

Measured Before Trusted

What our anomaly detectors catch, what they miss and how often they cry wolf, measured on synthetic agent-payment graphs with known answers and then checked against the whole live ledger. With two faults found in our own live rules and the corrections proposed.

Agentic Finance Graph — Research Note 03, version 1.0, 11 October 2026. N. Simonyan. Figures measured on 10 and 11 October 2026 from the validator reports of two synthetic runs (seeds 20261010 and 20261011), the weekly run of 11 October, and a read-only backtest over the live ledger; the generator, the validator and the reports are public at github.com/AgenticFinanceGraph/amlsim-agentic. The corrections described in section 7 shipped on 11 October 2026.

ABSTRACT. A detector that has never been run against data with known answers is a claim, not a measurement. We adapted IBM's AMLSim, a generator of synthetic bank transactions with laundering patterns hidden inside and a key that says where they are, to AI-agent wallets on Base: AMLSim's eight laundering typologies ported, thirteen agent-payment shapes added, each paired where it matters with a near-miss built *not* to be caught. We ran our production detectors over 2,813 and then 22,894 synthetic agents (66,900 and 652,664 transfers). Retry storms, bulk minting and dead endpoints were caught in full with no false alarms. The wallet-drain rule caught every drain but 40% of its alarms were first large purchases; a second test, whether the payment took 80% or more of what the wallet held at the block before, removed every false alarm on synthetic data and, on the live ledger, separated 39 true drains from 8 purchase-shaped payments among the 47 drain-shaped payments recorded since February. The near-misses exposed two faults in the live rules: storms that cross a calendar-hour boundary were invisible (the live rule reports 10 storm detections all time, a sliding ten-minute window 40), and every list was silently capped at 100 rows. Both corrections are measured here and held for the release. Across 314,826 agent-to-agent payments since the ledger began there is no cycle, no scatter-gather and no gather-scatter; the only layering shape the chain has shown is a same-amount fan-out (10 instances), which looks like fleet funding rather than laundering.

1. Why measure

Every figure this ledger publishes carries a definition and a measurement time. A detector is different from a figure: it is a rule that says *this shape is worth a human's attention*, and until today the only evidence that a rule did its job was that it fired on real data and a human agreed. That is one-sided. It cannot say what the rule missed, because nobody labels the misses, and it cannot say how often it would fire on honest behaviour, because the honest cases are not in the incident log either.

The standard answer in financial-crime detection is synthetic data with a key: generate a population of normal behaviour, inject patterns whose labels are known, run the detector, count. IBM's AMLSim [1] is the reference tool for banks. We adapted its design rather than its code (Apache-2.0, credited): accounts became ERC-8004 agents with one bound wallet each, banks

went away, cash in and cash out became funding from exchanges or treasuries, and AMLSim's single SAR flag became two labels, `expected_flags` (what a published rule should fire) and `is_suspicious` (whether the behaviour is harmful), because a bulk identity mint is a shape without harm and a slow drain is a harm without a shape. The generator writes AMLSim-style CSV and a labelled JSONL; the validator loads a run into a separate database in the ledger's own table shapes and imports the production detector modules unchanged, so what is measured is what runs every hour.

2. The shapes

AMLSim's eight typologies (fan-in, fan-out, cycle, bipartite, stack, random, scatter-gather, gather-scatter) are injected as the paper describes them, labelled suspicious, with no expected flag, because the ledger had no layering detector. Thirteen shapes are ours. The ones with a live rule: a *drain* (a payment at least 5x the wallet's previous largest, at least \$250, to an address never paid, after at least three earlier payments); a *retry storm* (fifty or more payments to one address inside ten minutes); a *mint burst* (twenty-five or more registrations by one minter inside an hour); an *endpoint streak* (five or more failed probes and never a success). Their near-misses: a drain under the floor, a drain to a known payee, a slow drain (emptied in four to six payments each under 5x), a storm spread over 25 to 50 minutes, a storm that crosses an hour boundary, a mint burst spread so that no hour reaches twenty-five, an endpoint that fails but answers once, and a large routing hop in and out of the wallet in one transaction, which must stay silent. Normal behaviour is AMLSim's six models plus the two shapes this ledger sees all day: many tiny x402 payments to a few sellers, and an occasional larger one-off purchase to a new address, legitimate and deliberately shaped like a drain.

3. The live rules, measured

Table 1. The four live rules over two synthetic runs. Hit/expected · false alarms · precision / recall. A hit is an injected instance the rule found; a false alarm is a detection on anything else.

