PRODUCT TEARDOWN · NO. 01

2,263 crowd sales, 55 graduations, nobody made money

pools.trade teardown — Uniswap's 14 days running its own launchpad

Uncle Onchain · Liquidity to Alpha

Product teardown · No. 01

Data as of 2026-08-18

Uniswap's token launch and trading platform on Robinhood Chain, 14 days after launch (2026-08-05 → 08-18). This report uses on-chain data, verified contract source, the front-end JS bundle and the official whitepaper as primary evidence to take apart its mechanism design, actual operation, and product strengths and weaknesses. Every key parameter can be traced to a transaction hash, contract address or source line.

2.43%
CROWD LAUNCH GRADUATION RATE (55 / 2,263, FULL ON-CHAIN SCAN)

−98.3%
DROP IN DAILY CROWD LAUNCH CREATIONS (8/5 → 8/16)

0.2899%
ACTUAL TOTAL FEE PER TRADE (LP 0.25% + PROTOCOL 0.04%)

$20
MEDIAN BID SIZE ACROSS THE PLATFORM (GRADUATING NEEDS ~$5,000)

−71%
AVERAGE PAPER LOSS VS CLEARING PRICE FOR PARTICIPANTS IN

13.55 ETH
ACCUMULATED IN THE TWO FEE VAULTS (~$25.7K)

	GRADUATED TOKENS (5/5 LOSING)	

Contents

0. Origin: the SOL nobody can take out

1. Demand: what is Uniswap actually solving

2. Mechanism: full rules and on-chain parameters

3. User journeys: three roles, as they really go

4. Operation: 14 days of data

5. Competitors

6. Risks and opportunities

7. My four judgments

8. Verdict on a three-layer experiment

Appendix: parameters, definitions, method

0. Origin: the SOL nobody can take out

While designing an IDO fundraising module, I arrived at an uncomfortable conclusion.

The rules are simple: 100 SOL cap per sale, 5 SOL per address; the money raised buys straight on the bonding curve, and then everyone receives tokens **at the latest price after the buying is done**. That way everyone's cost equals the opening price and nobody can arbitrage the spread — if you distributed at the average price, selling immediately would be risk-free profit.

The cost is a leftover chunk of tokens that never lands in anyone's pocket. So when everyone sells together, the SOL they get back is always less than the 100 SOL they put in, and the difference stays in the pool forever. The platform didn't take it, users didn't, and the project can't either.

Me Who does that unrecoverable SOL serve?

Dev It props up pool depth, so the chart is harder to dump through.

Me So the beneficiary is the chart, not people. And on the day it goes to zero it gets swapped out — by the last sellers, not by the people who put the money in.

Dev By that logic, the long-term beneficiaries are the chain, and whoever runs fastest.

A crowd of retail buyers pooled money to make a permanent donation to a chain's pool, and subsidized the fastest runners along the way.

I wanted to know how others handle this, and found pools.trade — Uniswap itself, on a new chain, with liquidity explicitly permanently locked. The first answer is in §2.9: it isn't a design flaw, it's a curve you can draw precisely. The second answer is in §7.4.

> How to read this: chapters 1–6 are mechanism and on-chain data, written as far as possible only from what transaction hashes can prove; chapter 7 is my four judgments as a product manager. If you don't want opinions, skip straight to chapter 8.

1. Demand: what is Uniswap actually solving

1.1 The stated problem: three structural flaws of bonding curves

The CCA whitepaper (`whitepaper_cca.pdf`, 2025-11, authors including Hayden Adams and Ciamac Moallemi) states the goal plainly, in just two points: **price discovery** — "determining a fair, credible market price under uncertainty and information asymmetry"; **liquidity bootstrapping** — "ensuring enough capital forms to support active, stable trading while price stays tied to the latest information".

Its critique of existing designs is where the demand comes from. On pump.fun-style bonding curves: "Its core fragility is one-shot execution: a buyer's transaction executes immediately against the current AMM state, making the outcome highly dependent on transaction timing, mempool ordering and execution latency, and thus exposed to MEV attacks such as front-running and sandwiching." On traditional uniform-price / Dutch auctions: "Because the clearing price depends on total demand, bidders have no incentive to bid early, so participation concentrates at the last moment and turns into sniping on-chain."

Put the two critiques together and you get CCA's design constraints: **(a) no one-shot fills**, or it becomes a race of reflexes; **(b) no settlement only at the end**, or it becomes a race for the last second. The answer is "continuous clearing" — slice supply into tens of thousands of blocks released step by step, fill a small slice at a uniform price in each block, and give money that arrives early a price advantage.

1.2 The unstated demand: cold-starting a new chain and owning the issuance entry point

Reading only the whitepaper underestimates this product. Robinhood Chain is an Arbitrum Orbit chain launched in 2026 (chainId 4663, ~100ms blocks). For Uniswap, pools.trade solves several more commercial problems at once:

WHO	WANTS	HOW POOLS.TRADE DELIVERS
Uniswap protocol	Become the default AMM on the new chain; lock in the first entry point for issuance and the first trade	Every token must migrate into a Uniswap v4 pool after graduating (PoolManager `0x8366a3…`); LP is permanently locked and can't be moved
Uniswap front end / wallet / API	Order flow and distribution	Launch, subscription and secondary trading all happen inside pools.trade; tokens appear automatically in Uniswap-ecosystem aggregators and lists
Robinhood Chain	TVL, addresses, daily trades	10,506 tokens created on day one (DeFiLlama basis; Pons 7,210 over the same period)
Project issuers	Free launch + seed money + sustainable fee income	No platform fee; a successful Crowd sale delivers the raised ETH; an optional creator fee pays out long term
Retail buyers	Not sniped, able to buy, refunded on failure	Uniform clearing price, 4-hour window, full refund if it doesn't graduate

> Note: the "**no launchpad fees**" that the official blog (blog.uniswap.org, 2026-08-05) keeps stressing only holds for issuance. Measured on-chain, every

> pools.trade v4 pool carries `protocolFee = 1638800` (400 pips = 0.04% on both buys and sells), set uniformly by `V4FeeAdapter 0x6d0009…` and collected into TokenJar `0x2aC03e…`. Uniswap does charge on trading (see §2.8).

1.3 Target users

Working backwards from the mechanism's thresholds, the product actually serves six kinds of people, and only the first two are really looked after:

PROFILE	CORE NEED	MECHANISM FIT
Projects with a community	Fair launch + one-off fundraise	High Graduates as long as it can raise ≥$5K of ETH subscriptions
Crypto-native early participants	A cost below the clearing price	High The earlier you enter, the lower your average (quantified in §4.5)
Quant / script players	Rule-based arbitrage	High But the edge shifts from "fast fingers" to "accurate maths + deep pockets"
Traditional stock users (Robinhood's customers)	Understand it, get filled	Low 4-hour window, maxPrice invisible, refunds need self-signing (§3.4)
Anonymous launchers with no audience	Launch and someone buys	Very low 97.6% of sales fail to raise $5K
RWA / regulated asset issuers	Auditable price discovery	Prohibited The front-end disclaimer says explicitly: "Pools may not be used for assets other than meme coins"

2. Mechanism: full rules and on-chain parameters

2.1 Two product lines

pools.trade has only two launch modes, chosen in the creation dialog. Both share the same token factory (`TokenFactory 0x000000e2…`) and the same LiquidityLauncher

(`0x0000FffFBE8e…` v3.2.0); the difference is "how the price is set".

	INSTANT LAUNCH	CROWD LAUNCH (CCA)
Pricing	Classic bonding curve, tradeable immediately	Continuous clearing auction; tradeable only after the 4-hour window
Starting valuation	Fixed `initialTick = 198,050` → 2.507 ETH FDV ≈ **$4,760**	Floor FDV ≈ 0.53 ETH ≈ **$1,000** (at the ETH price at creation)
Total supply	1,000,000,000 (all on the curve)	1,000,000,000, of which **500,000,000 auctioned**, 500,000,000 reserved for LP
Threshold	None	Raise ≥ 5× floor FDV (measured 2.6405 ETH ≈ $5,000, i.e. a $10K graduation FDV), otherwise full refund
Can it fail	No "failure", just no buyers	Yes: below threshold means refund, and the token becomes a pool-less zombie
Anti-sniping	None (creator and buyers compete in the same block)	Yes: per-block uniform price + early-entry price advantage
Actual usage (8/16, one day)	**746**	**23**

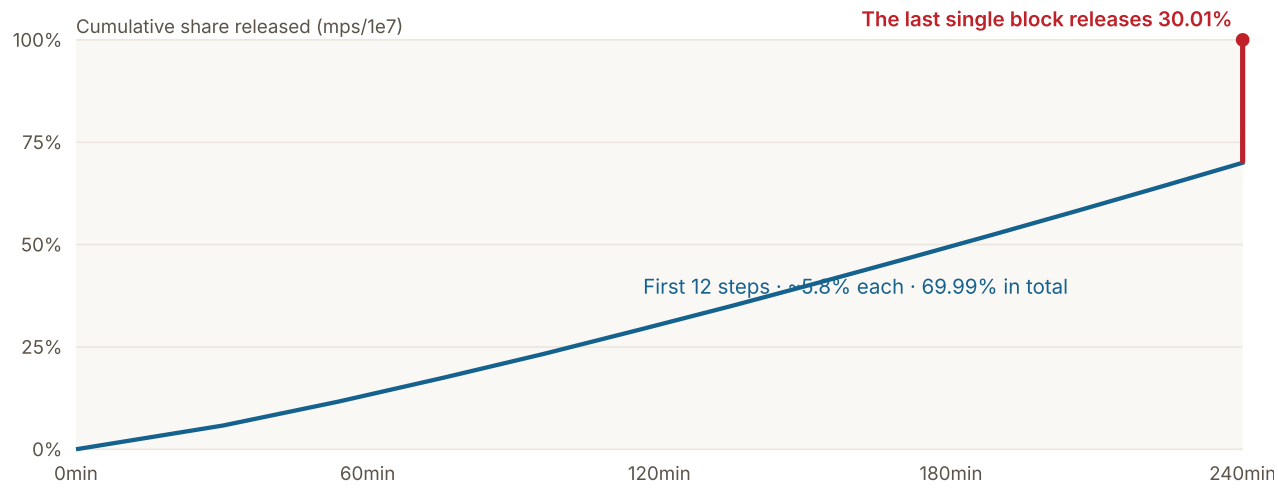
2.2 Instant Launch: every parameter is fixed in the contracts

Two strategy contracts correspond to "creator fee on" and "creator fee off", both with verified source: `0x23f82095…` (fee ON, with FeeSplitter `0xeFF166AA…`) and `0xad44D55E…` (fee OFF). Hard-coded constants: `TOTAL_SUPPLY = 1e9e18`, `LP_FEE = 2500` (0.25%), `TICK_SPACING = 25`, `MIN_LAUNCH_TICK = −160,100`, `MAX_INITIAL_TICK = 251,325`.

The creator can buy in the same transaction (`distributeWithNative`), which is the anti-sniping design the team describes as "the creator buys in the same block the token launches" — in substance it **reserves the first shot for the creator** rather than eliminating sniping. Example: NODEBOX's creation transaction `0x68a09148…` creates the token and self-buys 0.1 ETH in one multicall.

2.3 Crowd Launch timing: 4 hours split into 13 steps

This is the most misunderstood part. The window is **144,000 blocks ≈ 4 hours** (100ms blocks). The supply release schedule is stored on-chain as SSTORE2 data (Robincat's pointer `0x6123b3a5…`); decoded, it is 13 pairs of `(uint24 mps, uint40 blockDelta)`, and the mps sum to exactly `1e7` (100%).



The 13-step supply release curve. Each of the first 12 steps releases ~5.8% at an accelerating pace; the very last block releases 30.01% at once. Decoded from the SSTORE2 bytecode at Robincat's auction pointer.

STEP	MPS	SHARE OF SUPPLY	BLOCKS	≈ DURATION	CUMULATIVE
1	32	5.81%	18,157	30.3 min	5.81%
2	41	5.82%	14,195	23.7 min	11.63%
3–11	45 → 57	≈5.83% each	13,005 → 10,225	21.7 → 17.0 min	64.16%
12	58	5.84%	10,072	16.8 min	69.99%
13	**3,001,126**	**30.01%**	**1**	**0.1 s**	100.00%

Each of the first 12 steps releases ~5.8%, and the number of blocks per step **decreases** (18,157 → 10,072), meaning **supply is released faster and faster per unit of time**. The last block then releases the remaining **30.01%** (about 150M tokens) in one go.

