T3-PUMP-1 Plan-Review

Birdeye-Fallback-Behauptung & reale T3-Source-Health

Datum: 2026-05-14 (UTC) | Phase: T3-PUMP-1 Plan-Review | Master HEAD: `d351a3a`
Vorgänger: T3-PUMP-FUN-WS-AUDIT (vorhergehende Phase, WARNING)

PLAN-REVIEW **READ-ONLY** **3 SILENT-FAIL-LÜCKEN** **NO-GO MAINNET BIS FIX**

> **Kurzfassung für eilige Leser.** Die Log-Zeile `„Fallback auf Birdeye aktiv"` in `pump_fun_monitor.py:290` ist ein **nackter Log-Text gefolgt von `return`** — kein Birdeye-Aufruf, kein Source-Switch, keine Statemachine. Parallel dazu zeigt der separate Birdeye-Adapter in `listing_detector.py` seit 2026-05-01 (= Free-Tier-Reset) **0 erfolgreiche Antworten** (CU-Counter `birdeye_cu_2026-05` fehlt komplett im State-File). Die T3-Quellen-Pyramide ist effektiv auf **CoinGecko Trending + Reddit + 1× DEX Screener heute** geschrumpft. **3 silent-fail-Lücken** wurden im Code identifiziert. Empfehlung: Implementierungs-GO nach Operator-Decisions; **NO-GO Mainnet** bevor Source-Health klar ist.

1. Boundary-Snapshot

Item	Wert
master HEAD	`d351a3a`
Bot main.py PID (in-container)	1196
Bot Container Host-PID	2419901, running healthy
Worker Container Host-PID	2420443, running unhealthy (False-Positive)
BINANCE_TESTNET	true
cmd 13 / managed_proposals / history	cancelled / 0 / 0
HELIUS_API_KEY	SET (boolean only, kein Wert)
BIRDEYE_API_KEY	SET (boolean only, kein Wert)

2. Code-Pfad „Fallback auf Birdeye aktiv"

Lokation

Quelldatei: `/projekte/Steve-TradingBot/trading/tier3/pump_fun_monitor.py`, Zeile **290**:

```
while self._reconnect_cnt <= MAX_RECONNECTS:
    try:
        await self._connect_and_listen(external_queue)
    except Exception as e:
        logger.warning(f"[WS] Verbindung getrennt: {e}")

    if self._reconnect_cnt >= MAX_RECONNECTS:
        self.ws_status = 'FAILED'
        logger.error(f"[WS] FAILED nach {MAX_RECONNECTS} Versuchen — Fallback auf Birdeye aktiv")
        if not self._alert_sent:
            self._send_failure_alert()  ← Telegram-Notification
            self._alert_sent = True
        return  ← Methode ist hier zu Ende. KEIN Birdeye-Call.
```

> **BEFUND #1 (Critical):** Diese Log-Meldung ist eine **Behauptung ohne Wirkung**. Nach dem `logger.error` + Telegram-Alert kommt `return`. Es wird *kein* Birdeye-Adapter aktiviert, kein Failover-Flag gesetzt, kein Source-Pool umgeschaltet. Die „Fallback auf Birdeye aktiv"-Meldung ist also genau das, was wir nach dem Audit befürchtet haben: ein **Log-Statement, das eine Funktionalität verspricht, die der Code nicht hat**.

Telegram-Alert

Quelldatei: `pump_fun_monitor.py:440-451`:

```
"🔴 <b>Pump.fun WebSocket FAILED</b>\n"
f"Nach {MAX_RECONNECTS} Reconnect-Versuchen aufgegeben.\n"
"Fallback: Birdeye + DEX Screener aktiv.\n"
"<i>Steve Trading Bot — Tier 3>"
```

Das ist die Meldung, die du im Telegram bekommen hast. Sie ist nicht falsch im engeren Sinne (Birdeye + DEX Screener sind *parallel laufende* Quellen im listing_detector, die unabhängig vom Pump.fun-WS arbeiten) — aber sie suggeriert einen aktiven *Failover*, der so nicht passiert.

Wichtige Nebenerkenntnisse

- **PUMP_FUN_ENABLED-Flag existiert bereits** (`pump_fun_monitor.py:123 + 275`). Mein Audit-Backlog-Item T3-PUMP-5 ist damit hinfällig.
- Pump.fun nutzt nicht Pump.fun-API direkt, sondern **Helius WebSocket RPC** (`self.helius_ws_url + ?api-key={self.helius_api_key}`). Das HTTP 429 kommt also von *Helius*, nicht von Pump.fun.
- Reconnect-Strategie sauber: Backoff [1,2,4,8,30,30,30,30,30,30]s, 10 Versuche, dann clean exit. *Kein* Endlosschleifen-Bug.
- Telegram-Spam ist via `_alert_sent`-Flag verhindert (1 Notification pro WS-Cycle, nicht 1 pro Reconnect-Versuch).