Rule	2,813 agents	22,894 agents
drain_pattern	25/25 · 11 false · 69% / 100%	100/100 · 67 false · 60% / 100%
retry_storm	15/15 · 0 false · 100% / 100%	60/60 · 0 false · 100% / 100%
mint_burst	6/6 · 0 false · 100% / 100%	20/20 · 0 false · 100% / 100%
uri_timeout_streak	12/12 · 0 false · 100% / 100%	50/50 · 0 false · 100% / 100%

Every drain false alarm, in both runs, was the purchase shape: an agent that had only ever paid about \$3 a call buying something for \$260 to \$450 once. The rule's own published text says a legitimate one-off purchase looks the same as a drain. This is that sentence with a number on it: 40% of alarms at scale. Every near-miss stayed silent, which measures five blind spots by design rather than by accident. The liveness ladder's production `score()` agreed with the generator on all 2,813 agents, the pass-through rule on all 10 receipt cases, and the address-poisoning matcher on all 12 symbol and address cases and on 151 of 151 planted-then-paid losses end to end, with no false loss.

4. Two faults in the live rules

4.1 The calendar hour

The live storm rule groups payments by payer, payee and *calendar hour*, then asks whether fifty or more fell inside ten minutes. A storm of 56 payments that starts at minute 56 is split between two buckets and neither reaches fifty; a single further payment later in the same hour also defeats the "all within ten minutes" test. The synthetic near-miss was silent in 8 of 8 cases. On the live ledger, a sliding ten-minute window finds 40 storm detections where the live rule finds 10; 21 payer-payee pairs were never reported. The rule that shipped on 11 October reads *runs* rather than days: a run of payments with no gap over ten minutes, reported when some ten-minute span inside it holds fifty or more, keyed on the hour the run starts so that every detection the old rule logged keeps its id. On the live ledger it finds 45 storm detections from 28 payer-payee pairs, against 10 from 7; it passes the synthetic exam identically to the sliding window (every planted storm, every hour-straddling near-miss, no false alarm), including storms of 233, 195, 185 payments. Mint bursts have the same fault on a smaller scale: 88 minters by a sliding hour, 83 by the calendar hour.

4.2 The silent cap

Every live list ends with `LIMIT 100`. The 22,894-agent run injected 100 drains; on its first validation, with the generator as it stood that morning, the drain-shaped rows numbered 132, the list kept the newest 100 and the oldest 32 drains fell off without a word, which read, before we found it, as a recall of 68%. The functions now take a limit, the validator removes it, and the hourly run warns when a list fills, as the mint-burst query already did.

5. The candidate rules

Table 2. Candidate rules written against the synthetic data, measured the same way, not live. "Also caught" counts suspicious near-misses the live rule leaves silent and the candidate finds.

Candidate	2,813 agents	22,894 agents
retry storm, sliding 10-minute window (also caught: 30 hour-straddling storms)	15/15 · 0 false · 100% / 100%	60/60 · 0 false · 100% / 100%
mint burst, sliding 60-minute window	6/6 · 0 false · 100% / 100%	20/20 · 0 false · 100% / 100%
drain that emptied the wallet (took 80%+ of the balance held before it)	25/25 · 0 false · 100% / 100%	100/100 · 0 false · 100% / 100%
slow drain: 72-hour sum to one new payee (also caught: all 39 slow drains)	25/25 · 15 false · 63% / 100%	100/100 · 117 false · 46% / 100%
slow drain, wallet left near empty after the window	25/25 · 4 false · 86% / 100%	100/100 · 48 false · 68% / 100%
layering: cycle	8/8 · 3 false · 73% / 100%	25/25 · 2 false · 93% / 100%
layering: scatter-gather	4/4 · 1 false · 80% / 100%	15/15 · 0 false · 100% / 100%
layering: gather-scatter	4/4 · 0 false · 100% / 100%	13/15 · 4 false · 77% / 87%
layering: same-amount fan-in	6/6 · 0 false · 100% / 100%	20/20 · 0 false · 100% / 100%
layering: same-amount fan-out	6/6 · 0 false · 100% / 100%	20/20 · 0 false · 100% / 100%

The balance test reads, through an archive node, what the wallet held at the block before the payment. A drain takes what is there; a purchase leaves most of it. On the synthetic runs that single addition took the drain rule from 60% to 100% precision at unchanged recall. For slow drains the right denominator is not the balance before the first payment, because money can keep arriving during the window; the test instead reads what is *left* after the last payment, and a drained wallet ends near empty.

6. The same rules over the whole live ledger

Synthetic data measures the rules, not the chain. The second measurement runs the same candidate functions, read-only, over everything the ledger has stored since it began.

Table 3. Backtest over the live ledger, 2026-10-10. Drain-shaped payments are those the live rule would have opened with its 30-day window removed.

