CLAWDIT

Finding the void in your code

Security Audit Report

Bonker Protocol — Uniswap v4 Token Factory on Base

Bonker (bonker.wtf)

Version: 1.1

Date: July 1, 2026

Lead Auditor: CLAWDIT

Report ID: CLAWDIT-2026-004

Table of Contents

1. Executive Summary 3
2. Scope & Methodology 4
3. Findings Summary 5
4. Detailed Findings 6
5. Gas Optimizations 13
6. Recommendations 14
7. Disclaimer 15

1. Executive Summary

Bonker is a Base-deployed fork of Clanker's Uniswap v4 token-factory stack: a factory (Bonker) that deploys ERC-20s and single-sided v4 liquidity behind dynamic/static-fee hooks, an LP fee-conversion locker, a sniper-auction MEV module, and launch-time extensions (vault, airdrop, presale, dev-buy). CLAWDIT reviewed the full live production contract set with static analysis (Aderyn, Slither, Foundry lint) backed by full manual line-by-line review, focused on swap-time fund safety and abuse resistance for a Uniswap hook-allowlist submission. No critical or high-severity vulnerabilities were identified. The static analyzers' High-severity detectors (locked-ether, reentrancy-after-external-call, contract-name reuse) were triaged and found to be false positives or non-exploitable under the existing ReentrancyGuard, onlyFactory, and owner-trust controls. The protocol-fee delta math is correct across all four swap shapes, hooks are immutable and per-pool isolated (PoolId-keyed), and value flows follow checks-effects-interactions with SafeERC20 and ReentrancyGuard. The material findings are one medium-severity MEV/slippage issue on protocol-owned fee-conversion swaps and several low/informational items (dev-buy slippage, sniper-window swap reverts, deterministic-address griefing, broad owner emergency powers, unchecked raw approvals, floating pragma, and hardening notes). Bounding slippage on internal swaps and formalizing the owner-trust model would meaningfully strengthen the posture.

0	0	1	6	7
CRITICAL	HIGH	MEDIUM	LOW	INFO

Audit Details

Client	Bonker (bonker.wtf)
Project	Bonker Protocol — Uniswap v4 Token Factory on Base
Chain	Base (chainId 8453)
Audit Date	July 1, 2026
Lead Auditor	CLAWDIT

2. Scope & Methodology

Contracts in Scope

Bonker (Factory)
`0xD850DACe6c3E3B3cf09ABb92342Fab681013c8cB`
Chain: Base | 371 LOC

BonkerHookV2 (abstract base hook)
`Inherited by DynamicHook / StaticHook`
Chain: Base | 623 LOC

BonkerHookDynamicFeeV2 (DynamicHook)
`0x963E91A45148b39737b9DF10c5b897B55cA9e8cC`
Chain: Base | 244 LOC

BonkerHookStaticFeeV2 (StaticHook)
`0xC9156C1868E122eF5b3e6ed946e1E88ff7da68Cc`
Chain: Base | 51 LOC

BonkerLpLockerFeeConversion (LpLocker)
`0xBf05b1d5E356f3219D0086A4e09c969ADbe2e7d0`
Chain: Base | 793 LOC

BonkerSniperAuctionV2 (MevModule)
`0x6a04057180F8cc02E18DabEE3f3437E438BE657A`
Chain: Base | 471 LOC

BonkerFeeLocker (FeeLocker)
`0x473e52D89bE6ea78f94d1b5c62Bd1f01b1E32e21`
Chain: Base | 62 LOC

BonkerPoolExtensionAllowlist
`0x00d9C6dda8D6DC9fc735B8a10a62AA6AE070510C`
Chain: Base | 20 LOC

BonkerToken (per-launch template)
`CREATE2 per deployment (Deployer 0x140c9eb85b4204cfcc8e345ce7894edb315a3043)`

Chain: Base | 213 LOC

BonkerDeployer (library)
`0x140c9eb85b4204cfcc8e345ce7894edb315a3043`
Chain: Base | 29 LOC

BonkerUniv4EthDevBuy (DevBuy)
`0xc00Ab3631E82902f55B62EB95A0101eE2eb91a69`
Chain: Base | 244 LOC

BonkerVault (Vault)
`0x26a4654E85CD8cc3Ba08dBC05418c63300624c8e`
Chain: Base | 163 LOC

BonkerAirdropV2 (Airdrop)
`0xa727da00eDd0F98Dc5Fb5D3b9eA09646CB809A87`
Chain: Base | 269 LOC

BonkerPresaleEthToCreator (Presale)
`0x60729328cF4fd3E6996dee350308EcB238FCa5D7`
Chain: Base | 597 LOC

