Tier 3 — Pump & Dump / Volatility Hunter Architektur

Erstellt: **2026-05-16 12:28 UTC** · Read-only Architektur-Recon · Quellen: `trading/tier3/`, `trading/news/listing_detector.py`, `trading/main.py`, `TIER3_VOLATILITY.md`

> **Risiko-Markierung:** Tier 3 ist der riskanteste Trading-Pfad des Bots. Operator-Doku spricht von Hit-Rate 70-85 %, Max-Loss -10 %, aber Pump-and-Dump-Patterns +50-200 % oder -50 % crashen in 1-4 Stunden. T3 hat ein *isoliertes Portfolio* (eigene Cash-Quote = 20 % des Gesamtkapitals) damit Verluste das T1/T2-Hauptkapital nicht torpedieren.

1. Was ist Tier 3?

Tier 3 ist der *Hunter*-Subsystem-Block des Bots: er sucht nach neuen Coin-Listings, frühen Pump.fun-Token-Launches, "Whale"-Käufen und Social-Trending-Signalen und versucht in den ersten Stunden eines neuen Tokens dabei zu sein.

Konzeptuell entkoppelt: Tier 3 läuft in einem **eigenen Portfolio** (eigene Cash-Quote, eigene Position-Liste, eigene Exit-Regeln). Damit kann das primäre T1/T2 Long-Term-Trading (Binance Spot-Pairs auf Testnet aktuell) sauber neben dem hochriskanten T3-Hunter laufen, ohne dass T3-Verluste T1/T2-Kapital zerstören.

Tier-1 vs Tier-2 vs Tier-3 — Definition aus Code & Docs

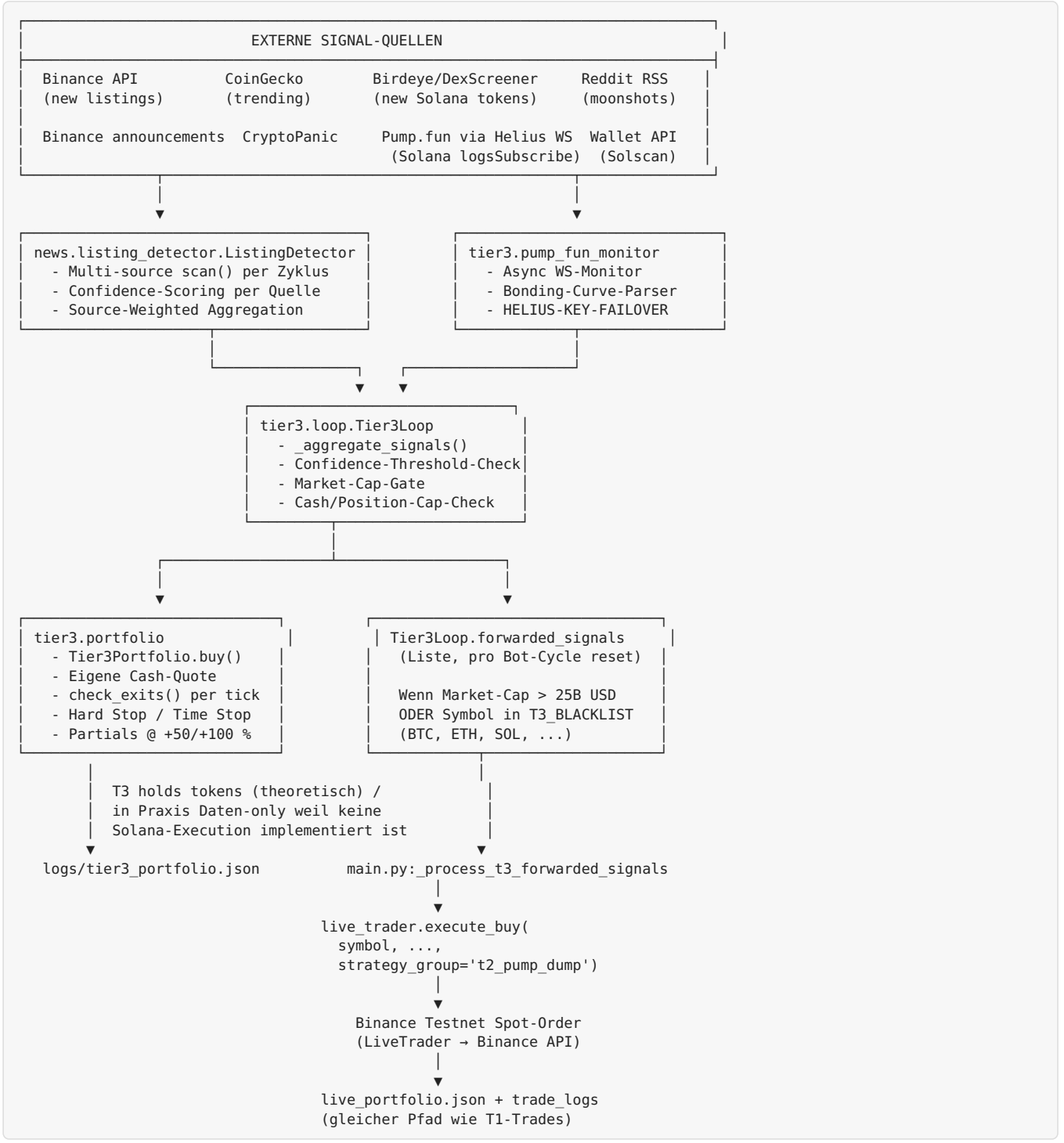
Tier	strategy_group (T-SPLIT-2)	Zweck	Execution
T1 Standard	`t1_core`	Klassisches Multi-Strategy + Legacy-Scanner-Loop (Score + News + Fear&Greed). Binance Spot-Pairs, hauptsächlich Large-Caps.	Binance Testnet (live) via `LiveTrader`
T2 Pump & Dump	`t2_pump_dump`	Tier-3-Forwarded-Signale, die wegen Market-Cap-Gate *doch* in T1/T2-Portfolio handelbar sind (Large-Caps wie SOL, ETH). Bleiben auf Binance Spot.	Binance Testnet via `LiveTrader` + T1/T2-Risk-Regeln
T3 Copy Trading (geplant)	`t3_copy_trading`	Geplant für native Solana / DEX-Execution via Jupiter. **Aktuell nicht implementiert.**	nicht aktiv