> This "30% all at the end" design has a clear purpose: it guarantees there is still plenty of supply to fill at the final moment, so any order submitted within the window still has a chance to be filled; and it stops the final clearing price from being inflated by "nothing left to sell". The side effect: the last block carries 30% of the whole sale's volume, and whoever's order is still alive at that moment (maxPrice above the clearing price) takes that chunk.

2.4 How the clearing price forms: three invariants

The core logic is in `ContinuousClearingAuction.sol`'s `_checkpointAtBlock()` and `_iterateOverTicksAndFindClearingPrice()`. It boils down to three invariants:

1. **Per-block uniform price**: every buyer filled in the same block pays exactly the same price; there is no ordering advantage. Order within a block has no effect on the fill price.
2. **Only goes up (ratchet)**: the source comment states `"The new clearing price can never be lower than the current clearing price"`. The clearing price is a monotonically non-decreasing ratchet. Cancelling orders doesn't pull it back.
3. **Rounded up to the tick**: the price can only land on integer multiples of `tickSpacing`; Robincat's `tickSpacing = 1% of the floor price`, so the price ladder moves in 1% steps.

Per-block pricing formula (verified with real data):

QUANTITY	FORMULA
Remaining-progress factor `f`	`f = (1e7 − cumulativeMps) / 1e7`
Clearing price `P`	`P = max( P_prev , ceil_tick( D_eff × f / S_remaining ) )`
Raised in this block	`raised_block = D_eff × deltaMps / 1e7`
Tokens filled in this block	`tokens_block = raised_block / P`; unsold supply rolls over to the next block

> **The price changes block by block, not step by step.** Every block creates a checkpoint and recomputes the clearing price; the 13-step table only decides "how much supply this block releases" — each step's mps is **a per-block**

> **release rate** (32 mps per block in step 1, 58 in step 12, 3,001,126 in the last block). So a 4-hour crowd sale logically has 144,000 clearing prices; step changes only alter the release speed, not the pricing frequency.

Here `D_eff` is **the sum of the "effective amounts" of all orders whose maxPrice is strictly above the current clearing price**. The contract keeps history in an inverse-price accumulator `cumulativeMpsPerPrice += (deltaMps << 192) / clearingPrice`, so any order can compute its tokens in O(1) when it exits — the key to keeping gas constant across 4 hours and 144,000 blocks.

2.5 Budget normalization: why orders are "always fully spent"

The order hint in the front end says `"Your budget for this launch window, always fully filled."` That's accurate at the contract level; the mechanism hides in an easy-to-miss normalization step.

The key is `BidLib.toEffectiveAmount()`:

```
effectiveAmount = amountQ96 × 1e7 / (1e7 − bid.startCumulativeMps)
```

When an order enters the demand pool, its amount is first "**normalized by the issuance progress remaining at submission**". Submit 1 ETH at 50% progress and the effective amount entering the pool is 2 ETH. So per block the order actually spends:

```
spend_per_block = amount × deltaMps / (1e7 − startCumulativeMps)
```

Summing `deltaMps` over all remaining blocks gives exactly `1e7 − startCumulativeMps`, so **cumulative spend equals the full budget**. The source comment itself explains the purpose of this step: `"an important normalization step to ensure that we can calculate the currencyRaised ... using the original supply schedule"` .

A budget goes unspent in only two cases:

CASE	CONSEQUENCE	MEASURED
Outbid: the clearing price rises to ≥ your maxPrice	The order stops filling immediately; the unspent part is refundable	Simulation: a bot with maxPrice = 20 ETH FDV, in an auction that finally ran to 340× the floor, spent only **1.8%** of its budget; 98.2% was refunded
Sold out early: all 500M tokens sold	The contract rejects new orders (`AuctionSoldOut`); existing orders stop filling	SnowOn sold all 500M tokens; `remainingSupply = 0`

So the real advantage of entering early is not "spending more" but "everything you buy is cheap supply" — your budget is spread over the blocks where the clearing price was still low. The numbers are in §4.5.

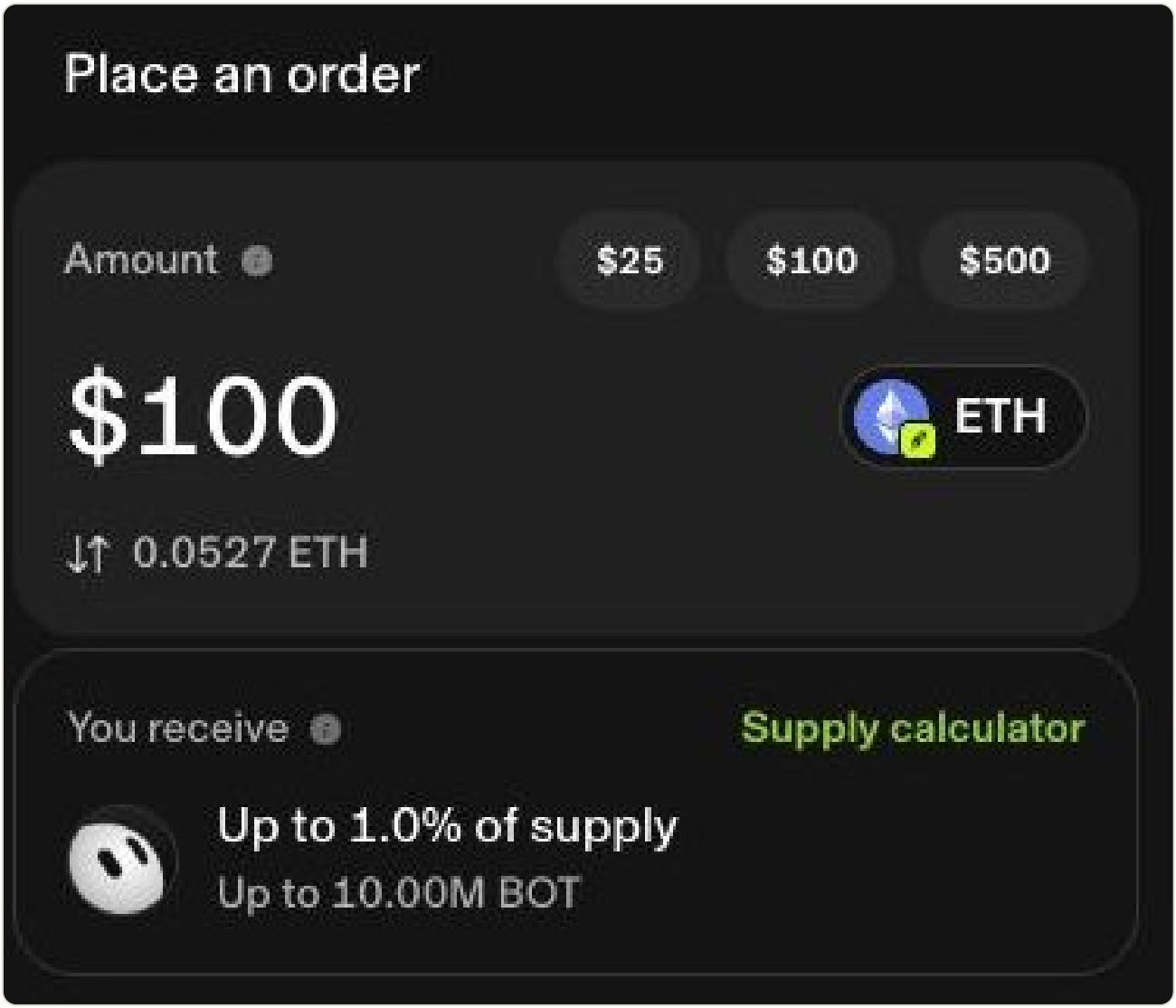
2.6 maxPrice: users can't set it, but it decides everything

At the contract level `submitBid(maxPriceQ96, amount, …)`'s maxPrice is completely free (capped at `MAX_BID_PRICE = 2.0859e39`, i.e. an upper bound equivalent to ~25,000 ETH FDV); but the pools.trade front end **never exposes this input**, only mentioning it in a hint: `"Your order stays active up to a 25,000 ETH fully diluted valuation."` In other words, an ordinary user placing an order implicitly accepts "buy at any price".

This default cap is so high it can practically never be hit. A fully diluted valuation of 25,000 ETH at $1,900 per ETH is about **$47.5M**; yet since pools.trade launched, **the highest clearing price ever reached was 493.87 ETH FDV** (Absolute Goat, ~$938K, see §4.2), a 50× gap. In practice the default order is a **no-limit order**: it never stops filling because "the price got too high"; it just keeps buying until the budget runs out or supply sells out. That explains why in SnowOn's 306 bids, **225** had their price cap at the same value.

Three maxPrice clusters are clearly visible on-chain (measured on INCEL and Robincat):

CLUSTER	MAXPRICE	LIKELY SOURCE
Front-end default	25,000 ETH FDV (≈47,000× floor)	Ordinary users; of SnowOn's 306 bids, **225** fall in this cluster
Script A	~20 ETH FDV (38× floor)	The same address `0xDCf60e2B…` repeatedly uses four fixed amounts — 0.000010 / 0.114286 / 0.057143 / 0.028571 ETH — and shows up in Robincat, HC and INCEL
Script B	A few dozen ticks above the floor	Only wants floor-price supply; outbid as soon as the price moves



The actual order panel (captured 2026-08-18): only the amount, quick-amount buttons and "Up to 1.0% of supply", with no price-cap input at all. What the user places here is a no-limit order.

> The product problem: maxPrice is **the only risk-control knob** in this mechanism, and the front end hides it. Retail can't express "the most valuation I'll accept", while scripts can — which effectively hands scripts an asymmetric tool (at the cost of possibly getting nothing).

2.7 Graduation, migration, refunds, zombie tokens

1. **Threshold check**: `currencyRaised ≥ requiredCurrencyRaised`. Measured, the threshold is always 5× the floor FDV (Robincat 2.640532 ETH, HC 2.657584 ETH, SnowOn 2.652057 ETH), matching the "$10K FDV graduation line" shown in the front end.

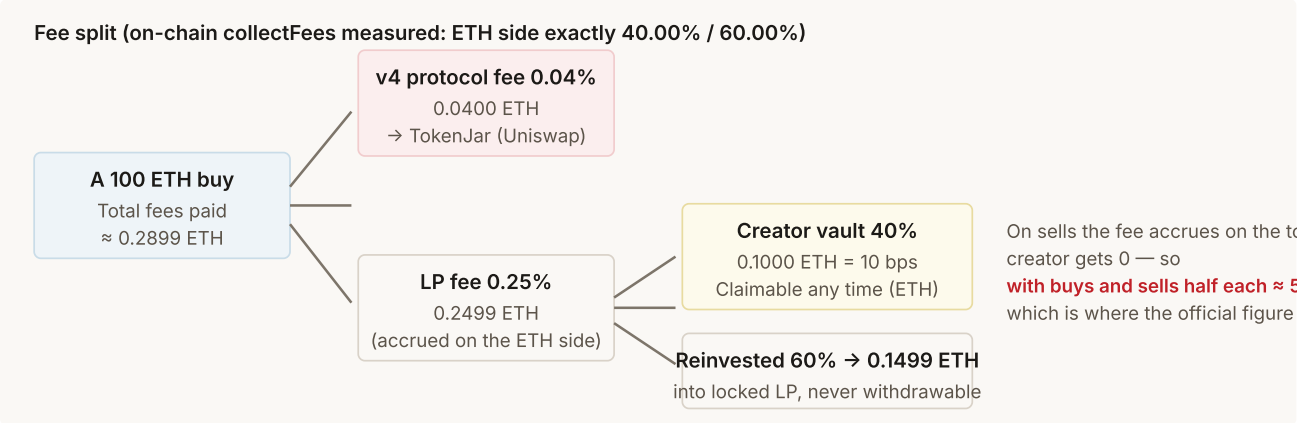
2. **Migration**: once the threshold is met, LBPStrategy (`0x05d55239…`) injects the raised ETH and the 500M reserved tokens into a Uniswap v4 pool (fee 2500, tickSpacing 50, no hook); the recipient of the LP position is FeeSplitter, **not the creator** — this is how "permanently locked liquidity" is implemented.