3. Birdeye-Adapter — wird real aufgerufen?

Lokation

Es gibt einen *separaten*, real implementierten Birdeye-Adapter in `/projekte/Steve-TradingBot/trading/news/listing_detector.py:259-...` mit Method `check_birdeye_new_tokens()`. Dieser wird im T3-Scan-Cycle alle 5 Minuten aufgerufen — **unabhängig vom Pump.fun-WS-Status**.

State-Machine

In `listing_detector.py:95-104`:

```
self._birdeye_status     = 'ONLINE'      # ONLINE | RATE_LIMITED | FALLBACK_L2 | FALLBACK_L3 | DEX_PAUSED
self._birdeye_retry_cnt  = 0
self._birdeye_alert_sent = False
```

BEFUND #2 (High): Diese Status-Variablen sind **nur in-memory**. Sie werden nicht in `self.state` (= persistentes JSON-File) gespeichert. Bei jedem Bot-Restart wird der Status auf `ONLINE` zurückgesetzt — selbst wenn Birdeye seit Wochen nichts liefert.

Code-Branches im 5-Minuten-Call

```
r = requests.get(BIRDEYE_BASE + '/defi/tokenlist', ...)

if r.status_code == 200:
    self.state['birdeye_cu_{month_key}'] += BIRDEYE_CU_PER_CALL
    self._birdeye_status = 'ONLINE'
elif r.status_code == 429:
    self._birdeye_retry_cnt += 1
    self._birdeye_status = 'FALLBACK_L2'
    logger.warning("🔀 Birdeye Failover → Stufe 2 (DEX Screener)")
    return []
# ← KEIN ELSE-BRANCH für 401/403/500/503/Timeout!
```

BEFUND #3 (High): Es gibt **keinen Branch für 401/403/500/503**. Antwortet Birdeye z.B. mit `401 Unauthorized` (API-Key invalid/expired/suspendiert), wird die `if`-Kette stillschweigend übergangen, `results` bleibt leer, kein Log, kein Status-Change, kein Alert. **Silent fail.**

Beweis im State-File

```
birdeye_cu_2026-03:  896
birdeye_cu_2026-04:  999
<birdeye_cu_2026-05 FELD FEHLT KOMPLETT>
last_birdeye_check:  2026-05-14T09:13:32.757271+00:00
```

Interpretation:

- März/April: erfolgreiche Calls (CU-Counter wuchs), aber sehr wenige (896 + 999 = 1895 CU von 30.000/Monat erlaubt). Birdeye wurde nicht intensiv genutzt.
- Mai: **0 erfolgreiche Calls** (Feld wird erst beim ersten 200-Response angelegt).
- Aber `last_birdeye_check` wird laufend aktualisiert — d.h. der Adapter wird *aufgerufen*, antwortet aber nie mit 200.
- Wenn jeder Mai-Call mit 429 antworten würde, würden wir „`🔀 Birdeye Failover → Stufe 2"`-Logs sehen (1×/5min). Die **letzten 109 Mentions dieses Logs sind alle vor 2026-04-28** — danach Stille. Was bedeutet: die Birdeye-API antwortet im Mai vermutlich mit etwas, das *keinen* Branch trifft — also 401 oder 403 oder Timeout.

4. T3-Source-Health (24h / 7d-Window aus aktivem bot_stdout.log)

Quelle	Aufrufe (Log-Total)	Echte Signale heute	Letzte Lieferung	Status
Pump.fun WS (Helius)	66 "verbunden"-Versuche	0	07:38 → sofort 429	DOWN
Birdeye API	0 Signal-Logs	0	vor 2026-05-01	SILENT FAIL
DEX Screener	340 Polls (Tokens-Header)	1 (00:16)	2026-05-14 00:16	MARGINAL
CoinGecko Trending	354 Polls	**16 Aggregate-Signale**	2026-05-14 09:03	ACTIVE
Reddit CryptoMoonShots	130 Polls	3 Ticker-Lieferungen	2026-05-14 08:13	ACTIVE
Binance ExchangeInfo	0 Logs heute	0	?	UNKLAR
@binanceannouncements	0 Logs heute	0	?	UNKLAR
@BinanceKillers (Web-Scraper)	—	2 Signale	2026-05-14 07:56	ACTIVE
Wallet Tracker (Copy-Trade)	—	—	—	DISABLED (COPY_TRADING_ENABLED=false)

Effektive Quellen-Pyramide heute: 3 von 8 dokumentierten Quellen liefern echte Daten (CoinGecko, Reddit, @BinanceKillers). DEX Screener marginal. Pump.fun + Birdeye komplett still. Binance-Quellen unklar. Wallet Tracker bewusst aus.

