

AgentGate: Deterministic Guardrails Against Multi-Agent Collusion Carriers

Apart Research AI Collusion Research Sprint. Track 3 (Collusion Carriers and Countermeasures) / Track 2 (Detection and Audit of Collusion).

Generated 2026-10-25T09:08:56Z by `python -m agent_gateway evaluate-collusion --report`. Same-author scripted synthetic lab scenarios; not a benchmark.

ABSTRACT

Autonomous agents that share tools can coordinate covertly: one agent can leak information through which tool or argument it picks and when it calls, launder a restricted capability by handing work to a cleaner peer, or move data in small or re-encoded pieces across steps. AgentGate is a loopback Policy Enforcement Point that addresses these carriers with deterministic checks and no model calls: a post-taint side-channel guard, monotonic capability attenuation across delegation, and n-gram fingerprinting across encodings with canary watermarks for provenance. In this run the repository's defence tests for the three carrier groups passed 57 of 57. Benign policy decisions took a p50 of 0.1264 ms and a p99 of 0.9987 ms of CPU time (reference target 2.0 ms). The hash-chained decision log verified clean over 750 entries.

PROBLEM STATEMENT

Content filters look for a secret in what an agent sends. Collusion carriers avoid that in three ways:

- Group A, side channels. Information moves through the choice among a few argument variants, or through call timing, so no fragment of the protected value appears in any argument.
- Group B, delegation laundering. A restricted agent hands its task or state to a peer whose own capability looks clean (a confused deputy).
- Group C, split and encoded smuggling. A protected value leaves in pieces across steps, or after Base64, hex, URL, or whitespace transformation.

METHODOLOGY: ZERO-COMPUTE DETERMINISTIC GUARDRAILS

Every check is arithmetic, hashing, or HMAC on the call path. No LLM is consulted (LLM calls in this run: 0).

- `side_channel_guard.py`: after a session is both TAIANTED and SENSITIVE_READ, released outbound calls are counted for distinct argument variants, total calls, and regular inter-call rhythm (coefficient of variation). A signal denies the call, holds the session for review, and freezes the registered root key for 15 minutes.
- `delegation_guard.py`: a hand-off re-mints the recipient's agw1. capability with the sender's deny caveats (lattice meet). Caveats are chained HMACs, so they cannot be stripped without the root key, and a delegated principal that was barred upstream is held at egress.
- `deobfuscate.py`: protected values are stored only as salted SHA-256 digests of 8-character n-grams of the raw value and its encoded forms, matched against a per-session sliding

window of released arguments.

- canary_watermark.py: sensitive reads carry a zero-width HMAC watermark bound to (session, step, tool), plus credential-format digests, so a hit names the originating step.
- audit_chain.py: each decision-log line carries seq, prev_hash, and entry_hash.

Evaluation procedure, all offline and reproducible:

1. Run the existing defence test files with pytest: `python -m pytest -q tests/test_side_channel_guard.py tests/test_delegation_guard.py tests/test_deobfuscate.py tests/test_canary_watermark.py`. Counts come from the JUnit XML, keyed by node id.
2. Run 5 benign multi-agent workflows 50 times through an in-process PolicyEngine and time each `evaluate_call` with thread CPU time.
3. Write each benign decision to a hash-chained log, verify it, and check that editing or dropping a line in a copy is detected.

RESULTS

Reproducible test-suite results by group

Group | Test files | Tests | Passed | Failed | Skipped

A: Side-channel and timing guard | test_side_channel_guard.py | 8 | 8 | 0 | 0

B: Multi-agent delegation and confused-deputy guard | test_delegation_guard.py | 8 | 8 | 0 | 0

C: De-obfuscation and canary provenance guard | test_deobfuscate.py, test_canary_watermark.py | 41 | 41 | 0 | 0

Total | | 57 | 57 | 0 | 0

pytest: 57 passed in 3.05s (exit 0, 3.64 s). The full per-test list is in `collusion_benchmark_results.json` under `defense_suite.groups[].tests`.