3. **Failure means refund**: at the contract level it is **pull-based** (users call `exitBid` themselves); the front-end copy says "automatically refunded", a slight overstatement. In practice all 13 Robincat bids were refunded and the auction contract's ETH balance is 0; 5 of them were refunded by **the creator sending an aggregate3 batch on the buyers' behalf**. Funds are safe, but the experience still needs the user to sign a transaction.

4. **Zombie tokens**: tokens from failed sales **are already minted before the auction starts**, just with no pool and zero circulation. Burning the 500M reserve requires someone to actively call `migrate()`, hit the catch branch and burn to `0xdEaD`. Across 2,985 crowd sales since launch this has happened only **twice** `MigrationFailed` — meaning almost no failed sale is ever cleaned up, leaving thousands of permanently dangling empty tokens on-chain.

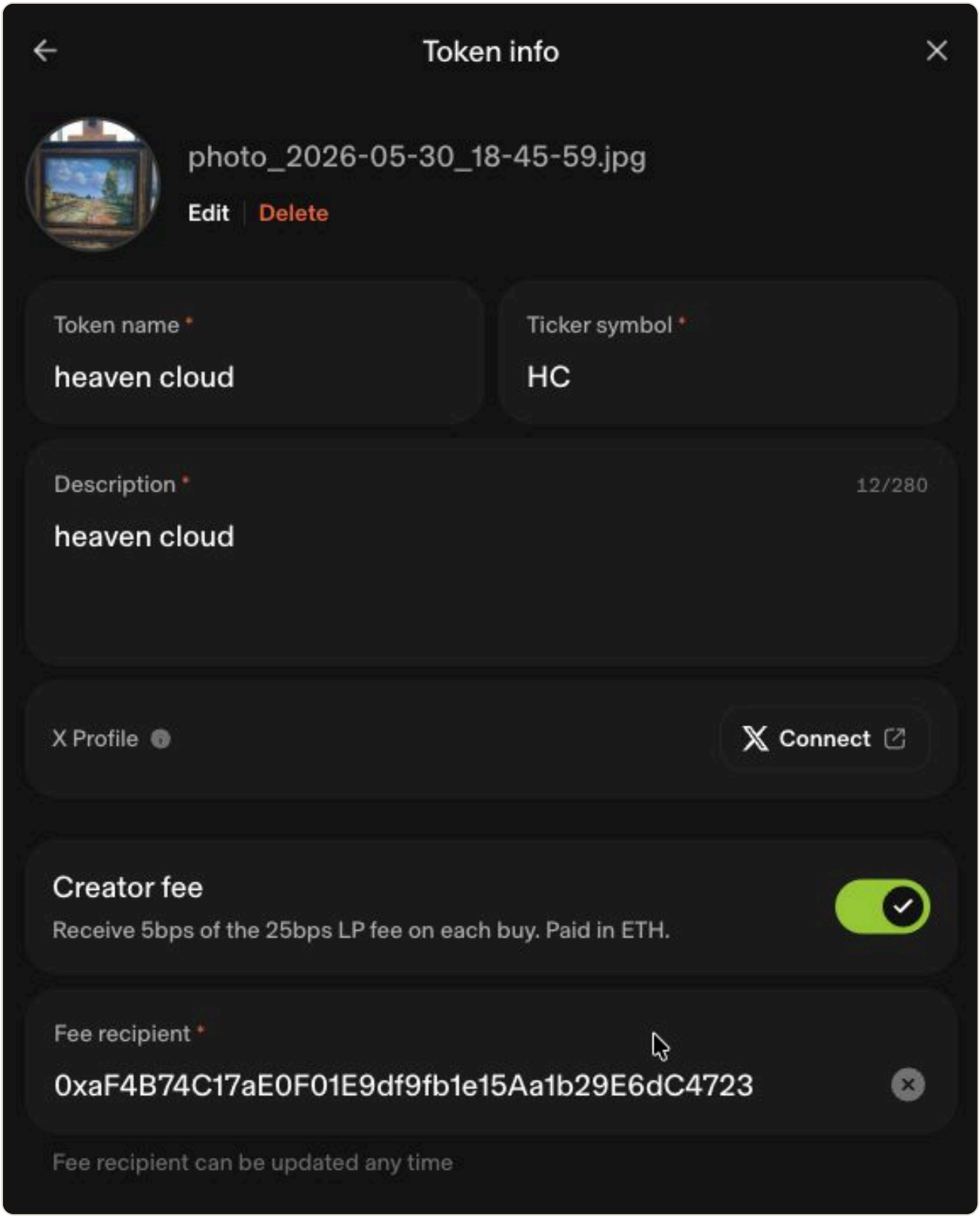
2.8 Fees in full: three numbers and one difference in basis

This is where users get confused most easily. The on-chain facts:



Where the fees of a 100 ETH buy go. The 40%/60% ETH-side split was verified from the internal transfers of three real collectFees transactions.

(1) The real total fee is 0.2899%, not 0.25%. Every pools.trade v4 pool has in slot0 `lpFee = 2500`, `protocolFee = 1638800`. 1638800 = (400 << 12) | 400, i.e. 400 pips on both buys and sells. v4 fees compound in series: `400 + 2500 × (1e6 − 400) / 1e6 = 2,899 pips ≈ 0.2899%`. The same configuration was read on the NODEBOX, MYTHICAL and FRONG pools separately.



The front-end form at token creation: the creator fee reads "Receive 5bps of the 25bps LP fee on each buy. Paid in ETH." Another line on the creator settings page says "40% of the ETH side" — both are right, just on different bases.

(2) The creator fee's "5bps" and the on-chain "40%" are the same thing.

FeeSplitter `0xeFF166AA…`The constructor of `0xd35E9CA7…` hard-codes two beneficiaries: the creator vault **4000/10000** with `0xf9526Dd3…` hard-codes two beneficiaries: the

creator vault **6000/10000**, and the reinvest contract `collectFees`. The internal transfers of three real

COLLECTFEES TX HASH	ETH-SIDE TOTAL	→ CREATOR VAULT	→ REINVEST	SHARE
`0xc50b75ab3708fcd0a7f20bebc8499c44ddd5ca429586656245a61e04eb23fcfb`	0.000635707	0.000254283	0.000381424	40.00 / 60.00
`0x9c61a371a93fd045cd8bfe191c5e64e49e8d70d9654edda8301bd81e06999e41`	0.000236261	0.000094504	0.000141756	40.00 / 60.00
`0x320705a07ea728ee4e560fab131fc0035639f9b103b850287f5610d283ce4af4`	0.000002797	0.000001119	0.000001678	40.00 / 60.00

In the same transactions **100% of the token side goes to reinvest**; the creator gets no tokens. And Uniswap v4 fees are denominated in "the side coming in": on a buy (ETH → token) the fee is booked on the ETH side, on a sell on the token side. So:

- Each **buy**: the creator gets `0.25% × 40% = 10 bps`;
- Each **sell**: the creator gets **0**;
- With buys and sells half each, the average is **5 bps** of total volume — exactly where the front end's `"Receive 5bps of the 25bps LP fee on each buy"` and the blog's `"0.05% of the 25bps fee"` come from.

Another line in the front end already has it right: the creator settings page's `"Receiving 40% of the ETH side of the 25bps LP fee"`. Neither number is false; they're on different bases: **5bps is the average assuming half buys, half sells; 10bps is the true figure on each buy**. When real trading skews to sells it falls below 5bps — FRONG's history is 62.1% sells, so the creator's actual take is about **3.8 bps**.

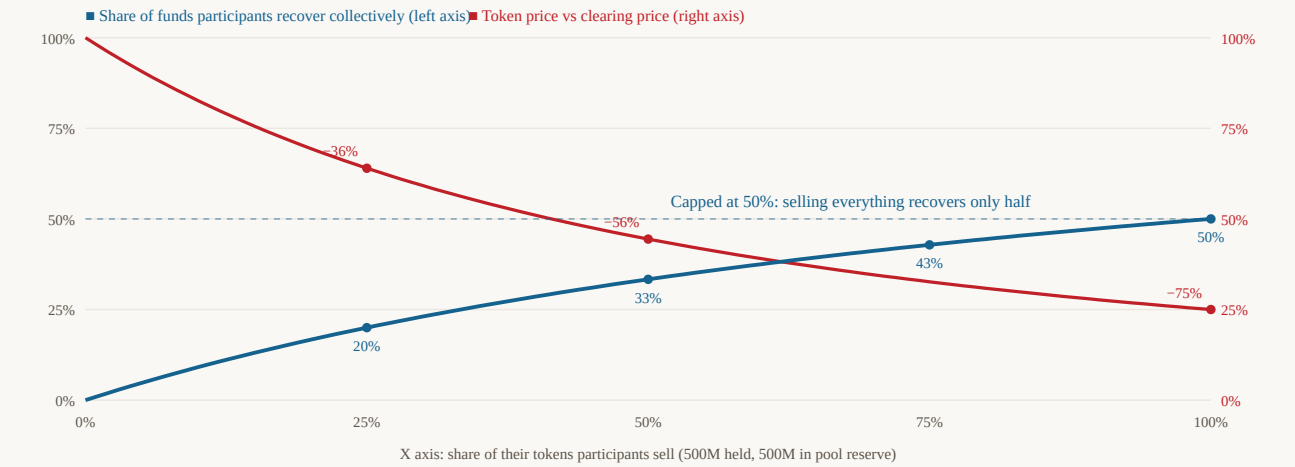
> **Withdrawals from the creator vault go 100% to the beneficiary.** The internal transfers of two `claim` transactions show the vault paying out the address's accumulated share in full in one go, with no further split:
> `0x9497933c88bc3e2d4c79f4ed24c9627b536b1499fcc0ab838c998d65a78d826e` → 0.005384245 ETH paid in full to `0x5628Df44eDA23B433C3797011952345BefEC8979`
> `0xb1665172fe52aa44a39f41f056fc6af2066f54b2e72c760802410b50e1b92667` → 0.006770342 ETH paid in full to `0xA2F84275242e387C516dd53EB79C1799132c8AeD`
> So the chain "40% to the creator vault, vault share 100% claimable" closes

completely. Some sources treat "0.05% ÷ 0.25% = 20%" as the creator's split; that's a coincidence from dividing two different bases, not the 40/60 executed on-chain.

(3) The reinvested 60% never comes back. The front end says explicitly `"Protocol-held. Added to this token's liquidity — not part of your earnings and not claimable."` As of 2026-08-18, the two vaults held: creator vault **6.554929 ETH** unclaimed (23,475 transactions), reinvest contract **6.993135 ETH** (70). About 13.55 ETH ≈ $25.7K in total — the entire fee pool the platform accumulated for creators and pools in 14 days.

2.9 The other side of "permanently locked liquidity": the maths of a collective exit

After a Crowd Launch migrates, the LP is a **single-range position** `[−84900, +88750]` whose upper price bound is about 7,147× the clearing price. This means the pool's ETH only gets swapped out when the price rises; if all buyers sell back, getting the originally injected ETH back requires the price to fall extremely far:



Share of funds participants can recover by selling collectively (blue line, left axis) and token price vs clearing price (red line, right axis). The blue line tops out at 50%.

The supply structure at graduation is fixed: **500M tokens go to bidders; 500M go, together with all the ETH raised, into that irrevocable position** (`reservedTokenAmountForLP` = 500,000,000; unsold supply is burned to `0xdEaD`). In other words, the tokens in participants' hands exactly equal the pool's token reserve.

Under constant product, when the amount sold is $s\times$ the pool reserve, the share of funds recovered is $s/(1+s)$, and the price falls to $1/(1+s)^2$:

PARTICIPANTS SELL	AMOUNT SOLD	FUNDS RECOVERED COLLECTIVELY	PRICE VS CLEARING	EQUIVALENT DROP
25%	125M	20.0%	64.0%	−36.0%
50%	250M	33.3%	44.4%	−55.6%
75%	375M	42.9%	32.7%	−67.3%
100% (dump everything)	**500M**	**50.0%**	**25.0%**	**−75.0%**

The key conclusion is the cap: **even if every participant dumps every token into the pool, collectively they recover at most half their money, and the price stops at 25% of the clearing price.** The other half stays in the position forever — not "hard to recover" but unrecoverable given the circulating supply. Recovering 90% would require selling 9× the pool reserve, and those tokens don't exist.

> Basis: pools.trade uses a v4 single-range position rather than full-range constant product; the range [−84900, +88750] spans a price ratio of about 3.5×10^7. Within the selling pressure participants can reach, it behaves almost exactly like constant product (the range's lower bound makes the actual recovery slightly above 50%, by 1–2 percentage points).