5. Root-Cause-Hypothese

Hypothese A — Birdeye-API-Key invalid/expired

Wahrscheinlichste Erklärung für den Mai-Silent-Fail. Birdeye Free-Tier-Accounts werden bei Inaktivität oder ToS-Verstoß suspendiert; manche Provider rotieren Keys bei Inaktivität. Antwort wäre 401/403, ohne dass der Code es bemerkt (Befund #3). Test wäre ein einmaliger Direct-Call (außerhalb Bot) mit dem konfigurierten Key.

Hypothese B — Free-Tier-CU-Reset hat den Tracker zerschossen

Unwahrscheinlich. Der Monats-Key `birdeye_cu_2026-05` würde beim ersten erfolgreichen Call angelegt — also egal, ob March/April-Counter noch da sind.

Hypothese C — Helius-API-Key auf IP-Blocklist (für Pump.fun-WS)

Wahrscheinlich für das Pump.fun-WS-Problem. HTTP 429 sofort beim Handshake spricht für IP-basiertes Rate-Limit auf Helius-Seite. Strato-OpenVZ-IPs landen häufig auf RPC-Provider-Blocklists. Kein interner Bot-Bug. Alternativen: Helius Plan auf Paid upgraden ODER andere Solana-RPC-Provider (Quicknode, Triton, GenesysGo) ODER eigene Solana-RPC.

Hypothese D — Pump.fun-API hat sich geändert

Möglich aber sekundär. Helius bietet WS-Subscribe auf das Pump.fun-Program; wenn Pump.fun das Program umzieht (`PUMP_FUN_PROGRAM`-Constant), würde der Subscribe ins Leere laufen — aber Helius würde dann eher leere Streams liefern, nicht HTTP 429.

6. Echte Fallback-Kette (Soll vs IST)

Ebene	Soll (laut Code-Kommentar)	Ist
L1 primär	Pump.fun WS via Helius	down (429)
L2 sekundär	Birdeye (im listing_detector parallel)	silent fail (vermutlich 401/403)
L3 tertiär	DEX Screener	läuft, aber nur 1 echtes Signal heute (Dedup-Filter sehr aggressiv)
Daueraktiv	CoinGecko, Reddit, Web-Scraper	läuft

Realität: T3 lebt aktuell von CoinGecko + Reddit + 1 Web-Scraper. Die „pump-spezifischen" Quellen (Pump.fun/Birdeye/DEX Screener) sind effektiv ausgefallen oder marginalisiert. Conf-Aggregation: alle Signale heute kommen mit nur *1 Quelle* → Conf 65–69% < T3-Threshold 80% → **Skip**. Defensives Verhalten — kein BUY/SELL aus diesem Setup ableitbar.

7. Operator-Entscheidungspunkte

1. **D1 — Birdeye-Key-Test:** Soll ich (in einer eigenen kleinen Read-only-Probe, außerhalb Bot, mit explizitem Operator-GO) einen einmaligen Direct-Call gegen `https://public-api.birdeye.so/defi/tokenlist` mit dem konfigurierten Key machen, um den HTTP-Status zu identifizieren (200/401/403/429)? Liefert Klarheit über Hypothese A.
2. **D2 — Helius-Plan:** Wartet auf Pump.fun-WS-Wiederbelebung — Helius-Free-Tier reicht bei dieser Last *nicht aus* (sofort 429). Soll Helius-Plan upgegradet werden (Paid ~$50-100/Mo) oder Provider gewechselt werden (Quicknode/Triton)? Oder Pump.fun bewusst deaktiviert via `PUMP_FUN_ENABLED=false`?
3. **D3 — Log-Korrektur jetzt vs später:** „Fallback auf Birdeye aktiv" ist faktisch falsch — soll der Text in einer minimal-PR korrigiert werden (1 String-Change) oder bleibt das bis zur größeren T3-Source-Health-Phase liegen?
4. **D4 — Backend-Telemetrie:** Soll eine Source-Health-Metrik in die DB geschrieben werden (z.B. `bot_statuses.metadata_json` oder neue Tabelle `t3_source_health`)? Wird Voraussetzung für T3-PUMP-4 Filament-Dashboard.
5. **D5 — Mainnet-Block:** Vor MH-7 Mainnet sollte die Source-Pyramide klar diversifiziert sein. Aktuell de facto Single-Source (CoinGecko Trending mit Conf ≤ 70%). NO-GO empfohlen bis Birdeye repariert oder explizit als „permanent off + ersetzt durch X" markiert ist.