Anmerkung: der Code-Bezeichner *T3* in diesem Dokument bezeichnet das *Subsystem* in `trading/tier3/`. Das ist nicht identisch mit der *strategy_group* = *t3_copy_trading*, die ein T-SPLIT-2-Origin-Tag in der DB ist. Der T3-Subsystem-Code forwarded seine Signale an T1/T2 — also der *Empfänger* bekommt `strategy_group = t2_pump_dump` nicht `t3_copy_trading`.

2. Modul-Struktur

Datei	LoC	Klasse	Rolle
`tier3/loop.py`	~700	`Tier3Loop`	Orchestrator: hat Portfolio, ListingDetector, PriceFetcher, PumpFunMonitor, WalletTracker. Wird vom Bot bei jedem Scan-Cycle aufgerufen.
`tier3/portfolio.py`	~500	`Tier3Portfolio`	Isoliertes Tier-3-Portfolio (eigene Cash-Quote 20 % des Bot-Kapitals). buy / partial_sell / check_exits / state-persistence.
`tier3/price_fetcher.py`	~300	`Tier3PriceFetcher`	Preise via Birdeye / DexScreener für Solana-Tokens. *Wird heute live nur READ-only genutzt; T3-Subsystem führt KEINE Solana-Orders aus.*
`tier3/pump_fun_monitor.py`	~470	`PumpFunMonitor`	WebSocket-Monitor auf Solana mainnet via Helius RPC `logsSubscribe` auf Pump.fun-Programm. Bonding-Curve-Detection + Graduation-Detection. *HELIUS-KEY-FAILOVER aktiv seit 2026-05-16.*
`tier3/wallet_tracker.py`	~220	`WalletTracker`	Verfolgt Whale-Wallets (Token-Adressen). Heute **deaktiviert** via `COPY_TRADING_ENABLED=false`.
`tier3/helius_credits.py`	~50	—	Helius-Credit-Counter (Token-Verbrauch tracken).
`news/listing_detector.py`	~900	`ListingDetector`	Multi-Source-Scanner: Binance-New-Listings, CoinGecko-Trending, Birdeye-New-Tokens, Binance-Announcements, DexScreener-Boosts, Reddit-CryptoMoonShots, CryptoPanic-Hot, Pump.fun-Signals (vom Monitor), Wallet-Copy-Signals. Aggregiert in einheitliches Signal-Format.

3. Datenfluss (End-to-End)



4. Wichtige Erkenntnisse / Beobachtungen

4.1 T3 hat kein eigenes Solana-Execution-Modul

Obwohl T3 als "Volatility Hunter" für Solana-Token gedacht ist, führt der Code **keine echten Solana-Orders** aus. Der T3-Portfolio-Buy ist nur eine *simulierte* State-Mutation in `tier3_portfolio.json`. Die einzige reale Execution erfolgt für T3-Forwarded-Signale via `live_trader.execute_buy` auf Binance Spot — und nur wenn das Symbol auf Binance handelbar ist.