Grounded in SnowOn's real data: all 81.05 ETH raised went into the pool, and participants hold 500M tokens. **Even if these 188 addresses all exit together, they can get back at most ~40.5 ETH; the other ~40.5 ETH stays in the position forever.** On top of that, the reinvested LP fees (6.99 ETH accumulated platform-wide in 14 days) are a completely separate pot; the front end says "not claimable", and no one can take them out by any route.

This isn't a trick; it's an inherent property of constant-product AMMs, and any issuance with locked LP behaves the same. But its distributional consequence is worth spelling out: **locked liquidity shifts the "exit cost" from the project to the buyers as a group**. The project takes the raised ETH (in Crowd mode, the entire raise), while the ETH in the pool belongs to the LP position, held by the protocol and never withdrawable; if buyers want to leave together, they can only trample each other down this curve. This is also the structural reason SnowOn participants are broadly underwater in §4.3.

3. User journeys: three roles, as they really go

3.1 Issuers: 6 steps, free throughout, but pricing power goes to the contract

1. **Connect wallet → Launch a token**. A dialog with two choices; Crowd Launch is selected by default.
2. **Fill in metadata**: name, symbol, image, social links. The token is minted by `TokenFactory` (UERC20, 18 decimals).
3. **Choose whether to enable the creator fee**. Enabling it points the LP position's recipient at the vault-backed FeeSplitter (`0xeFF166AA…`) and lets you set a payout address; the front end notes `"Fee recipient can be updated any time"`.
4. **Confirm**. The creator **doesn't put up a cent of capital** — Crowd mode has no creator self-buy; floor price, supply, duration and graduation line are all fixed by the contract and the issuer can't adjust them.
5. **The 4-hour window**: the only thing to do is bring people in. The issuer has no mechanical lever during these 4 hours (can't extend, can't lower the threshold, can't top up themselves — topping up costs real money and raises everyone's cost).
6. **Settlement**: if the threshold is met, the raised ETH arrives in full (`fundsRecipient` via the LBPStrategy migration path); if not, everything goes to zero and the token becomes a zombie.

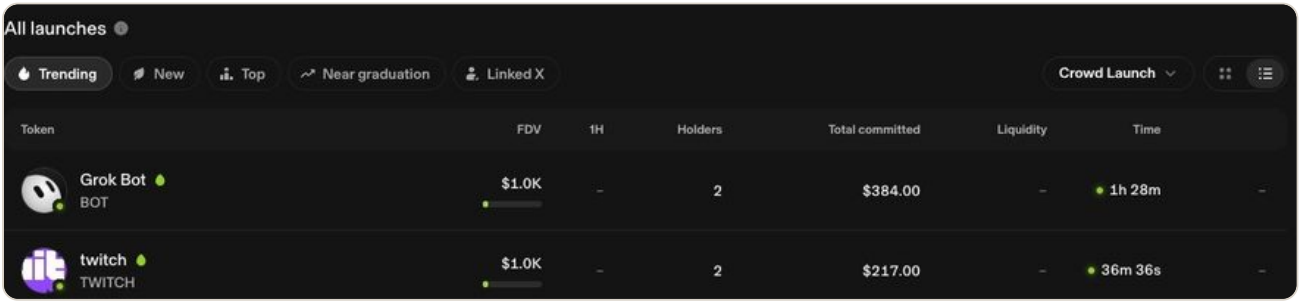
Example: a failed launch (HC "heaven cloud"). The addresses below can all be checked directly on Blockscout:

ITEM	ADDRESS / TX HASH
Token contract	`0x2a36de09912D6380382a0404961D6994aC2890BA`
Auction contract	`0xB94aa16a6fBAB31b0dd46341876924aa2CD66B0d`
Creation tx	`0x594bfdebf85a95cd259bad411e971d4d6b85d5024a8b211747a0ca47607efb76`
Creator / fee recipient	`0xaF4B74C17aE0F01E9df9fb1e15Aa1b29E6dC4723`
Refund tx	`0x2b6033b2bf06d82d50764786581d33a941087edc6bdf8df39c5efc03ad5f90c4`

5 bids in total, adding up to **0.186524 ETH**; threshold 2.657584 ETH, completion **7.0%**; the clearing price stayed at the floor throughout (front end showed `"Price never moved off its floor"`), and everything was refunded (refund tx `0x2b6033b2…`). 4 of the 5 bids came from the same script address that keeps appearing across auctions, `0xDCf60e2BBbD4B3f6263f4cc6500175bFb8B532A8`, with exactly the same amount mix as in Robincat and INCEL — **"someone bid" doesn't mean "real demand exists"**.

3.2 Participants: 5 steps, with three hidden hurdles

1. **Find a live Crowd Launch**. This is the first hurdle: the home page's default list is all tokens; to see live crowd sales you must click the **Near graduation** tab or use the `All tokens ▾ → Crowd Launch` filter at top right. URL parameters don't work; you have to click. Usually only 3–4 sales are live at once.



All results after switching the filter to Crowd Launch (2026-08-18 19:0x): **only 2 crowd sales live on the whole platform**, each with 2 holder addresses, both still stuck at the $1.0K floor FDV. And this filter is hidden in a dropdown labelled "All tokens"; none of the five top-level tabs (Trending / New / Top / Near graduation / Linked X) is "crowd sales in progress".

1. **Open the token page**; a "How Crowd Launches work" pop-up blocks it first, and you must click Continue to use the order panel.
2. **Place an order**: you can only enter an amount (with $25 / $100 / $500 shortcuts) and you see "You receive / Up to X% of supply". **maxPrice is invisible** (the second hurdle). The Supply calculator next to it is a pure division estimator (order ÷ graduation FDV), which the front end itself labels `"Numbers provided are estimates only."`
3. **Wait 4 hours**. In the meantime the clearing price may have risen to where you'd never want to buy, and you have no way to actively reduce risk (you can exit the order, but filled portions are irreversible).
4. **Settlement**: if it graduates, claim tokens in the portfolio; if not, claim the refund — which requires **signing a transaction yourself** (the third hurdle; the front-end copy says "automatically refunded").

3.3 Secondary traders: this is where the real traffic is

Almost all meaningful volume across the platform's 14 days comes from secondary trading in a handful of Instant Launch tokens, not from crowd sales. Front-end readings on 2026-08-18: FRONG FDV $4.2M / 24h volume $1.0M / 13,175 holders; swappy $1.5M / $394.9K; MYTHIC AGENT $1.4M / $147.9K. Among all graduated Crowd Launch tokens, the best, SnowOn, did only $6.5K in 24h.

3.4 Where the front-end wording and the contracts differ

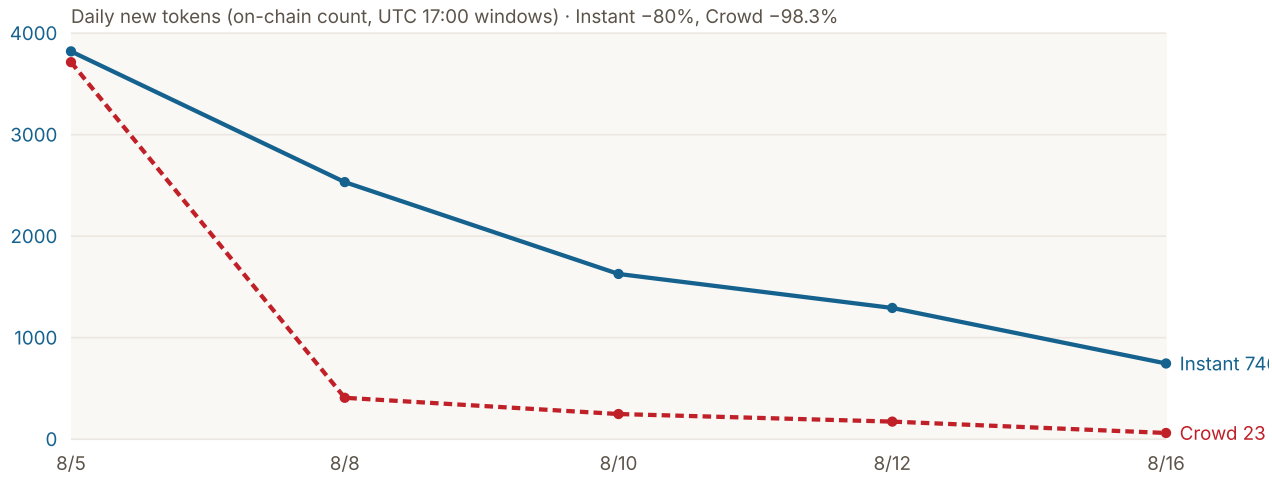
After comparing line by line the front end's full i18n dictionary (776 `rhcca.*` strings), the official blog and the verified contracts, these differences still stand (the creator fee's "5bps vs 40%" is explained in §2.8 as a difference in basis and isn't counted):

FRONT END / OFFICIAL WORDING	CONTRACT REALITY	NATURE
"No launchpad fees"	Issuance is indeed free; but every pool carries a 0.04% two-way protocol fee that goes to Uniswap's TokenJar	Omission
"automatically refunded"	Pull-based; the user must sign `exitBid`; creators were seen batch-refunding on buyers' behalf	Overstated
No top-level entry for "live crowd sales" at all	Fully public and queryable at the contract level	Information-architecture flaw
maxPrice can't be set; a tooltip only mentions the 25,000 ETH FDV cap	`submitBid` first-class parameter that decides whether you get outbid	Capability not exposed
"always fully filled"	**Accurate** (see the normalization in §2.5), unless outbid or sold out	Consistent
"Liquidity locked forever"	**Accurate**; the LP position belongs to FeeSplitter / protocol custody, the creator can't take it	Consistent
"TWAP bids prevent bundling"	**Accurate**; per-block uniform pricing does remove same-block ordering advantage	Consistent

> One easily missed disclaimer: "Any use of Pools for an asset other than a meme coin is prohibited. CCA is available for other types of assets, subject to other applicable terms and conditions." That is, the pools.trade front end is explicitly restricted to memes, while the CCA auction mechanism is reserved for other (regulated) asset classes. That line more or less writes the product's strategic intent on its face.

4. Operation: 14 days of data

4.1 The big picture: Instant halved, Crowd to zero



Daily new tokens (on-chain count, windows aligned to UTC 17:00). Note the two lines use different y scales: Instant on the left axis up to 4,000, Crowd rescaled to a full scale of 1,500 to compare slopes.

WINDOW (UTC 17:00)	INSTANT	CROWD	TOTAL	VS DAY ONE
8/5 → 8/6 (day one)	3,821	1,393	5,214	100%
8/8 → 8/9	2,533	153	2,686	51.5%
8/10 → 8/11	1,628	93	1,721	33.0%
8/12 → 8/13	1,293	65	1,358	26.0%
8/16 → 8/17	746	23	769	14.7%

The two lines decay at completely different speeds: Instant Launch dropped **80%**, a typical cooling for a launch platform; Crowd Launch dropped **98.3%**, from 1,393 sales a day to 23 — effectively **a death sentence from the market**. Launchers voted with their feet for "tradeable immediately".

> Day one's 1,393 Crowd sales include many tentative (zero-cost) creations and can't be read as real demand; even so, the 1,393 → 23 slope is far steeper than Instant's, showing the decay comes from the mechanism itself rather than overall interest.