BonkerPresaleAllowlist
`0x824a424808F9a20a0bBf831634e2f460D8db256B`
Chain: Base | 133 LOC

MEV modules (DescendingFees / BlockDelay / TimeDelay) + SniperUtilV2 + OwnerAdmins
`In-repo allowlistable modules + shared base`
Chain: Base | 574 LOC

Repository

`Commit: 8bfaf1e`
https://github.com/bonker-wtf/contracts

Methodology

- Full-protocol manual line-by-line review of the live production contract set (~5,300 nSLOC across src/)
- Static analysis with Aderyn v0.6.8 (Cyfrin) — 4 High / 18 Low detectors triaged against manual review
- Static analysis with Slither v0.11.5 (attempted; detector coverage limited by the project's required via-IR build)
- Foundry compilation and linting (forge build --via-ir, forge lint)

- Uniswap v4 flash-accounting / settlement correctness analysis (unlock context, ERC-6909 claims, take/settle/mint/burn/sync)
- Protocol-fee delta / sign / rounding re-derivation across all four swap shapes
- Reentrancy & callback-ordering analysis across the swap -> hookFeeClaim -> lpLocker -> mevModule -> poolExtension call graph
- Access-control & owner-power matrix
- Permissionless-pool (initializePoolOpen) and hostile-paired-token abuse analysis
- Clanker-fork divergence review
- Vesting / presale / airdrop accounting review
- CEI / non-standard-ERC20 safety review

Tools Used

- Aderyn v0.6.8 (Cyfrin static analyzer)
- Slither v0.11.5
- Foundry (forge build --via-ir, forge lint)
- Manual review
- Uniswap v4-core / v4-periphery source cross-reference

3. Findings Summary

ID	TITLE	SEVERITY	STATUS
BNK-01	LP fee-conversion swap-backs execute with no slippage protection (MEV-extractable)	Medium	Acknowledged
BNK-02	Creator dev-buy final swap uses amountOutMinimum = 1	Low	Acknowledged
BNK-03	Sniper-auction window causes standard interface swaps to revert	Low	Acknowledged
BNK-04	Deterministic token-address squatting / cross-chain deploy DoS	Low	Acknowledged
BNK-05	Broad owner emergency powers and full-balance sweeps	Low	Acknowledged
BNK-06	Factory token approvals use raw approve() with unchecked return	Low	Acknowledged
BNK-07	Floating compiler pragma	Low	Acknowledged
BNK-08	protocolFee is a single shared storage slot across all pools	Informational	Acknowledged
BNK-09	Permissionless open pools accept arbitrary paired tokens	Informational	Acknowledged
BNK-10	Component allowlisting relies on supportsInterface self-attestation + owner trust	Informational	Acknowledged
BNK-11	Presale token-claim rounding leaves dust stranded	Informational	Acknowledged
BNK-12	Pool-extension afterSwap failures are silently swallowed	Informational	Acknowledged
BNK-13	Missing zero-address validation on recipient/admin setters	Informational	Acknowledged
BNK-14	nonReentrant is not the first modifier	Informational	Acknowledged

4. Detailed Findings

BNK-01 LP fee-conversion swap-backs execute with no slippage protection (MEV-extractable)

Medium | Acknowledged

`BonkerLpLockerFeeConversion.sol:594-679 (_uniSwapUnlocked / _uniSwapLocked)`

DESCRIPTION

When the hook claims LP fees during a user swap (_beforeSwap -> _lpLockerFeeClaim -> collectRewardsWithoutUnlock), fees owed to recipients who elected currency conversion are swapped back through the same pool. Both swap paths run with no minimum-output protection: _uniSwapUnlocked sets sqrtPriceLimitX96 to the extreme tick and takes whatever the pool returns, and _uniSwapLocked passes amountOutMinimum: 0. Because the conversion executes inside the user's swap transaction, it is trivially sandwichable.

IMPACT

An MEV searcher can sandwich any user swap that triggers a fee conversion and extract value from the reward recipients' accrued LP fees. Losses are bounded by the per-swap fee amount but recur on every conversion, eroding creator/recipient fee income. Principal liquidity and pool solvency are unaffected.

VULNERABLE CODE

```
// _uniSwapLocked
amountOutMinimum: 0, // minimum amount we expect to receive
```

RECOMMENDATION

Enforce a minimum output on both conversion paths derived from the pool's pre-swap sqrtPrice (a bounded deviation) or a short TWAP, and/or bound the conversion swap's price impact. Alternatively make conversion opt-in per collection with a caller-supplied slippage parameter.