Beweis: keine `jupiter_swap`, `solana_buy`, `orca_swap` o.ä. Aufrufe im Code. `tier3.portfolio.buy()` schreibt nur den State.

> **Implikation:** T3 ist heute *Daten-/Signal-Layer*, kein Execution-Layer. Das Risiko-Statement aus der README ("Pump-and-Dump +50-200 %") gilt nur theoretisch — der Bot kauft die Pump-Tokens nicht auf Solana. Stattdessen: Wenn ein Pump-Token auf Binance Listed wird, forwarded T3 die Signal-Detection an T1/T2 und der Binance-Spot-Buy findet dort statt.

4.2 Market-Cap-Gate

`Tier3Loop._should_forward_to_t1` (loop.py:127): wenn das Symbol entweder

- in `T3_BLACKLIST` ist (BTC, ETH, SOL, BNB, ...) — Large-Caps
- *oder* Market-Cap > `T3_MAX_MARKET_CAP_USD` (Default 25 Mrd USD via CoinGecko-Lookup mit 1h-Cache)

→ Signal wird in `self.forwarded_signals` gepusht und vom Haupt-Bot als BUY-Kandidat an T1/T2 weitergereicht (mit

`strategy_group=t2_pump_dump`).

4.3 WebSocket-Monitor (Pump.fun via Helius)

`PumpFunMonitor` öffnet eine WS-Verbindung zu `wss://mainnet.helius-rpc.com/?api-key=…` und schickt eine JSON-RPC-Subscription:

```
{
  "jsonrpc":"2.0", "id": <N>,
  "method": "logsSubscribe",
  "params": [
    {"mentions":["6EF8rrecthR5Dkzon8Nwu78hRvfCKubJ14M5uBEwF6P"]},
    {"commitment":"processed"}
  ]
}
```

Die Adresse ist das Pump.fun-Solana-Programm. Jede Notification enthält Solana-Log-Lines, die der Monitor regex-/string-parsed für:

- **Bonding-Curve %**: aus Programm-Logs extrahiert, Threshold = `BC_THRESHOLD = 85 %` → "fast graduation-ready"
- **Graduation Event**: Migration aus Bonding-Curve zu Raydium/AMM-Pool

Heutiges **HELIUS-KEY-FAILOVER**: zwei Helius-Keys konfiguriert (`HELIUS_API_KEY` + `HELIUS_API_KEY_FALLBACK`), rotiert automatisch bei HTTP 429.

4.4 Confidence-Aggregation

`ListingDetector.scan()` ruft alle 9 Quellen ab und liefert eine Liste von Roh-Signalen `{symbol, confidence, source, reason}`. `Tier3Loop._aggregate_signals` gruppiert nach Symbol und kombiniert die confidences (höhere Wertung bei Multi-Source-Convergence).

Pro Quelle gibt es eine Max-Confidence-Deckelung:

- Reddit → max 0.45 (unvalidierte Posts)
- CoinGecko/Birdeye → 0.6-0.8
- Binance-Listing-Announcement → 0.85+
- Pump.fun (Graduation-Event) → 0.9
- Wallet-Tracker (Whale-Buy) → 0.95 (heute deaktiviert)

4.5 Exit-Polling Tighter-Loop

Hard-Stop / Time-Stop / Partials sind für Pump-Tokens kritisch, weil ein -16 % Crash in 1 Minute möglich ist. `Tier3Loop.check_exits_only()` wird alle 10s vom Main-Bot aufgerufen (zusätzlich zum 120s Scan-Cycle). Damit reagiert T3 viel schneller als der reguläre T1-Pfad (Stichwort ASPEN-Incident mit -16.8 % Slippage).

