libraries are directly callable; helper library bytecode headroom

Both libraries are ordinary contracts with external functions. A direct call runs on the library's own empty storage (zero addresses, zero parameters) and therefore reverts or acts on nothing the factory owns; it has no effect on the factory. PBTCForgeHelperLib compiles to 21,846 bytes of the 24,576-byte EIP-170 limit, leaving 2,730 bytes for future changes; the heaviest launch attempt uses about 11,800,104 gas of the 14,500,000 stipend.

I-02 The two launch pools open at slightly different prices

The main position keeps the presale price exactly (the secondary share removes tokens and PBTC in the same ratio). The Token/WETH position receives tokens in proportion to the PBTC sold but WETH bought at up to slippageBps plus pool fees

below the oracle price, so the token opens a few percent cheaper there (the 2% sale bound, both pool fees and the WETH leg tolerance). An arbitrageur can take that difference at launch; the amount is bounded by the secondary share.

I-03 Affiliate commission accounting

A referred creation can credit more than it brings in: with the COAL discount and Team Finance together, a creation leaves 4.75 USD of fee PBTC in the factory against a 10 USD commission (`pbtcDue` and `tfPart` are each rounded up separately). The buffer of all other creation fees absorbs this, and claims are paid first come, first served from it, so the owner has to leave enough behind when withdrawing fees and to top the factory up with a transfer if referred discounted creations ever dominate (`FactoryLaunch.js`, "Affiliate commission"). Only wallets the owner approved can be named, so the owner chooses who can draw on the buffer and can revoke a wallet that refers creations made only for the commission.

I-04 The presale router trusts the caller's presale address; bonding sell operators receive proceeds

The router does not check that `presale` is a factory project. It approves only the PBTC from the caller's own swap, measures spend by balance delta and reverts if the presale overdraws, so a hostile presale takes at most that PBTC (`PresaleRouter.js`). `setSellOperator` is a standing approval and the PBTC goes to the operator; the frontend should pass only factory-derived presale addresses and request operator approval only for the audited router.

I-05 Creator parameters beyond what the frontend offers

Within the factory's checks a creator may set a presale window of up to 365 days (buyers' PBTC is held until then if the soft cap is unmet), an arbitrarily high fixed price or stepped USD hard cap, whale overflow with an unbounded vesting horizon, and a USD soft cap whose buyer-count condition (`ceil(usdSoftCap / 1,000 USD)` buyers) is satisfiable with one-wei buys for fresh addresses. The 25% minimum-sold condition of stepped presales gates only the USD path. None of these lets the creator take buyers' funds.

I-06 Unused fields and interface differences

`CreateParams.presaleInit` and `CreateParams.mode` are accepted and not read. CoalStaking holds storage its reward logic does not read: `v2Active`, `weightOf`, `userRewardPerWeightPaid`, `rpwDelta` (written every epoch) and the `openzeppelin.storage.Pausable` namespace. The fixed presale stores `hardCap` but does not enforce it; its `calculateBuyExactIn` reports `pbtcAccepted = pbtcIn` for an input that the buy rejects as below one token wei. `IPBTCForgePresaleSteppedPrice.startTime()` has no counterpart on the contract (`presaleStart`). The liquidity minimum for projects without presale is 1,900 USD (`MIN_DIRECT_LIQUIDITY_USD`), below the 2,000 USD the app asks for, so a price move before mining does not fail a creation. Presale clones snapshot `oracle`, `treasury`, `swapRouter` and `slippageBps` at creation, so owner changes do not reach running presales. The oracle has no storage gap.

I-07 The create router refunds its whole PBTC balance

The refund is `balanceOf(address(this))`, not the unused part of the swap output. The router holds no PBTC between transactions, so in operation the two are the same amount.

I-08 Market depth of the fee swaps

The Team Finance budget swap at creation sells about 140 USD of PBTC with `amountOutMinimum` 2% below the TWAP value. At the reviewed depth one such sale moves the spot about 1.9%, so a second Team Finance creation inside the same window, before arbitrage restores the price, reverts in the swap and the creation reverts as a whole. The bonding curve's COAL burn swap applies the same 2% bound to the COAL/PBTC pool and sends the PBTC to the treasury when the bound is missed.