BNK-02 Creator dev-buy final swap uses amountOutMinimum = 1

Low | Acknowledged

`BonkerUniv4EthDevBuy.sol:176 (_performDevBuy)`

DESCRIPTION

The dev-buy converts launch ETH into the freshly deployed token via the Universal Router with the final pairedToken->token leg hardcoded to amountOutMinimum: 1. (The intermediate WETH->pairedToken leg does honor a caller-supplied minimum.) The buy runs atomically during deployToken at the creator-chosen starting tick.

IMPACT

If the deployment transaction is visible in the public mempool, a searcher can sandwich the dev-buy and extract value from the creator's ETH. Exposure is limited to the creator's own dev-buy amount and to launches submitted without private-mempool protection.

RECOMMENDATION

Accept and enforce a caller-supplied amountOutMinimum on the final buy leg (as already done for the intermediate leg), or recommend creators submit launches via a private mempool / bundle.

BNK-03 Sniper-auction window causes standard interface swaps to revert

Low | Acknowledged

`BonkerSniperAuctionV2.sol:279-300, 426-465 (_pullPayment / beforeSwap)`

DESCRIPTION

During the sniper-protection window (up to MAX_MEV_MODULE_DELAY = 2 minutes) beforeSwap reverts NotAuctionBlock before the first auction block, and on the exact auction block _pullPayment abi.decode(auctionData,(address)) and safeTransferFrom(WETH, payee) revert for any swap that does not carry a valid encoded payer with a WETH approval. A standard Uniswap-interface swap (empty hookData) therefore reverts during this window.

IMPACT

Intended anti-sniper behavior, but swaps routed through a generic interface or router revert for the first blocks after launch. Relevant to Uniswap hook-allowlist UX expectations.

RECOMMENDATION

Document the auction window for integrators and expose a live 'MEV active' read (e.g. mevModuleEnabled) so front-ends can warn users or defer swaps until the window ends.

BNK-04 Deterministic token-address squatting / cross-chain deploy DoS

Low | Acknowledged

`BonkerDeployer.sol:15-17; Bonker.sol:155-161 (deployTokenZeroSupply)`

DESCRIPTION

Tokens are deployed with CREATE2 salt = keccak256(tokenAdmin, salt). The full init code (which embeds admin/name/symbol/metadata/originatingChainId) determines the address, so an attacker who replays a victim's public deployment parameters on a destination chain can pre-occupy the deterministic address, making the victim's deployTokenZeroSupply / bridge-mint deployment revert on address collision.

IMPACT

Griefing / denial of service of cross-chain deterministic deployments. No funds are stolen and the squatted contract carries the victim's own admin and parameters.

RECOMMENDATION

Incorporate a deployer-controlled nonce or msg.sender into the CREATE2 salt, or execute cross-chain deploys promptly via a trusted relayer.

BNK-05 Broad owner emergency powers and full-balance sweeps

Low | Acknowledged

`BonkerLpLockerFeeConversion.sol:775-786 (withdrawETH/withdrawERC20); Bonker.sol:81-87 (claimTeamFees)`

DESCRIPTION

The LP locker owner can withdraw the entire ETH and any ERC-20 balance, and the factory owner/admin can sweep the factory's entire balance of any token via claimTeamFees. Locked LP positions (ERC-721) are not withdrawable and the locker holds only transient fee balances during normal operation, so principal liquidity is protected; nonetheless these are meaningful trusted-owner powers.

IMPACT

A compromised or malicious owner could seize stranded/transient token balances and re-point fee flows. The hooks themselves are immutable and non-upgradeable, so live pools cannot be re-pointed or bricked by the owner.

RECOMMENDATION

Hold owner/admin roles behind a multisig + timelock, publish the owner addresses, and document the exact scope of emergency powers for allowlist reviewers.

BNK-06 Factory token approvals use raw approve() with unchecked return

Low | Acknowledged

`Bonker.sol:289, 357 (_initializeLiquidity / _triggerExtensions)`

DESCRIPTION

The factory grants token allowances with raw IERC20.approve() and ignores the boolean return value, while the rest of the codebase consistently uses SafeERC20. The approved token is the freshly deployed BonkerToken (standard, returns true), so this is presently safe, but the path is not future-proof against a non-standard token. Surfaced by static analysis (Aderyn L-12/L-13).

IMPACT

No impact for the standard BonkerToken; a non-standard token in this path that returns false or reverts atypically on approve could break deployment or leave an unexpected allowance.

RECOMMENDATION

Use SafeERC20.forceApprove for the locker and extension approvals, consistent with the rest of the codebase.