Measure	Value
Drain-shaped payments since the ledger began, and their value	47 · \$98,857
...that took 80% or more of what the wallet held at the block before	39
...that took less (purchase-shaped: the wallet kept most of its money)	8
...whose balance could not be read	0
Retry storms: live calendar-hour rule · sliding ten-minute window · the shipped run rule · payer-payee pairs (old rule · shipped rule)	10 · 40 · 45 · 7 · 28
Mint-burst minters: calendar hour · sliding hour · the shipped run rule	83 · 88 · 88
Slow-drain windows by the sum rule · of which a single payment · of which already a drain	156 · 65 · 47
...left near empty after the window (the test does not discriminate on the chain; see the reading below)	130
Agent-to-agent counted payments walked for layering	314,826
Cycles · scatter-gather · gather-scatter · same-amount fan-in · same-amount fan-out	0 · 0 · 0 · 0 · 10
Verified address-poisoning losses, and	2 · 0 (both below the \$250 floor; the poisoning detector is the one that counts them)

Measure	Value
whether the drain rule saw them	

Four readings. First, on the live ledger the drain shape is mostly real: of 47 payments the rule would have opened, 39 took 80% or more of what the wallet held, 23 of them the whole balance (a fraction of 1.000 at three decimals), and 8 left most of it, which is what a purchase does. The balance test turns a shape into a verdict with a reason, and the reason is on the chain; it does not say who emptied the wallet or why, and a wallet swept to its own treasury looks the same as one swept by a thief. Second, the storm under-count is not a synthetic artefact: 21 real pairs, the largest with 233 payments to one seller inside ten minutes, were never reported. Third, the slow-drain test that worked on synthetic data does not discriminate on the chain: 130 of 156 windows ended with the wallet near empty, because agent hot wallets are funded as they pay and run near zero as a matter of course; the synthetic population held balances and the real one does not. The slow-drain kind is therefore not ready, and the generator's population needs a hot-wallet model before it is measured again. Fourth, the layering that AMLSim exists to find has not happened between agent wallets on Base: zero cycles, zero scatter-gather, zero gather-scatter across 314,826 payments. The only structure present is one agent paying many others the same amount, 10 times, which is how a fleet is funded, not how money is hidden. We keep the layering detectors running and unpublished until the first real instance exists.

7. What we do not assert, and what changes

Precision and recall measured on synthetic shapes say how a rule behaves on those shapes. They say nothing about how often the shapes occur on the chain, and a shape the generator did not think of is not measured at all; the near-misses are the ones we thought of. The balance test depends on an archive node answering for the block before each payment; where it cannot, the detection stays a shape. The cycle detector's recall on the first validation of 10 October was 75% on the small run and 68% on the large one, for a reason found the same afternoon: the walk stopped at eight hops while AMLSim cycles have up to ten members. With the walk at twelve hops, which is the code that ships, the same runs re-validated on 10 and 11 October find 8 of 8 and 25 of 25 planted cycles at 73% and 93% precision; Table 2 shows the re-validation, and the 10 October reports are kept beside it in the repository. The weakest layering number left is gather-scatter recall on the large run, 87%, which is not explained.

What shipped on 11 October 2026: the storm and burst rules read runs with a peak (definitions `retry_storm.v2` and `mint_burst.v2`) and the 35 storm runs and 71 burst runs the old rules missed or split were added to the log, with the old rows kept as history; the drain rule carries the balance fraction as evidence on every detection (`drain_pattern.v2`) and, after a month of side-by-side numbers, as a filter; the same-amount fan-out is a public kind, `layering_fan`; lists warn when they fill; the slow-drain kind waits for a second test that holds on hot wallets (a variant written on 11 October reads only the ledger, requiring the new payee to have taken at least 80% of the wallet's outflow in the 72 hours, two or more payments and varying amounts: 10 of 10 planted slow drains with one false alarm on both the baseline and a half-hot-wallet population, and 44 live windows all time where the sum rule alone opens 156; those 44 are read by hand before

anything is published); the validator runs weekly and its table is published beside the rules. The generator, the validator, the reports and this note's inputs are in the public repository, so the measurement can be repeated by anyone, including on their own detectors.

References

1. IBM, *AMLSim: a multi-agent based simulator that generates synthetic banking transaction data together with a set of known money laundering patterns*, github.com/IBM/AMLSim (Apache-2.0).
2. Agentic Finance Graph, *amlsim-agentic*: generator, validator, configurations and the reports used here (examples/afg-2k and afg-20k, re-validated 11 October with the 10 October reports kept as report.2026-10-10; weekly-2026-10-11; hot-2026-10-11), github.com/AgenticFinanceGraph/amlsim-agentic (Apache-2.0), 10 and 11 October 2026.
3. Agentic Finance Graph, *Research Note 02: What the Counted Money Is*, version 1.2, 2 October 2026 (Section 8.4, address poisoning).
4. Agentic Finance Graph, detector definitions and published records, agenticfinancegraph.com/api/detectors.