4.2 Graduation rate: 2.43%, and graduating isn't success

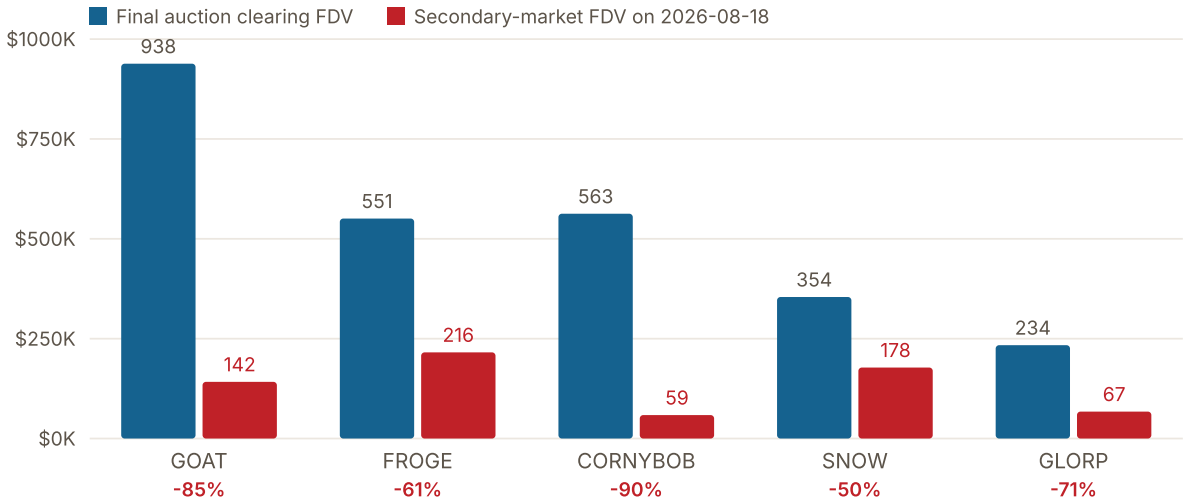
The numbers in this section come from a full scan: every LBPStrategy `InitializerCreated` and `Migrated` event since pools.trade launched (blocks 27,788,000 to 39,666,441), split by migration config into "crowd sales started from the pools.trade front end" and "crowd sales started by third-party integrations" — the test being 500M reserved for LP, remaining tokens sent to `0xdEaD`, and pool parameters fee 2500 / tickSpacing 50.

2,263 TOTAL POOLS.TRADE-STYLE CROWD SALES	**55** GRADUATED AND CREATED A POOL (2.43%)	**916 ETH** TOTAL RAISED BY GRADUATED SALES ≈ $1.74M
6.17 ETH MEDIAN RAISE OF GRADUATED SALES (~$11.7K)		

> Basis: the front-end list page shows only about 11 of the graduated tokens; the rest need a search or a direct link, so counting from the front-end list badly undercounts graduations. Every number in this section comes from a full scan of on-chain events.

Raises among graduated sales are extremely uneven: 6 at ≥50 ETH, 18 at 10–50 ETH, 8 at 5–10 ETH, 16 just over the line (2.6–5 ETH). The biggest raise isn't the much-discussed SnowOn but **Absolute Goat (GOAT) at 146.21 ETH**.

But performance after graduation is more telling than the graduation rate. Convert each sale's final clearing price to FDV and compare with the secondary-market FDV on 2026-08-18:



The five graduated tokens with a current price available: final clearing FDV vs current secondary-market FDV (ETH at $1,900). The clearing price is what every participant in the auction actually paid, so the gap between the bars is participants' paper P&L.

TOKEN	RAISED	CLEARING FDV	CLEARING / FLOOR	CURRENT FDV	PARTICIPANT P&L
Absolute Goat (GOAT)	146.21 ETH	≈ $938K	942.5×	$141.9K	**−85%**
SnowOn (SNOW)	81.05 ETH	≈ $354K	351.6×	$177.7K	**−50%**
FROGE	70.03 ETH	≈ $551K	552.9×	$215.7K	**−61%**
Corny on the Bob	56.35 ETH	≈ $563K	560.3×	$58.7K	**−90%**
Glorp	50.17 ETH	≈ $234K	234.6×	$67.4K	**−71%**

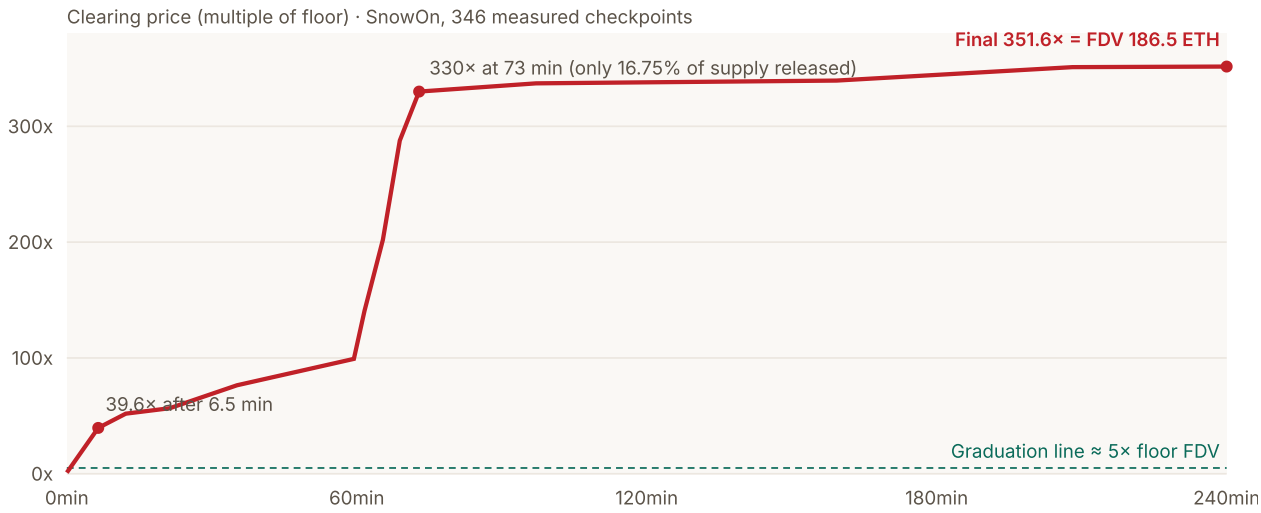
Five samples, five losses, −71% on average. This isn't cherry-picking — it's every token among the top 6 raises whose current price can be found in the front end. Together with the clearing-price path in §4.3, the mechanism-level explanation is: demand within the 4-hour window pushes the clearing price to 150–940× the floor, and once the window closes there's no new buying to support that price. **What CCA discovers is not a "fair price" but "the clearing price of a new-issue game".**

> Over the same period third-party integrations ran 722 crowd sales through the same CCA factory, and 65.3% triggered migration. That looks good but **can't be compared with 2.43%**: sampling 150 of them, about 3/4 of the "graduated" ones raised less than 0.1 of their quote unit (23% even less than 0.001), meaning these integrators set the graduation threshold near zero — migration for them is a certainty, not an achievement. Their quote currencies weren't stablecoins or RWAs either, but ETH and other meme tokens (TIRED, Bark, ROBINHOODS, STONKBROKER, FRONG).

4.3 Case study: SnowOn (SNOW)'s four hours

The second-largest graduated raise, and the sale with the most complete clearing-price record, is worth looking at item by item. Auction `0x9c6ef3E1…`, token `0xb851CeBc…`.

81.05 ETH RAISED ≈ $154K, 30.6× THE THRESHOLD	**351.6×** FINAL CLEARING / FLOOR; FDV 186.5 ETH ≈ $354K	**306 bids / 188 addresses** 126.23 ETH BID IN TOTAL; 64.2% ACTUALLY FILLED
−50% CURRENT FDV $177.7K VS CLEARING FDV $354K		



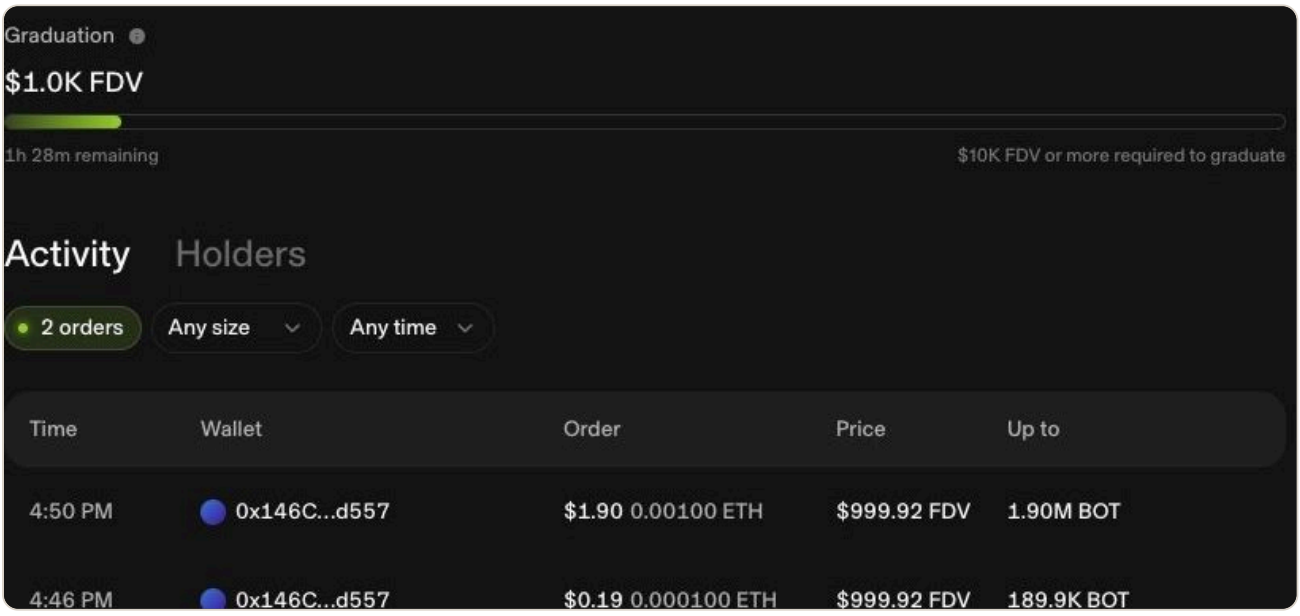
SnowOn's actual clearing-price path (346 CheckpointUpdated events). The price covered 94% of its rise in the first 73 minutes, when only 16.75% of supply had been released.

A few facts that must be called out:

- **The price covered 94% of the way in the first 73 minutes.** The clearing price was 39.6× the floor at t+6.5min and 330× at t+72.9min — when supply had released only **16.75%**. Over the remaining 167 minutes and 83% of supply, the price only crept from 330× to 351.6×. That's exactly what the mechanism predicts: demand concentrates early, early demand fixes the price in one go, and the rest becomes "distributing at an already-set price".
- **85% of orders came in during the first half** (time quartiles: 147 / 112 / 28 / 19 bids). So the "early entry gets a better price" incentive **did work** — the problem the whitepaper set out to solve, "participation concentrated at the last moment", was solved in this sample. It's the most solid point in CCA's design.
- **But the quality of price discovery is another matter.** The clearing price was set at $354K FDV; the secondary market is now $177.7K, with just $6.5K of 24h volume and 447 holder addresses. All participants are collectively about half underwater — and this is the **best** of the 5 comparable samples. The mechanism successfully "discovered" a price the market doesn't accept — because demand during the 4-hour window came mainly from the new-issue game rather than a view on value.
- **All 500M tokens sold out** (`totalCleared = 500,000,000` (`remainingSupply = 0`); of the 126.23 ETH bid, 45.18 ETH was refunded because it was outbid or the sale sold out.

4.4 Failed samples: two side by side

	HC (HEAVEN CLOUD)	ROBINCAT
Auction contract	`0xB94aa16a6fBAB31b0dd46341876924aa2CD66B0d`	`0xf38BF5441eC52aDEeb9EA141ba1c42f17E0918fe`
Threshold	2.657584 ETH	2.640532 ETH
Actually raised	0.186524 ETH (7.0%)	0.876883 ETH (33.2%)
Bids	5	13
Final clearing price	Stayed at the floor (1.000×)	3.350× floor
Script involvement	4/5 bids from `0xDCf60e2BBbD4B3f6263f4cc6500175bFb8B532A8`	4/13 bids from the same address, with the same four fixed amounts
Outcome	Fully refunded	All 13 refunded, contract balance 0; 5 batch-refunded by the creator



A crowd sale in the middle of failing (Grok Bot, captured 2026-08-18, 1h28m left): progress bar $1.0K / $10K FDV, only 2 orders in the whole sale — $1.90 and $0.19 — from the same wallet. This isn't an outlier; it's what the 2.4% graduation rate in §4.2 looks like day to day.