BNK-07 Floating compiler pragma

Low | Acknowledged

`All src/*.sol (e.g. Bonker.sol:2) — pragma solidity ^0.8.28`

DESCRIPTION

Every contract declares a floating pragma (^0.8.28). Deployed bytecode should be pinned to the exact audited compiler version to avoid compilation under an unaudited 0.8.x release. Surfaced by static analysis (Aderyn L-14).

IMPACT

Contracts could be compiled with a different, unaudited 0.8.x compiler, risking compiler-version-specific behavior in future deployments.

RECOMMENDATION

Pin the pragma to solidity 0.8.28 across the suite.

BNK-08 protocolFee is a single shared storage slot across all pools

Informational | Acknowledged

`BonkerHookV2.sol:54, 100-102, 443-494`

DESCRIPTION

protocolFee is a contract-level storage variable (not per-pool) re-derived on every swap by _setFee before the fee delta math. Correctness relies on the invariant that _setProtocolFee is always paired with updateDynamicLPFee and that nested same-pool swaps (locker swap-back, dynamic-fee simulateSwap) re-establish it. The review confirmed the invariant holds in the current code, and cross-pool fees commingle only into the common factory beneficiary (benign).

IMPACT

No exploit identified. Flagged because a future change that mints protocol-fee deltas without first calling _setFee, or that interleaves another pool's swap between a pool's before/after callbacks, could corrupt fee accounting.

RECOMMENDATION

Consider keying protocolFee by PoolId for defense-in-depth, or add an assertion/test that _setFee ran for the current pool before minting fee deltas.

BNK-09 Permissionless open pools accept arbitrary paired tokens

Informational | Acknowledged

`BonkerHookV2.sol:168-197 (initializePoolOpen)`

DESCRIPTION

initializePoolOpen lets anyone bind an arbitrary paired token to the hook (weth-as-bonker and pool extensions are rejected). A hostile paired ERC-20 (reverting/reentrant/fee-on-transfer) can make swaps or the protocol-fee take() on its own pool fail, but per-pool state is isolated by PoolId and the shared hook is not bricked for other pools.

IMPACT

Self-inflicted denial of service limited to the attacker's own open pool; no impact on factory pools or other users.

RECOMMENDATION

Acceptable as designed. Front-ends and indexers should treat open-pool tokens as untrusted and not surface them as Bonker-verified.

BNK-10 Component allowlisting relies on supportsInterface self-attestation + owner trust

Informational | Acknowledged

`Bonker.sol:98-150 (setHook/setLocker/setMevModule/setExtension)`

DESCRIPTION

Each setter gates only on an ERC-165 supportsInterface self-attestation before enabling a component; the safety of the component is a matter of owner discretion. An owner-allowlisted hook, locker, MEV module, or extension is fully trusted within deployToken and the swap path. (Aderyn's reentrancy-after-external-call High flags on these setters were reviewed as non-exploitable: they are owner-gated and set only a boolean mapping.)

IMPACT

The security of launched pools depends on the owner enabling only audited components. Not exploitable by third parties.

RECOMMENDATION

Maintain a published, change-controlled list of enabled components and gate any new component on an audit.

BNK-11 Presale token-claim rounding leaves dust stranded

Informational | Acknowledged

`BonkerPresaleEthToCreator.sol:457, 484 (claimTokens)`

DESCRIPTION

Per-user token amounts are computed as tokenSupply * ethBuyIn / ethRaised with integer division, so a negligible amount of token dust can remain unclaimable in the presale contract after all contributors claim.

IMPACT

Immaterial value permanently stranded; no contributor is over- or under-paid beyond rounding.

RECOMMENDATION

Optionally sweep residual dust to the presale owner after full distribution, or accept as negligible.

BNK-12 Pool-extension afterSwap failures are silently swallowed

Informational | Acknowledged

`BonkerHookV2.sol:350-398 (_runPoolExtension / _runPoolExtensionHelper)`

DESCRIPTION

_runPoolExtension calls the extension's afterSwap via an external self-call wrapped in try/catch and emits PoolExtensionFailed on revert without failing the user swap. This is intentional isolation but means extension execution is best-effort.

IMPACT

No swap-safety impact; extensions that must run reliably require off-chain monitoring of the PoolExtensionFailed event.

RECOMMENDATION

Document the best-effort semantics for any future pool-extension integrator.

BNK-13 Missing zero-address validation on recipient/admin setters

Informational | Acknowledged

`Bonker.sol:73-77 (setTeamFeeRecipient); BonkerPresaleEthToCreator.sol:137-141 (setBonkerFeeRecipient) and 402-429 (withdrawFromPresale recipient); BonkerToken.sol:79-86 (updateAdmin)`

DESCRIPTION

Several privileged setters and the presale withdraw recipient accept an address argument without a zero-address check. A misconfigured fee recipient could route fees to address(0), and a contributor could burn their own refund by passing recipient = address(0). Surfaced by static analysis (Aderyn H-2/L-9).