4.6 Signal-Quellen-Aktivität (heute)

Quelle	Status	Bemerkung
Binance new listings	aktiv	API-basiert, kein API-Key
CoinGecko Trending	aktiv	Free Public-API, 10-min Cache
Birdeye new Solana tokens	aktiv	Free tier
Binance announcements	aktiv	HTTP-Scraping mit Anti-Spam-Cache
DexScreener boosts	aktiv	Free Public-API
Reddit r/CryptoMoonShots RSS	aktiv	15-min Cache
CryptoPanic Hot	aktiv	Free tier
Pump.fun Bonding-Curve / Graduation	seit 12:19 UTC aktiv (HELIUS-KEY-FAILOVER)	vorher Tage offline wegen 429
Wallet-Tracker (Whale-Buy)	**deaktiviert**	`COPY_TRADING_ENABLED=false`
Twitter/X Trending	**deaktiviert**	API $100/Mo zu teuer

5. Gedanken & Anmerkungen

5.1 Sinn der aktuellen T3-Architektur

T3 ist primär ein *Signal-Multiplier* für T1/T2. Das macht Sinn, weil:

- T1/T2 läuft auf Binance Spot — geprüfte, regulierte Pairs
- T3 monitort den "Wilden Westen" der Solana-Memecoins
- Nur wenn ein Memecoin so groß ist dass er auf Binance Listed wird, kann der Bot tatsächlich handeln → genau das Filter-Pattern

Das isolierte T3-Portfolio (mit eigener buy/sell/exit-Logik) wirkt eher wie *Vorbereitungs-Infrastruktur für echte Solana-Execution in der Zukunft*. Heute ist es ein State-Tracker ohne realen Trading-Effekt auf Solana.

> **Gedanke:** Wenn T3 nicht real auf Solana traden soll und nur als T1/T2-Signal-Forwarder dienen soll, könnte

`tier3.portfolio` kapselt fast eliminiert werden (oder nur als "Last-Seen-Signals"-Cache überleben). Das würde den Code um ca. 700 LoC reduzieren. Aber: kostet kaum etwas heute, blockiert keine Funktion, und ist Vorbereitung für eine kommende Phase MH-1 / T3-EXECUTION.

5.2 Risiko-Punkte (current)

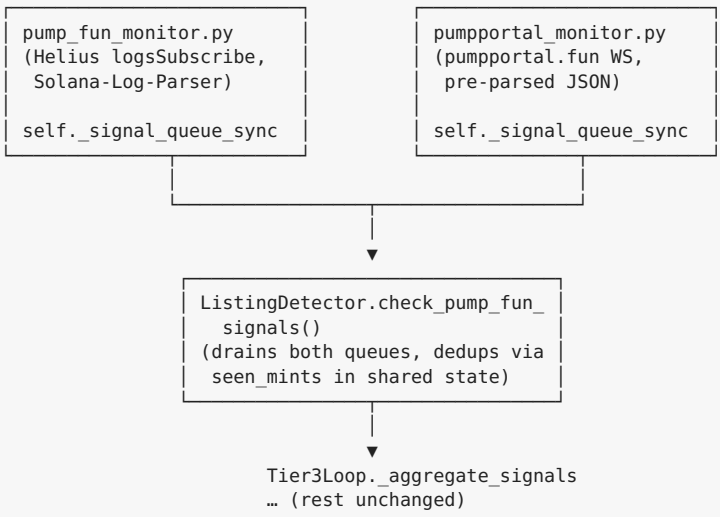
Risiko	Schwere	Status
Helius-Quota-Exhaustion	mittel	heute durch FAILOVER abgemildert; bei Doppel-Exhaustion → Pump.fun-Signals fehlen
Forwarded-Signal-Burst → Cap-Hit auf T1/T2	mittel	T1/T2 hat eigene `max_positions`-Logik; T3-Forwards landen auf der Warteschlange wenn voll
Reddit-Signal-False-Positive	niedrig	confidence max 0.45 + Multi-Source-Aggregation
Birdeye / DexScreener Rate-Limit	niedrig	15-min Cache built-in
CoinGecko Market-Cap-Lookup für Gate	niedrig	1h-Cache; bei Network-Fehler wird Signal NICHT forwarded (defensiv)
T3-Subsystem-Eigenzustand divergiert von Bot-Hauptzustand	theoretisch	keine geteilte State-Datei zwischen T3- und T1-Portfolio; isolierte JSONs

6. Mögliche Integration: pumpportal.fun / pumpdev.io

6.1 Warum sinnvoll

- **Redundanz**: heute kommt ein Pump.fun-Signal nur via Helius. Wenn beide Helius-Keys 429 zeigen → keine Pump.fun-Signale mehr (Birdeye-Fallback meldet andere Patterns, nicht das gleiche)
- **Pre-parsed Events**: pumpportal.fun liefert bereits gemappte JSON-Events (mint, marketCapSol, vSolInBondingCurve, …). Spart Solana-Log-Parsing-Code
- **Kostenfrei + ohne Auth**: `wss://pumpportal.fun/api/data` ist public, kein API-Key
- **Niedrigere Latenz** möglich (pumpportal.fun aggregiert direkt aus dem Solana-Mempool, oft 100-500 ms früher als logsSubscribe-Notification)

6.2 Empfohlene Architektur (Option D — parallel)



Vorteile: kein Single-Point-of-Failure. Wenn eine Quelle 429/down, läuft die andere weiter. Dedup via existing `seen_mints`-Set verhindert Doppel-Käufe.