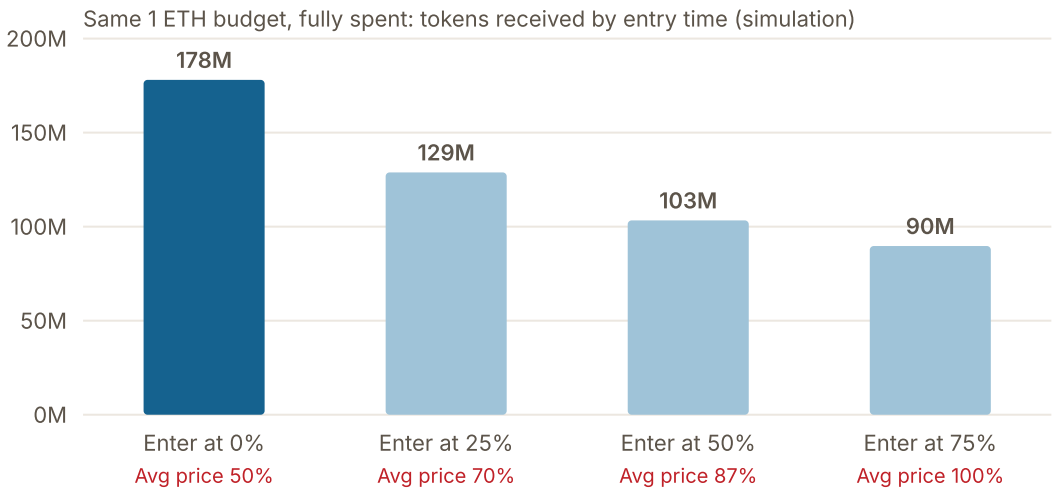
Robincat's record is complete enough to recompute all 13 real orders block by block with the pricing rules in §2.4–2.5. The recomputed clearing-price path matches the on-chain `CheckpointUpdated` events almost point for point:

SUPPLY RELEASED	ON-CHAIN CLEARING PRICE	RECOMPUTED BY THE RULES
0.271%	1.000×	1.000×
0.394%	1.483×	1.490×
0.556%	1.699×	1.700×
1.075%	1.972×	1.980×
2.258%	2.527×	2.530×
4.699%	3.138×	3.140×
100%	3.350×	3.350×

The raise matches too: on-chain **0.876883 ETH**, recomputed **0.876645 ETH**; the 0.000238 ETH difference comes from the partial fill of the last order "right at the clearing price". Notably, **the budgets of the first 12 orders sum to exactly 0.876645 ETH** — every one of them spent 100% of its budget, exactly as the normalization in §2.5 predicts. The 13th order's price cap equalled the clearing price at the time, so it barely filled and finally exited via `exitPartiallyFilledBid`.

4.5 Scenario simulation: what is early entry actually worth

Running controlled experiments on the verified model (floor and tick from Robincat's real parameters, 500M supply, all orders with the front end's default high maxPrice):



Scenario A: four orders of 1 ETH each, all with the front end's default maxPrice, differing only in entry time. All spend 100% of their budget; the difference shows up in the average fill price (red labels show the average as a share of the final clearing price).

SCENARIO	ENTRY PROGRESS	BUDGET	SPENT	TOKENS RECEIVED	AVERAGE FILL FDV	VS FINAL CLEARING
A: four buyers, 1 ETH each Final clearing 21.1×	0%	1 ETH	100%	178.04M	5.62 ETH	50.4%
	25%	1 ETH	100%	128.83M	7.76 ETH	69.6%
	50%	1 ETH	100%	103.32M	9.68 ETH	86.8%
	75%	1 ETH	100%	89.70M	11.15 ETH	100.0%
B: 20 orders × 0.5 ETH Final clearing 44.1×	First order 0%	0.5 ETH	100%	37.75M	13.24 ETH	56.9%
	Last order 47.5%	0.5 ETH	100%	21.48M	23.27 ETH	100.0%
C: 60 orders × 2 ETH (SnowOn scale) Final clearing 520.9×	First order 0%	2 ETH	100%	13.98M	143.10 ETH	52.0%
	Last order 44.2%	2 ETH	100%	7.27M	275.10 ETH	100.0%
D: low maxPrice, outbid Final clearing 340×	0%	0.114 ETH	**1.8%**	0.26M	8.15 ETH	2.4%

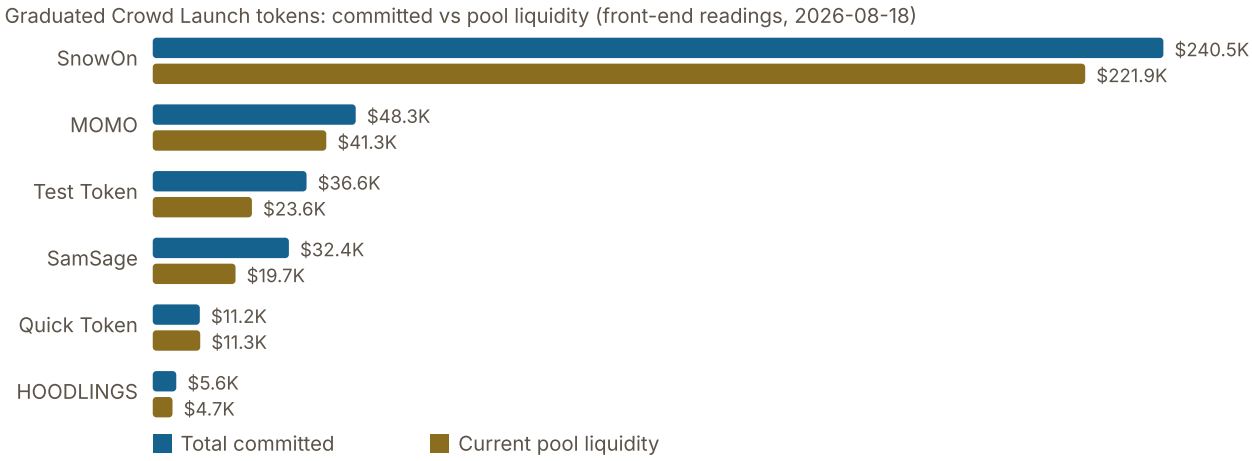
Three conclusions you can trade on directly:

1. **The upside of entering early is stable at about 2×**. Across three demand levels, the first-to-last token ratio falls between 1.76× and 1.98×, with an average fill price of about **50%–57%** of the final clearing price. This "2×" isn't luck but the mechanism's structural ceiling — early money is spread along the whole upward path.
2. **The stronger the demand, the more early entry is worth**, but everyone's absolute cost is pushed up: in scenario C, 120 ETH of demand drives the clearing price to 520.9× the floor (FDV ≈ $523K), and even the first order's average is already 143

ETH FDV. New-issue profits rely on the secondary market taking over, and §4.3 shows it often can't.

3. **A low maxPrice cuts both ways**: scenario D's bot pushes its average down to 2.4% of the final clearing price (extremely cheap), at the cost of buying only 0.26M tokens with 98.2% of its money refunded. That's exactly the strategy of the script address `0xDCf60e2B…` — sweep floor-price supply with a very low cap; it fills completely in quiet sales and comes away almost empty in hot ones.

4.6 Secondary-market structure: extremely concentrated



Total committed and current pool liquidity for graduated Crowd Launch tokens (front-end readings, 2026-08-18). Pool liquidity roughly equals the amount committed, because the raised ETH and the 500M reserve are injected together into a single-range position.

For graduated crowd tokens, the relationship between commitments and final pool liquidity is clear: **pools receive liquidity of about 0.9–1.0× the amount committed** (SnowOn $240.5K committed / $221.9K liquidity), because the raised ETH and the 500M reserve go together into a single-range position. But these pools barely trade.

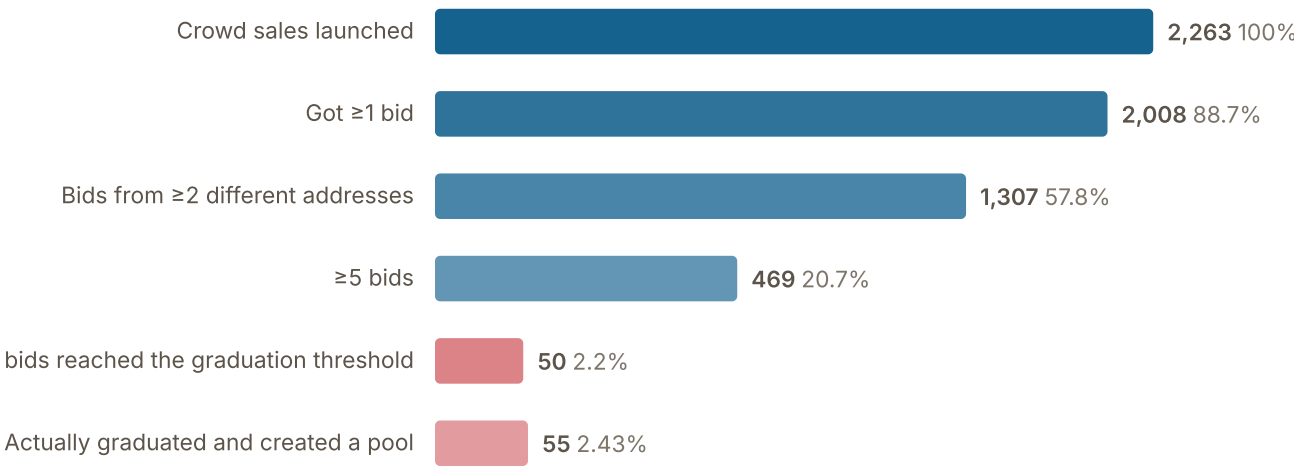
Across the whole chain (including non-pools.trade pools), the last hour's **33,795** swaps were spread over **1,005 pools**; the top 10 pools took **42.1%**, the top 3 **23.9%**, while **463 pools (46%)** had ≤2 trades in the whole hour. The head effect is more extreme than on a typical launch platform.

Now the decay of the hottest token: FRONG's day-one media snapshot was FDV **$12.1M**, 24h volume **$20.6M**, $60.96M cumulative, 12,206 buying wallets (The Defiant, 2026-08-05); by 8/18 it was FDV **$4.2M**, 24h **$1.0M**, 13,175 holders. **Valuation −65%, daily volume −95%, while holder count barely moved** — the classic "people still here, money no longer moving". FRONG's pool also holds only

71.71M tokens (7.2% of supply), so holdings are widely dispersed in wallets rather than in the pool.

4.7 Conversion funnel and who participates

Joining the 2,263 crowd sales with all-time `BidSubmitted` events (17,855 bids; 17,836 after removing one anomalous sale not priced in ETH) draws the product's real conversion funnel:



The full conversion funnel for pools.trade-style crowd sales. Bar lengths use a square-root scale to show the tail. "Total bids reached the threshold" and "actually graduated" differ slightly because graduation uses actual fills (`currencyRaised`), while total bids include amounts refunded after being outbid or a sell-out.

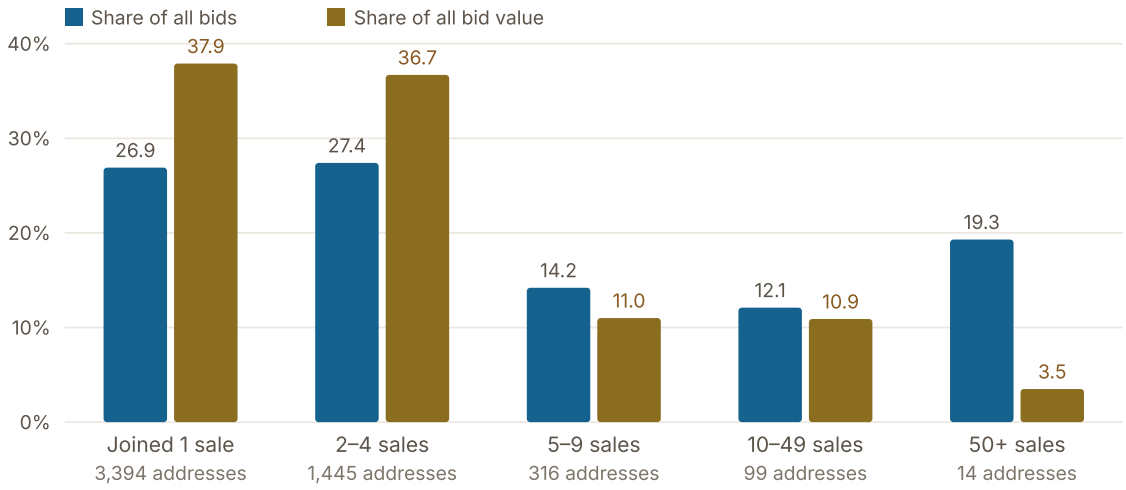
Three break points stand out:

1. **88.7% of sales got at least one bid** — so "nobody saw it" isn't the main cause of death; there are enough sweeping scripts on-chain that almost every sale gets an order.
2. **From "has a bid" to "has 5 bids" loses three-quarters** (2,008 → 469). The median number of bids in sales with any bids is just **2**, and 34.0% of sales have only 1. The real cause of death is **not enough people**.
3. **From "5 bids" to "reaching the threshold" loses another 89%** (469 → 50). Even with people, there isn't enough money: the platform's **median bid is just 0.0105 ETH ≈ $20**, while the threshold needs about 2.65 ETH ≈ $5,000 — that's 250 median-size orders.