IMPACT

Owner/user footgun leading to lost funds on misconfiguration; not third-party exploitable.

RECOMMENDATION

Add require(addr != address(0)) guards on recipient and admin setters.

BNK-14 nonReentrant is not the first modifier

Informational | Acknowledged

`BonkerLpLockerFeeConversion.sol:108, 775; BonkerPresaleEthToCreator.sol:353, 406`

DESCRIPTION

Several functions place nonReentrant after other modifiers (e.g. onlyFactory nonReentrant). The preceding modifiers perform no external calls, so guard effectiveness is unaffected today, but placing nonReentrant first is the defensive convention. Surfaced by static analysis (Aderyn L-5).

IMPACT

No exploit; hardening / best-practice only.

RECOMMENDATION

Order nonReentrant as the first modifier on guarded functions.

5. Gas Optimizations

GAS-01 Residual ERC-20 approvals after extension trigger

`Bonker.sol:357 (_triggerExtensions)`

The factory approves each extension for exactly extensionSupply before calling receiveTokens. If an extension pulls less than approved, a residual allowance persists. Zero out the allowance after the call or rely on exact pulls.

Estimated Savings: Hygiene / allowance-safety

GAS-02 Repeated storage struct reads in locker fee handling

`BonkerLpLockerFeeConversion.sol:422-554 (_handleFees)`

_tokenRewards[token] and feePreferences[token] are read from storage multiple times per collection. Cache into memory once to save repeated SLOADs across the distribution loops.

Estimated Savings: Multiple SLOADs per fee collection

GAS-03 Auction gas-peg exponentiation can overflow for large block gaps

`BonkerSniperAuctionV2.sol:263-272 (_getBaseAuctionGasPeg)`

block.basefee * 1125**n / 1000**n reverts (overflow) for large owner-configured block gaps and recomputes the multiplier each call. Bound n or precompute the 1125^n/1000^n factor.

Estimated Savings: Avoids overflow revert on misconfiguration; minor gas

6. Recommendations

1. Add slippage / price-impact bounds to all protocol-owned internal swaps (LP fee conversion and the dev-buy final leg).
2. Place owner/admin roles behind a multisig + timelock and publish the scope of emergency powers for allowlist reviewers.
3. Publish and change-control the set of factory-enabled hooks, lockers, MEV modules, and extensions; gate new components on an audit.
4. Pin the compiler pragma to 0.8.28, use SafeERC20.forceApprove for the factory's token approvals, and add zero-address guards on fee-recipient and admin setters.
5. Add Foundry invariant/fuzz tests asserting fee conservation and delta reconciliation across all four swap shapes and nested locker swap-backs.
6. Document the <=2-minute sniper-auction window behavior for interface/router integrators and expose a live 'MEV active' read.
7. Consider PoolId-keyed protocolFee for defense-in-depth against future hook changes.

Code Quality Assessment

8/10

- Consistent checks-effects-interactions ordering with OpenZeppelin ReentrancyGuard on state-changing entrypoints.
- SafeERC20 used throughout (two raw factory approvals excepted, BNK-06), with balance-delta accounting that tolerates fee-on-transfer and non-standard ERC-20s (FeeLocker.storeFees, locker fee bring-in).
- Hooks are immutable and non-upgradeable and per-pool state is PoolId-keyed, so the factory owner cannot re-point or brick live pools.
- Protocol-fee delta math was re-derived across all four swap shapes (exact-in/out x for/against the launched token) and found sign- and currency-correct.
- Static analysis (Aderyn v0.6.8, Slither v0.11.5, forge lint) surfaced only hygiene-level true positives (unsafe approve, floating pragma, missing zero-address checks); the High-severity detectors (locked-ether, reentrancy-after-external-call, contract-name reuse) were reviewed and dismissed as false positives or non-exploitable under existing guards.
- Complexity concentrates in the LP-locker fee-conversion and dynamic-fee simulateSwap paths, and the system relies heavily on trusted owner/admin roles for component allowlisting and emergency withdrawals.

7. Disclaimer

Important Notice

This audit report is provided "as is" and for informational purposes only. CLAWDIT makes no warranties, express or implied, regarding the findings, recommendations, or any other information provided in this report. The audit is not a guarantee of the absence of bugs or vulnerabilities. Smart contracts are inherently risky, and users should always do their own research before interacting with any smart contract.

CLAWDIT

Finding the void in your code

clawdit.xyz