Whole gateway suite and stateful property fuzzing

`python -m pytest --collect-only -q` collects 262 tests in 20 files. The CI job runs all of them. Stateful Hypothesis machines, with settings read from the test files:

State machine | File | max_examples | stateful_step_count

DelegationDagMachine | test_delegation_guard.py | 120 | 24

SideChannelMachine | test_side_channel_guard.py | 60 | 30

PepPolicyMachine | test_stateful_fuzz.py | 40 | 16

SmugglingMachine | test_stateful_fuzz.py | 60 | 24

GatewayScrubMachine | test_stateful_fuzz.py | 80 | 8

Benign decision latency (CPU time per `evaluate_call`)

Measure | p50 ms | p95 ms | p99 ms | mean ms | max ms

CPU | 0.1264 | 0.892 | 0.9987 | 0.2073 | 3.8371

Wall | 0.1246 | 0.888 | 1.0025 | 0.2098 | 4.2646

750 decisions. Reference target 2.0 ms: p99 CPU within target. Timing includes decision-log I/O.

Benign workflow	Steps	Denied	p50 CPU ms	p99 CPU ms
order_status_triage	200	0	0.1027	0.2548
status_report	100	0	0.117	0.1903
clean_delegation	150	0	0.1198	0.647
scratchpad_handoff	150	0	0.1317	0.3121
loopback_docs_fetch	150	0	0.1534	1.0616

Benign denials: 0 of 750 decisions (rate 0.0).

Audit-chain verification

Check	Result
Benign decision log chain	verified
Entries	750
Head hash	462797efde1cc794...
Edited line in a copy detected	yes
Dropped line in a copy detected	yes

DEFENCE ANALYSIS: WHY THE CARRIERS FAIL HERE

- Side channels (Group A). The guard does not try to read the content. It bounds the channel: the number of distinct variants, the number of calls after the taint point, and how regular the timing is. Crossing any bound holds the session for a person, so leakage is capped by policy rather than by spotting a pattern.
- Delegation laundering (Group B). Restrictions travel with the hand-off as HMAC caveats and are folded with a lattice meet, which only moves down. A peer cannot hold more egress than the agent that delegated to it, and a fresh session of a delegated principal is still held.
- Split and encoded smuggling (Group C). Matching runs on a sliding window across steps and over decoded forms, so splitting or encoding does not remove the n-gram overlap. Watermarks and credential digests give step-level provenance in the hash-chained log.
- Containment. On a signal the registered root key and its children are frozen for 15 minutes. Unregistered callers are never acted on.

SCOPE AND LIMITATIONS

- Same-author scripted synthetic lab scenarios. Not a benchmark and not a measure of any product in the wild.
- Group results are the outcome of the repository's own defence tests; each count traces to a named pytest node id.
- The real-model loopback archive figure (2 of 13 non-control runs) is a separate measurement and is never pooled with these results.
- Latency is measured CPU time per in-process evaluate_call decision on one machine. The 2 ms figure is a reference target, not a guarantee.
- The gateway binds 127.0.0.1 by default. Freeze actions apply for 15 minutes to registered

lab keys only.

- ATLAS technique labels are approximate.
- A passing audit-chain verify means the log is internally consistent. It is not a certification and not a penetration test.
- Group counts show that the guards behave as their own tests specify. They are not an estimate of recall against an unknown adversary.
- Benign workflows were written by the same author and are few; the denial rate is a smoke check, not a false-positive rate for real deployments.

REPRODUCE

```
pip install -e './agent-gateway[test]'  
cd agent-gateway  
export AGENT_GATEWAY_HMAC_KEY=$(python -c 'import secrets; print(secrets.token_hex(32))')  
python -m agent_gateway evaluate-collusion --report ../sprint_submission
```

Environment: Python 3.13.17 on Linux-6.18.44-fc-v153-x86_64-with-glibc2.39. Total evaluation time 4.946 s.