I-09 The creator holds the token's CCIP admin role

Each project token records a CCIP admin: the project creator at creation, and the replaced token's admin when a launch replaces the token. Chainlink's registry module lets only this address propose itself as the token's administrator in the CCIP `TokenAdminRegistry`, where the administrator chooses the token pool. The holder can hand the role on or set it to the zero address with `setCCIPAdmin`; once it is zero nobody can set it again. The role gives no power over the token: `setCCIPAdmin` writes one address, and no balance, supply, transfer or approval path reads it. Whoever holds it controls the token's CCIP configuration, and with it the tokens that users send through a pool that holder chose; holders who do not bridge are not affected. Tokens created before this release have no such role and need Chainlink's manual registration.

6. Properties verified

- Launch sale bounds on the live pool (mainnet fork, SecondaryLeg.js; the tests assert the bounds and print the figures quoted here): a 9,000 USD raise with a 900 USD secondary share sells about 146 USD of PBTC before the 2% bound at current depth and the rest stays in the main position; with the oracle price 3% above the market the sale is the minimum, between 25 and 29 USD of PBTC (about 27.9), and returns at least 24 USD of WETH; with the oracle price 12% above the market nothing sells and the leg reverts SecondaryRouteOffPrice, which the factory skips without a record (FactoryLaunch.js).
- Pool opening against real Uniswap V3 (mainnet fork, LpOpen.js): a fresh pool is created at the launch price; empty pools pre-initialized at 2x, 0.5x, 100x, 0.01x and at the edges of the tick range are moved to the launch price by the one-wei swap and used; foreign liquidity across the launch price, a one-sided dust position between the foreign price and the launch price, and an extreme tick filled to the per-tick cap are all detected as unusable.
- Preempted attempt and candidate count: a preempted attempt is neither recorded nor paused (FactoryLaunch.js) and costs about 480,000 gas on the mainnet fork (reviewer measurement, not kept in the repository); a pool creation costs about 4,613,465 gas, so covering 64 candidates needs about 295M gas in the block of the launch, against a 40M block gas limit and a 16,777,216 transaction cap; the candidate set depends on the previous block's hash.
- Launch distribution and price (LaunchDistribution.js): for a fixed presale sold out and partly filled, a stepped presale at 1.5x and 4x sold out, just above its soft cap, and bought only in its last step (sold out and not), a sold-out bonding curve in two tiers, and a launch without presale, the tests assert that treasury, vesting, staking, buyers, both pools and the burn add up to the total supply, that the factory holds none of the token and no more PBTC than before, that the main pool opens at the presale price (fixed) or the end price (stepped, bonding) and above it only when a stepped presale raised more than its tokens are worth at the end price, that both pools hold tokens in proportion to the PBTC behind them, and that the stepped presale's getLaunchPrice equals the pool price.
- Gas of the heaviest launch shapes (mainnet fork, LaunchGas.js; the tests assert the bounds and print the figures quoted here). Launch attempts made by upkeep, each asserted with a margin for the stand-in contracts against the 14,500,000 stipend: plain fixed launch 10,856,419; fresh-token launch with Team Finance, presale staking and vesting 11,566,582 (PBTC pool prepared) and 11,571,131 (WETH pool prepared); stepped with both pools prepared 11,800,104. Creation plus launch without presale runs in the creator's own transaction and uses 11,924,831, asserted against the 16,777,216 transaction cap. UPKEEP_MIN_GAS (about 15.33M) is under the transaction cap.
- Observation rings: a Uniswap V3 pool writes at most one observation per second, so a ring with more slots than the window has seconds always holds an observation old enough for the window, which setWindow enforces and ringCoversWindow reports. The configured rings are 2,000 (PBTC/USDC), 2,056 (COAL/PBTC) and 6,700 (WETH/USDC) slots for a 1,800-second window; ring size never decreases. A reviewer measured the alternative on the mainnet fork: a 100-slot ring is filled by 100 dust swaps in distinct seconds inside one window (about 1 USD of fees, 12,922,808 gas) and every quote reverts OLD until the window elapses, while a ring of 1,801 slots kept answering after 1,801 writes in 1,801 seconds. The configured ring sizes are therefore the relevant fact.
- Presale math and state machine: a one-wei exact-output quote on each presale type is above zero, and the fixed presale charges exactly what it quoted; bonding buys charge ceil and sells pay floor of the same integral, 40 random rounds by two buyers keep the balance at least totalRaisedPbtc + creatorFeesAccumulated and a full unwind ends with totalTokensSold == 0 (ForgeRules.js); pro-rata refunds never exceed the pool (FactoryLaunch.js). Reviewer proofs of concept, not kept in the repository, add: stepped ladder prices average (start + end) / 2 and a multi-step buy matched a JavaScript replica to the wei; 60 random buys and sells by three accounts kept the balance equal to totalRaisedPbtc + creatorFeesAccumulated; the sell-out raise is path-independent within 1 wei; launched and failed are exclusive, refunds require failed and withdrawRaised is one-shot.
- Routers: exact-input buys refund the PBTC a capped buy did not use on the path, single-hop and ETH entry points; a presale that under-reports or fabricates its result takes no PBTC from the router; paths that do not end in PBTC or are not made of whole hops are rejected; expired trades and swap legs below the caller's minimum revert; an exact-output sell through the router delivers exactly the requested output, takes only the tokens the swap needs and leaves the router empty (PresaleRouter.js, ForgeRules.js). In a reviewer run on the mainnet fork, not kept in the repository, 100 exact-output buy samples cleared with no shortfall.
- Staking invariants: the repository's CoalStaking.js runs 120 random stake, withdraw and claim actions by four users with gaps of up to two days and checks that what was minted never exceeds the budget and reaches 99.9% of it; a separate two-staker test checks that getReward pays what pendingRewardsExact showed within two seconds of emission. Reviewer proofs of concept, not kept in the repository, add: PBTCForgeStaking randomized sequences (160 steps, 5 users, both modes) never paid more than totalRewards, every withdrawal and claim succeeded, and after exits and burnUnallocated