In other words, for a project with no community, the $10K graduation threshold means gathering hundreds of retail buyers; and §4.1 shows daily launches fell 98.3%

in two weeks, meaning the retail willing to show up is leaving fast. **This is a funnel whose denominator is growing while its numerator shrinks.**

Who participates: 5,268 addresses, with an extremely skewed structure



Bidding addresses grouped by how many crowd sales they joined, comparing each group's share of all bids and of all bid value. Value excludes the anomalous sale not priced in ETH.

All bids come from **5,268 distinct addresses**, totalling **1,321 ETH** (≈$2.5M, 14 days). There is a very clear split in structure:

- **64.4% of addresses joined only 1 sale** yet contributed **37.9% of the value** — real retail: come once, place one order, done.
- **14 addresses (0.3%) joined 50+ sales**, contributing **19.3% of bids** but only **3.5% of value** — high-frequency, tiny amounts: textbook automated sweeping.

ADDRESS	SALES JOINED	BIDS	TOTAL ETH	BEHAVIOUR
`0xefea6039fed264e9a1d3319d7d814a79b1a633a1`	1,499	1,513	12.22	Covers **66%** of all sales, dropping a fixed 0.008 ETH in almost every one
`0xc835721c699235b7a5dfdcb122b143f6d55bceb4`	197	520	0.015	Amounts as low as 0.00001 ETH: probing / placeholder
`0xf511e1c9ebf4bb1577e91ee07b5d1fc07bde87f9`	146	151	2.02	Amount tweaked per sale (0.0130–0.0131), like a script back-solving from FDV
`0x107f1d453b2945bcd476c3cf3c1b378f4f6ecdb9`	127	127	6.35	Exactly 0.05 ETH per sale, one order each
`0xdcf60e2bbbd4b3f6263f4cc6500175bfb8b532a8`	124	316	9.97	Fixed four-amount mix 0.00001 / 0.028571 / 0.057143 / 0.114286, price cap at 20 ETH FDV
`0xd835a42e7a4a8693a326cd7e03c53ca1dd101cfa`	108	108	2.63	Only uses two amounts, 0.02 and 0.03
`0x9d6154be6a1bd36bea03747c553838a47a808912`	92	96	0.30	Small tweaks around 0.0026
`0xb1a3a02a64392d23a7782aa65bbb70487e728575`	80	89	0.49	Small tweaks around 0.0052

The fifth address in the table is the one repeatedly seen in Robincat, HC and INCEL; on-chain it is confirmed to have swept **124 sales**. Together with the second address it covered **188 sales** (9.4% of sales with bids), and in **64 of them every bid came from these two** — that is, **3.2% of crowd sales never had a single real retail participant**, and the "someone has bid" shown in the front end was entirely scripts.

> **An unexpected finding: launchers and bidders are the same addresses.** The top two bidding addresses in the table are also the LP-position recipients in third-party integration crowd-sale configs (`positionRecipient`) — 42 sales for the first, 35 for the second. The same operator both launches tokens in bulk and sprinkles small orders across other people's sales. That explains why "88.7% of sales have bids" while only

> 2.43% graduate: **a good share of the "participation" is manufactured by the issuance side itself.**

5. Competitors

	CROWD LAUNCH (CCA)	INSTANT LAUNCH	PONS (SAME-CHAIN RIVAL)	CLASSIC BONDING-CURVE PLATFORMS
Pricing mechanism	Continuous clearing auction, per-block uniform price, 144,000 blocks	Bonding curve, instant fills	bonding curve + v4 hook	Bonding curv one-shot execution
Anti-sniping	**Structurally solved**: same block, same price; ordering gives no edge	None, only "creator buys first in the same block"	Relies on hooks; details not public	None; it's a fi over gas and mempool position
Barrier to entry	High: wait 4 hours, maxPrice invisible, must understand the clearing price	Low: same as buying spot	Low	Low
Failure protection	**Yes**: full refund below $10K FDV	No such concept	None	None
Liquidity ownership	Permanently locked, protocol custody, creator can't withdraw	Also permanently locked	Locked (basis not disclosed)	Mostly locked auto-added t pool
Fees	0 on issuance; 0.2899% on trades (LP 0.25% + protocol 0.04%)		Not publicly verified	Per the officia blog, "other platforms ~1%
Creator income	Optional: 10 bps of each buy (= 40% of the ETH side of the LP fee)		Yes (basis not disclosed)	Some platforr
Measured adoption	**23 sales on 8/16; graduation rate 2.43%**	**746 on 8/16**	7,210 tokens on day one (DeFiLlama); PONS −49% the same week	—

Across the board, CCA is **genuinely ahead of every rival on mechanism**: it's the only design that at once removes same-block ordering advantage, gives early money a positive incentive, refunds on failure and makes liquidity irrevocable. But **on adoption**

it lost to its own sibling Instant Launch, 1 : 32. This comparison is the single most important sentence in the report: **mechanism superiority did not turn into willingness to use it**.

> Pons on the same chain deserves a mention: it upgraded to a v4 hook a week before pools.trade launched and had been listed in Uniswap's own launchpad aggregator six days earlier — so Uniswap was sending traffic to a rival while building pools.trade itself. Day one: pools.trade 10,506 tokens vs Pons 7,210; Uniswap won day-one traffic with "zero fees + official brand". Two weeks later both are cooling fast (the PONS token −49% the same week). This looks more like the whole sector ebbing than any one player failing.

6. Risks and opportunities

6.1 Mechanism risks

RISK	HOW IT WORKS	SEVERITY
Clearing price systematically detached from market price	Demand within the 4-hour window is mainly a new-issue game, and the clearing price gets fixed by concentrated early demand (SnowOn covered 94% of its rise in 73 minutes). The secondary market then reprices; all 5 comparable graduated tokens now trade below their clearing price, −71% on average	High
Sunk cost of a collective exit	Single-range locked LP: for buyers to recover half the ETH, the price must fall 74% (§2.9). The project takes the whole raise; buyers bear the exit cost collectively	High
The "scientists" didn't disappear; they changed form	Anti-sniping removes the edge of being fast, but the mechanism hands the edge to whoever "calculates accurately, has more money and understands the rules": the fixed 2× from early entry, sweeping the floor with a low maxPrice, and the concentrated harvest window of the last block's 30% are all programmable strategies. Script addresses reusing fixed amount mixes across sales are already identifiable on-chain	Medium-high
30% concentrated in the last block	The last block fills 30% of the sale's supply at a price set by the preceding 4 hours. Any large order still alive at that moment can take a huge chunk in one go	Medium
Zombie-token pile-up	Tokens from failed sales are minted but have no pool and zero circulation; they're only burned when someone manually calls `migrate()`; only 2 `MigrationFailed` ever, meaning thousands of empty tokens left dangling forever	Low (dirty data)
Locked LP ≠ safe token	Locking only guarantees the pool can't be pulled; it doesn't stop a team dumping tokens bought in the auction, or immediate post-graduation distribution and sell pressure	Medium

6.2 Product risks: this is where Crowd Launch really fails

1. **All the complexity is left to users, without the matching control.** Users have to understand per-block uniform pricing, the clearing-price ratchet, 13-step release and effective-amount normalization — yet the only risk knob, maxPrice, is hidden by the front end. Complexity without control is the textbook product-design flaw.
2. **No top-level entry.** A product sold on "crowd sales" has no "live crowd sales" list on its home page; you need the Near graduation tab or a dropdown filter, and URL parameters don't work. Only 3–4 sales are live at once, and even those are hard to find.
3. **A 4-hour wait clashes with the user instinct of "tradeable any time".** This is fatal friction for memes: memes are about immediacy and narrative windows, and CCA forces a 4-hour cooling-off period in. Launchers' 1:32 vote with their feet is the quantified result of this clash.
4. **Gaps between copy and implementation erode trust** (§3.4). Especially "automatically refunded" and "no launchpad fees" — both are things users will run into themselves.

6.3 Business risks: the fee scale supports no narrative

After 14 days the two fee vaults hold **13.55 ETH ≈ $25.7K** in total (creator vault 6.554929 ETH unclaimed + reinvest pool 6.993135 ETH), and chain-wide protocol fees total `protocolFeesAccrued(ETH) = 0.0433 ETH`. Even extrapolating linearly from the best day, this is negligible for Uniswap. **So pools.trade's business goal can't be fee income**; it can only be strategic positioning: grabbing the new chain's asset-issuance entry point and default AMM slot. By that standard it already has what it wanted (every graduated token's pool is a Uniswap v4 pool), at the cost of a crowd-sale product with very low usage.

6.4 Opportunities

- **CCA's right battlefield isn't memes.** The whitepaper's two goals (price discovery, liquidity bootstrapping) and the disclaimer's line "CCA is available for other types of assets" already point the way: assets that need auditable, manipulation-resistant pricing — RWAs, tokenized stocks, fund units, new assets that need compliant issuance. In those settings, "4-hour window + uniform price + refund on failure" turns from a weakness into a strength. **But to be clear: no such use case is visible on-chain yet.** Third-party integrators have already run 722 crowd sales on the same factory; sampling shows their quote currencies are still ETH or other

meme tokens (TIRED, Bark, ROBINHOODS, etc.), and most set the graduation threshold near zero. The factory is being reused, but the reusers are still launching memes.

- **There are clear, cheap product fixes**: expose maxPrice (even just three presets: conservative / neutral / no limit), make "live crowd sales" a top-level entry, push refunds in a batch at migration (creators already do this by hand), and before an order show a sensitivity table — "if the clearing price rises to X you get Y" — instead of a pure division estimator.
- **The 2× structural edge of early entry is a sustainable strategy space**. For active traders, the most certain return in CCA sales doesn't come from "predicting the final value" but from "sweeping floor supply with a low maxPrice in quiet sales + entering first in hot ones". Neither needs forecasting, only monitoring.
- **Stitch Crowd and Instant together**: today they're completely separate. A more sensible shape is "Instant first; once a real volume threshold is reached, trigger a CCA follow-on issue", so CCA serves demand that has already been proven instead of cold-starting new coins with zero demand.

7. My four judgments

The first six chapters are mechanism and data. This chapter is my judgment as a product manager, kept separate from the facts.

7.1 Cutting fees to 0.25% cut in the wrong place

For a MEME launchpad, a 1% trading fee is the right level. Players happily accepted 3% burn taxes on early degen coins, because what MEME players care about has never been the fee but whether the coin still has volatility and upside — in a category that can do 100× in 7 days, daily doubles or halvings are normal later on, and a 0.7% fee difference is lost in the noise. Fees that are too low directly mean the platform has no income and creators have no motivation; nobody wants to drive traffic and spread the word for the coin.

Do the maths on the platform's best token (FRONG, front-end reading on 8/18: ~$1.0M daily volume):

FEE SETTING	DAILY FEE POOL	CREATOR'S DAILY INCOME	NOTES
pools.trade actual (0.2899%)	$2,899	≈ $379	Creators only get 10 bps on buys; FRONG's history is 37.9% buys
For comparison: 1% fee	$10,000	—	The official blog treats "other platforms ~1%" as the rival it beat

Over 14 days the whole platform accumulated only **13.55 ETH ≈ $25.7K** in fees for creators and pools. Low fees benefited nobody: the platform gave up revenue (it never planned to take any), creators got too little to fund any promotion, and the fees buyers saved mean nothing on an asset that moves 30% intraday. This carries spot-trading values (low fees, high capital efficiency) into a market that doesn't decide on fees — the thing Uniswap is best at isn't a competitive edge here.

To be fair: low fees do cut the cost of market making and high-frequency arbitrage and genuinely help secondary liquidity; FRONG's pool depth is thick for a meme of its size. But a launchpad's bottleneck has never been secondary friction; it's whether anyone will help push the coin out. 0.25% solves a problem the latter doesn't have.

7.2 On Uniswap running its own launchpad

In the business world this is no big deal. But the timeline is worth laying out: per The Defiant (2026-08-06), Pons on the same chain upgraded to a v4 hook a week before pools.trade launched, and was listed in Uniswap's own launchpad aggregator **6 days** before launch; 6 days later pools.trade launched with "zero platform fees + official brand", and its day-one token count **10,506 vs 7,210** overtook it.