8. Priorisiertes Fix-Backlog

ID	Prio	Beschreibung	Aufwand	Restart?
T3-PUMP-1A	**P1**	Birdeye-Adapter: `else`-Branch für 401/403/500/Timeout + persistenter `_birdeye_status` in state.json + alert pro Status-Change (entry/exit) statt nur „retry_cnt>=1". Closes Befund #2 + #3.	~60 min Code, ~20 min Tests	ja, Bot-Recreate
T3-PUMP-1B	**P1**	Birdeye-Diagnose: einmaliger Direct-Call gegen den konfigurierten Key + Operator-Entscheidung (Reaktivieren / Replacen / Disablen). Voraussetzung für 1A-Test.	10 min	nein
T3-PUMP-1C	P2	Log-Korrektur `pump_fun_monitor.py:290 + 447`: „Fallback auf Birdeye aktiv" → ehrliche Formulierung (z.B. „WS abgegeben — andere T3-Quellen laufen parallel weiter").	5 min	ja, Bot-Recreate (oder lazy bei nächstem Restart)
T3-PUMP-2	P2	Pump.fun-WS Cooldown nach 10× FAILED erhöhen (1 h statt sofort wieder beim nächsten `run()`) → verhindert Helius-IP-Block-Vertiefung. Idempotent mit T3-PUMP-1C.	20 min Code	ja
T3-PUMP-3	P3	Helius-Plan-Entscheidung: Paid / Provider-Wechsel / Pump.fun deaktivieren. Operator-Decision D2.	—	—
T3-PUMP-4	P3	T3-Source-Health-Filament-Dashboard (read-only): pro Quelle „letzter erfolgreicher Call", „Signale 24h/7d", Status-Badge. Frühwarnsystem für stille Ausfälle.	4-6h	nein (GUI-Container)
~~T3-	—	~~PUMP_FUN_ENABLED-Flag~~ — bereits implementiert (`pump_fun_monitor.py:123`). Streichen.	0	—

PUMP-5~~				

9. GO/NO-GO für Code-Phase

GO für T3-PUMP-1A + T3-PUMP-1C als gebündelten kleinen PR nach Operator-GO + nach Operator-Entscheidungen D1+D3. Voraussetzung: T3-PUMP-1B Diagnose ergibt klares Bild (Key valid? expired? rate-limited?). Aufwand: ~90 min Code + ~30 min Test + ~30s Bot-Recreate. Boundaries identisch zu SEC-1c-3b (kein Mainnet, kein DB-Migration, BINANCE_TESTNET=true).

HOLD für T3-PUMP-2 (Cooldown): sinnvoll, aber abhängig von D2 (Helius-Plan-Entscheidung). Wenn Helius-Plan oder Pump.fun-Off entschieden wird, ist T3-PUMP-2 obsolet.

BACKLOG T3-PUMP-3 + T3-PUMP-4: später, abhängig von Operator-Decision D2 + D4.

10. Mainnet-Relevanz

NO-GO Mainnet bis T3-Source-Pyramide diversifiziert ist. Aktuell läuft T3 effektiv auf einer Quelle (CoinGecko Trending, niedrige Conf 65-70%). Das schützt im Testnet durch defensive Threshold-Logik vor BUYs. Im Mainnet würde:

- Bei einer kurzfristig schweigenden CoinGecko (z.B. Rate-Limit) **0 Signale** ankommen → Bot wäre faktisch blind, statt diversifiziert.
- Stille Adapter-Fails (Befund #3) wären schlimmer als laute, weil keine Telegram-Notification dafür existiert.

Empfehlung: T3-PUMP-1A + 1B + 1C abschließen **vor** MH-7. T3-PUMP-4 Dashboard als ergänzendes Frühwarn-Instrument idealerweise auch.

11. Boundaries dieser Plan-Review-Phase

- 0× Code geschrieben
- 0× Container-Restart
- 0× docker cp
- 0× Mainnet
- 0× DB-Write
- 0× Secret-Output (Whitelist-Checks nur boolean-Output für API-Keys; nach Hygiene-Verschärfung 2026-05-14 08:53 UTC explizite Var-Listen)
- 0× `env` dumps
- 0× `docker compose config` ohne Filter
- 0× `/proc/*/environ` bulk dump

Erstellt: 2026-05-14 (UTC) · Phase: T3-PUMP-1 Plan-Review · Master HEAD: `d351a3a` · Vorgänger: T3-PUMP-FUN-WS-AUDIT · Backlog-Pin: `memory/t3_pump_1_plan_review_20260514.md`