6.3 Implementations-Skizze

Schritt	Datei	LoC	Risiko
1. `tier3/pumpportal_monitor.py` (NEU)	NEU	~120	niedrig — Spiegelt Struktur von `pump_fun_monitor.py` minus Log-Parsing
2. `tier3/loop.py`: zweiten Monitor instanziieren + parallel starten	edit	~10	niedrig
3. `news/listing_detector.py`: `check_pump_fun_signals()` drained BEIDE Queues (oder beide Monitore pushen auf gemeinsamen Queue)	edit	~5	niedrig
4. Env-Flag `PUMPPORTAL_ENABLED` (default false)	env	3 Zeilen	niedrig
5. Tests (Connect-Mock + Event-Parsing + Dedup-Verify)	NEU	~150	niedrig
6. SOT-1d Cutover (build + recreate clawbot)	—	—	niedrig (Worker untouched)

6.4 Provider-Wahl: pumpportal.fun vs pumpdev.io

Aspekt	pumpportal.fun	pumpdev.io
WebSocket-URL	`wss://pumpportal.fun/api/data`	`wss://pumpdev.io/ws`
Verbreitung	Standard in Solana-Tooling	Newcomer (Artikel-Quelle)
Subscribe-Methode	`{"method":"subscribeNewToken"}`	`{"method":"subscribeNewToken"}`
Event-Schema	mint, marketCapSol, vSolInBondingCurve, signature	identisch
Auth/Cost	Free / Public	Free / Public (Monetisierung über 0,25 % Trading-Commission, betrifft uns nicht)
Reliability-Historie	seit ~2024 stabil im Solana-Ecosystem	jünger, weniger Tracking-Daten

> **Empfehlung:** pumpportal.fun als Default-Provider, mit einem env-Flag `PUMP_PROVIDER=pumpportal|pumpdev` für einfachen Swap. API-Shape ist identisch — der gleiche Code kann beide bedienen.

6.5 Boundaries für die Integration

- 0× Mainnet-Trading (Tier 3 bleibt Daten-/Signal-Layer; tatsächliche Käufe via T1/T2 auf Binance Testnet)
- 0× DB-Migration (neue Quelle pusht in vorhandenen Signal-Queue)
- 0× Strategie-Parameter-Änderung (Confidence-Aggregation bleibt)
- 1× Bot-Recreate für die env + Code-Änderung
- 0× Solana-Wallet-Action (kein Private-Key, kein `sendTransaction`)

7. Reihenfolge für nächste Schritte

1. **Operator-Decision**:
   - Provider: pumpportal.fun (Default) oder pumpdev.io?
   - Default-State: `PUMPPORTAL_ENABLED=true` direkt aktivieren, oder erst false setzen + manuelles Toggle?
   - BC-Threshold-Filter im neuen Monitor: gleiche 85 % wie Helius, oder strenger weil pumpportal.fun mehr neuere Tokens meldet?
2. **Phase 1**: `pumpportal_monitor.py` implementieren + Tests
3. **Phase 2**: Loop-Integration + Dedup-Wiring
4. **Phase 3**: Cutover + Live-Verify (warten auf erstes parallel-getriggertes Pump.fun-Signal aus beiden Quellen → muss nur 1× in `seen_mints` landen, nicht doppelt forwarden)
5. **Phase 4**: 24h Beobachtung, dann Default-Flag aktivieren wenn stabil

8. Was NICHT Teil dieses Plans ist

- Echte Solana-Execution (Jupiter-Swap o.ä.) — separater Block, größere Phase
- Wallet-Tracker re-aktivieren (Operator-Pause bleibt)
- T3-eigenes Portfolio modernisieren / refactoren (würde Risiko erhöhen ohne klaren Use-Case)
- MH-1 / COMMAND-BUS-V6 / managed_proposal-Aktivierung (separate Phasen, blockiert per Operator-Boundary)

Erstellt read-only auf Basis von: `trading/tier3/loop.py`, `trading/tier3/portfolio.py`, `trading/tier3/pump_fun_monitor.py`, `trading/tier3/price_fetcher.py`, `trading/tier3/wallet_tracker.py`, `trading/news/listing_detector.py`, `trading/main.py`, `TIER3_VOLATILITY.md`. Keine Code-Mutation, keine Container-Aktion für diese Analyse.