the pool held under 1e7 wei of dust; CoalStaking randomized runs (220 steps, 6 users, idle gaps up to 40 days, 1,100 to 1,160 simulated days) minted exactly the emission in force while someone was staked, pendingRewardsExact matched getReward to 1e-4 relative, boost steps were credited retroactively and totalWeighted returned to zero.

- Oracle precision on the live pools (mainnet fork, SecondaryLeg.js): a fresh deployment of the oracle against the live PBTC/USDC and COAL/PBTC pools returns 18-decimal quotes whose 6-decimal projections equal the 6-decimal quotes to the unit, with the fractional part the consumers divide by present (598.47 against 598 micro-USD for PBTC).
- Failure clock (ForgeRules.js): a bonding curve that leaves readiness between two failed attempts has its failure record cleared and the 14-day clock restarts with the next failure; the clone hooks emit Finalized and ClaimingEnabled in the launch transaction.
- CoalStaking upgrade rehearsal (mainnet fork, CoalStakingUpgrade.js): the live proxy is upgraded to the reviewed implementation through the real ProxyAdmin from its owner Safe; every plain storage slot and the Pausable namespace slot hold the same value before and after, the consumed reinitializer reverts InvalidInitialization for the owner, the pending rewards of up to six real stakers change by less than one minute of emission across the upgrade, each claim mints what the view showed one second earlier plus at most one second of emission, the claims of the following day stay under that day's budget, a full withdrawal returns exactly the stake, a new staker is paid its weighted share of a day to within 2 percent, and two bounded rollEpochs calls by a third party after 200 idle days change the pending amount by no more than three seconds of emission.
- CoalStaking over the whole schedule (CoalStakingSchedule.js): one staker from launch to the end, others entering in years 1, 5 and 8 and one leaving again; the boost reads 1.0x until day 15, one sixth more every 15 days and 2.0x from day 90; a 30-day window with two stakers at 2.0x pays out the window's budget split by stake; at every checkpoint through year 9 the COAL minted plus the pending amounts equal the schedule's budget since the first stake to within two seconds of emission and never exceed it; all of the 80M but a rounding remainder of under 40 COAL is paid by the end of year 10 and nothing after it; the 20M premint plus the schedule equals COAL's 100M cap, so the cap never blocks a claim; everyone withdraws their exact stake at the end and the weights return to zero.
- CoalStaking after a long gap (CoalStaking.js): 300 idle days are closed by anyone in three calls of 100 epochs each, the first of which advances the ledger by exactly 100 epochs, after which the staker's claim is at most the budget for the period and above 99.9% of it.
- Clone initialization: every implementation calls _disableInitializers() in its constructor and reverts InvalidInitialization on initialize (asserted in ForgeRules.js for CoalStaking and the oracle; the reviewers checked the token, staking and vesting implementations); clones are created and initialized in one factory transaction; token clones use CREATE2 from the factory.
- Storage layout: factory (slots 0 to 66; the reserved gap ends at the same slot as on the live proxy), oracle (WETH and wethUsdcPool in slots 5 and 6, which the live proxy holds empty) and CoalStaking (slots 0 to 27 and the gap slot for slot, the Pausable namespace declared, stepSecondsV2 15 days, epochSecondsV2 1 day) match the raw storage of the live proxies and the upgrades manifest. The upgrade scripts run the OpenZeppelin storage check against the live implementation, and it accepts all three upgrades on a mainnet fork.
- Token CCIP admin (TokenCcipAdmin.js): the creator is the admin from creation, also when a router pays for the project; only the current admin can hand the role on or clear it, and a cleared role stays empty, also against a second initialize; a token replaced at launch takes over the admin of the old one, cleared and handed-on included, and a token from before the role existed is replaced by one with the creator as admin; the token's function list is ERC-20, burn, permit and these two functions. On the mainnet fork the creator registers a token through Chainlink's RegistryModuleOwnerCustom and becomes its administrator in the TokenAdminRegistry; nobody else can, and nobody can once the role was cleared.
- Repository suite: 114 local tests pass; 22 tests run on the mainnet fork and pass.