Traditional internet businesses have a neutral name for this: first-party selling — the same entity provides the distribution channel and sells its own products on it. It isn't illegal and breaks no crypto-industry norm; its cost is spending part of the "neutral infrastructure" premium, which is exactly where Uniswap's credibility as an aggregation layer comes from.

The result is more noteworthy: Crowd Launch's daily launches fell **98.3%** in 14 days, graduation rate **2.43%**, with its sibling Instant Launch winning **32:1**. One objective consequence of first-party selling is that failure also becomes publicly verifiable data.

The other side must be acknowledged too: the public goods Uniswap delivered are real — v4 pools, permanently locked LP in protocol custody, fully open-source

verifiable contracts, zero issuance fees. Most launchpads' "locks" are verbal promises. On infrastructure quality, what it shipped is a notch above the rest of the sector.

7.3 The disclaimer is the most honest copy in this product

This platform may not really be meant for MEMEs. The whitepaper talks about using CCA for RWA and tokenized-stock issuance, yet pools.trade's first risk-warning pop-up says: `"Any use of Pools for an asset other than a meme coin is prohibited. CCA is available for other types of assets, subject to other applicable terms and conditions."` — the Pools front end **Prohibited** for non-meme assets, while the CCA mechanism "may be available for other types of assets, subject to other applicable terms". Translation: memes go through this front end; serious assets go through another contract.

Three pieces of evidence point the same way:

1. **The whitepaper never mentions memes.** Its two goals are "price discovery" and "liquidity bootstrapping", and the authors are Hayden Adams and Ciamac Moallemi (a Columbia financial-engineering professor). That line-up doesn't write papers for meme coins.
2. **The disclaimer draws the boundary clearly**, see above.
3. **The factory is already being reused externally**: since launch, **722** crowd sales have used non-pools.trade configurations (custom supply, fee 3000 / 10000, custom LP recipient), going through the same CCA contracts rather than the pools.trade front end. To be accurate, though: all 722 are still meme sales (quoted in ETH or meme tokens such as TIRED, Bark, ROBINHOODS), with no RWA or tokenized-stock use case. So "CCA is reserved for serious assets" is so far only intent in the whitepaper and terms, with no on-chain evidence yet.

So this was just a stress test for pools.trade's debut — and retail paid the bill: of 2,263 crowd sales, 55 graduated; in the other 2,208, participants' money was genuinely tied up for the 4-hour window (refundable, though you have to sign a transaction to get it); and participants in the 5 graduated tokens with the largest raises are all underwater, −71% on average. In two weeks and a few thousand meme coins, Uniswap got a stress-test report on a pricing engine it can port to serious assets — and for the Web3 world as a whole, ETH bought by retail with real money took part in yet another public social experiment.

But in fairness: it said so up front. The disclaimer is the most honest piece of copy in the whole product — before your first order it tells you this isn't a mature product but an experiment explicitly confined to memes. Most launchpads don't even write that sentence.

7.4 If Crowd really took off, who would win most

ETH. Because the pools can't be withdrawn, if this model catches on, a steady stream of ETH gets locked up. And as everyone knows, most tokens eventually go to zero, except BTC and a few majors — Bitcoin stays, altcoins come and go; it's a negative-sum game from the start. Crypto really is a dark forest.

First the verifiable part: the **81.05 ETH SnowOn raised went 100% into an irrevocable single-range LP position** (on-chain `fundsRecipient` = LBPStrategy, `positionRecipient` = FeeSplitter; the creator can't get the LP), with current pool liquidity of $221.9K. To swap that ETH back out, the token price has to fall, and there's a hard cap: the 500M tokens in participants' hands exactly equal the pool reserve, so dumping them all recovers at most about half (~40.5 ETH), while the price falls to 25% of the clearing price (§2.9).

But "permanently locked" is more accurate when split into two layers:

PART	WHAT'S ACTUALLY LOCKED
ETH principal raised into the pool	**Not permanently destroyed, but permanently redistributed.** The process of a token going to zero is precisely the process of the pool's ETH being swapped out — the ETH will always return to the market, but to **whoever leaves first**, not to the original funders. The pool is just a machine that moves late entrants' money to early leavers.
Reinvested 60% of LP fees	**This part really is locked forever.** The front end says plainly "not part of your earnings and not claimable"; 6.993135 ETH accumulated in 14 days, it only grows, and nobody has the right to take it out.
4 bp protocol fee	Goes to Uniswap's TokenJar; spendable.

So the more accurate statement is: **the winners are "the earliest leavers + Uniswap's protocol fee"**; the only part truly burned forever is the reinvested 60%,

and it happens to be the only money that "everyone paid in and no one can get back" — the purest public good in this mechanism, and the purest sunk cost.

As for "Bitcoin stays, altcoins come and go", 14 days of data don't contradict it: **2,263 crowd sales, 55 graduated (2.43%), all 5 comparable graduated tokens now below their clearing price, −71% on average**; the best one is down to $6.5K of daily volume. "Negative-sum game" is accurate in the accounting sense — trading fees, protocol fees and the reinvested lock all flow out of participants' principal, with no outside cash flow coming in.

The one thing to say in its favour: compared with the old world where "the project can pull the pool and run at any time", permanent locking at least turns the path to zero into a public, continuous curve everyone can calculate in advance. It doesn't eliminate losses, but it makes them calculable — and calculability is a rare step forward in this industry.

8. Verdict on a three-layer experiment

LAYER	WHAT IT TESTS	VERDICT
Technical experiment	Can an auction with per-block clearing, constant gas, refunds and irrevocable liquidity be built on-chain	Success 144,000 blocks of continuous clearing ran stably; the inverse-price accumulator keeps exit cost O(1); the 13-step release table sums exactly to 1e7; the model reproduces the on-chain clearing price point by point. Very high engineering quality.
Product experiment	Do meme launchers and retail want to use this mechanism	Failure (Crowd) Daily launches −98.3%, graduation rate 2.43% (55/2,263), median bid just $20; sibling Instant Launch won 32:1. More importantly, after graduation: all 5 comparable graduated tokens **fell below their clearing price, −71% on average**. Mechanism superiority didn't turn into willingness to use it, because the problems it solves (sniping, fair distribution) rank lower for meme users than "buy and sell right now".
Business experiment	Can it produce meaningful revenue or a strategic asset	Unproven Only $25.7K of fees accumulated in 14 days, so no revenue case; but the strategic position was won (default AMM + issuance entry on the new chain). Real business value depends on whether CCA can be ported to non-meme assets.

In one sentence: **pools.trade is a technically beautiful experiment pointed at the wrong use case.** It uses an auction mechanism meant for pricing serious assets to solve fair launches for meme coins — and the meme market doesn't really want fairness; it wants speed and narrative. Put the other way, the mechanism's real value most likely isn't in the pools.trade front end but in the CCA factory itself: in two weeks and a few thousand meme coins, Uniswap stress-tested and validated a pricing engine that can issue more serious assets. Seen that way, this wasn't a social experiment but **infrastructure validation paid for with meme traffic** — and it worked.

Appendix

A. Key parameters and addresses

ITEM	VALUE
Chain	Robinhood Chain (Arbitrum Orbit), chainId 4663, ~100ms blocks → 864,000 blocks/day
RPC / explorer	`rpc.mainnet.chain.robinhood.com` · `robinhoodchain.blockscout.com`
LiquidityLauncher v3.2.0	`0x0000FffFBE8efE702c8703aE3477FF5dE3d319C0`
TokenFactory (UERC20)	`0x000000e200088d55c39a11f609e5f667729ad49b`
CCA Factory v2.1.0	`0x000000001F26a0044BaA66024e7b6599c61963F8`
LBPStrategy (in use)	`0x05d552391067389EE44fec3924157ed33F976000`
InstantLaunchStrategy	`0x23f8209572b4A1C2Ad88a42749E830791fb027F1` (fee ON) · `0xad44D55E7f8337c3ce113FBB591486e85bE104b2` (fee OFF)
PoolManager (v4)	`0x8366a39CC670B4001A1121B8F6A443A643e40951`
FeeSplitter / creator vault / reinvest	`0xeFF166AA…` · `0xd35E9CA7…` · `0xf9526Dd3…`
V4FeeAdapter / TokenJar	`0x6d0009504d129cf5002dba61d9ae8575aa79314c` · `0x2aC03e14Cfe755426DaAEe0a4994184Ce81482F8`
Crowd window / release table	144,000 blocks ≈ 4 hours; 13 steps, mps total 1e7; last block releases 30.01%
Crowd supply structure	1B total; 500M auctioned; 500M reserved for LP; if it doesn't graduate the reserve is burned to `0xdEaD`
Graduation threshold	Raise ≥ 5 × floor FDV (≈2.64–2.66 ETH ≈ $5,000, i.e. a $10K graduation FDV)
Instant starting price	`initialTick 198,050` → 2.507424e-9 ETH/token → FDV 2.507 ETH
Trading fee	LP 0.25% (`lpFee 2500`) + protocol 0.04% both ways (`protocolFee 1638800`) = effective 0.2899%
Creator share	40% of the ETH side of the LP fee (= 10 bps per buy, 0 on sells); the other 60% is reinvested into locked LP and can't be

ITEM	VALUE
	claimed

B. Definitions and what isn't covered

B1. Two easily confused fee bases. "Creator fee 5 bps" and "40% of the ETH side" are the same thing: the first is the average with half buys and half sells, the second the true figure on each buy (§2.8). Another circulating calculation treats "0.05% ÷ 0.25% = 20%" as the creator's split; that's a coincidence from dividing two different bases, not the 40/60 split executed on-chain.

B2. How graduations are counted. The front-end list doesn't show all graduated tokens, so counting from the list badly undercounts. This report's 55 graduations and 2,263 launches come from a full scan of on-chain events (§4.2, Appendix C).

B3. What this report doesn't cover. ① Whether pools.trade's trade routing charges an extra front-end interface fee was not checked. ② TokenJar's total accumulated protocol fees were not aggregated over the whole period. ③ For the 722 third-party crowd sales, 150 were sampled for quote currency and graduation status; reading `requiredCurrencyRaised` for every sale was not done. ④ Chain-wide volume can't be reliably derived from PoolManager Swap events (other asset pools contaminate the ETH side), so every volume figure here uses the front end's per-token readings.

C. Method and data sources

Every number in this report comes from one of four kinds of primary sources; anything quoting official wording is marked as "wording", not fact:

- **Verified contract source + constructor arguments**: via the Blockscout API `/api/v2/smart-contracts/{address}` to fetch `source_code` and `decoded_constructor_args`; core logic read from `ContinuousClearingAuction.sol`, `BidLib.sol`, `DemandLib.sol`, `InstantLaunchStrategy`, `V4FeeAdapter`.
- **On-chain state and events**: public RPC `eth_call / eth_getLogs / eth_getCode / extsload`; event signatures decoded by hand (`BidSubmitted`, `CheckpointUpdated`, `BidExited`, `InitializerCreated`, `Migrated`); the release table decoded 8 bytes at a time from the SSTORE2 bytecode that `pointer()` points to.
- **Front-end artifacts**: pools.trade's JS assets fetched directly, the i18n dictionary chunk located from `manifest`, and 776 `rhcca.*` strings extracted to compare

"product promises" with "contract implementation".

- **Official and media**: Uniswap official blog (2026-08-05), CCA whitepaper (2025-11), The Defiant (08-05 / 08-06), pools.trade front-end readings (2026-08-18). ETH at $1,900; all dollar figures are for reference only.

Full scans added this round (reproducible): using `eth_getLogs` to scan all `BidSubmitted` events from block 27,600,000 → 39,666,441 in 250K-block slices (filtered by topic0 only, any address), giving 19,668 bids; then an address-filtered scan of LBPStrategy's `InitializerCreated` / `Migrated`, decoding words 4, 5, 7 and 8 of the migration config (reserved supply / remaining-token recipient / pool fee / tickSpacing) to separate pools.trade-style sales from third-party integrations, and finally joining bids to sales by auction address. For third-party sales, a further 150 were sampled one by one with `eth_call` reading `currency / currencyRaised / isGraduated` to judge whether their graduation thresholds are comparable. Details are exported in the accompanying CSV.

Data snapshot: 2026-08-18. On-chain counts state their block ranges in the text. Model code and scenario results are in the accompanying CSV.