7. Recommendations

1. Place a timelock in front of the ProxyAdmin, factory owner, oracle owner and COAL admin, and monitor the admin events (M-01).
2. Deepen PBTC/USDC liquidity and bound the oracle consumers that gate launches (M-02, L-03).
3. Let creators direct LP positions and fees to a chosen address (L-02); present pre-launch token addresses as provisional (L-01).
4. Before withdrawing fees, compare the factory's PBTC balance with affiliateOwedTotal; the contract enforces it, the operations should expect it (I-03).
5. Operations: watch ProjectFinalizationFailed, ProjectEvaluationSkipped and TeamFinanceLockFailed

8. Appendix

8.1 Verification artifacts

Artifact	Content
Project test suite	114 local tests (CoalStaking.js, CoalStakingSchedule.js, FactoryLaunch.js, LaunchDistribution.js, LaunchPaths.js, PresaleRouter.js, ForgeRules.js, TokenCcipAdmin.js) and 22 mainnet-fork tests (LaunchGas.js, LpOpen.js, SecondaryLeg.js, CoalStakingUpgrade.js, TokenCcipAdmin.js, run with FORK_BASE=1 FORK_BLOCK=55300000)
Reviewer proofs of concept	46 tests, 11 on the mainnet fork: owner library takeover, preempted candidates, secondary bound, mainnet-fork exact-output buys, oracle ring and TWAP cost measurements, staking randomizations, token and vesting checks, live reads of roles, pools, proxies and projects
Storage layout comparison	Raw slot reads of the three live proxies against the reviewed sources and the OpenZeppelin manifest
Bytecode sizes	HelperLib 21,846; Factory 19,770; PresaleRouter 19,004; BondingCurve 14,096; SteppedPrice 12,371; LiquidityLib 11,165; FixedPrice 9,870; Oracle 9,507; CoalStaking 9,272; Token 6,693 bytes

8.2 Project status on the factory

Ten projects exist, all launched or failed, none pending.

Projects 6 and 10 carry a usdcHeldForLock record of about 139 USDC each although their positions are in Team Finance and the factory's USDC balance is zero; no code path reads the field after launch, so the value has no effect.

Project id	Presale	State	Main LP position
1	none	launched	claimed by the creator
2	fixed	launched	claimed by the creator
3	fixed	failed, refunds open	
4	fixed	failed, refunds open	
5	fixed	launched	Team Finance
6	none	launched	Team Finance
7	fixed	failed, refunds open	
8	fixed	failed, refunds open	
9	fixed	failed, refunds open	
10	fixed	launched	Team Finance

8.3 Disclosure

Security reports are received through the contact published on the website.