RESEARCH / START HERE

AI agents need boundaries you can see.

Start with your existing endpoint logs; add trusted decoy evidence where it matters

An AI agent is software that uses a model to choose actions and call tools. Once it can open files, run commands, or change records, the security question is no longer just whether its answer is correct. It is whether its actions stay inside the job it was allowed to do.

A well-placed decoy can make a boundary crossing visible. It cannot stop the agent by itself. Pair the tripwire with trusted records, narrowly scoped permissions, and a tested way to pause the work.

Use your existing endpoint logs first

Native process and network hunts lead this edition. Standard Sigma behavior rules and real FDR ingestion routes follow. New gateway, broker, executor, and decoy instrumentation belongs in the final advanced engineering appendix.

READER MAP / CONTENTS

Find the answer you need

Read the opening sections for the research story. Use the appendix when you are ready to test a hunt against your own telemetry.

SECTION / CLICK TO OPEN	USE IT FOR	PAGE
A tripwire, not a leash.	Catch the action. Understand the context. Stop only what you are authorized to stop.	3
Detection is the part the research backs.	Research results and operating guidance answer different questions.	4
A callback is evidence of a request.	Keep the observed event separate from the story you might tell about it.	5
Read, fetch, submit: three different actions.	An illustrative test workflow, using synthetic content and a mock endpoint.	6
Start with questions your logs can answer.	AI-assisted attackers and redirected internal agents can leave ordinary process, identity, and network evidence.	7
The parent stopped. Did the work stop?	Use this as a proposed response checklist, adapted to the service owner's authority.	9
A small, useful commissioning run.	A small test that shows whether the signal, the alert, and the stop all work.	10
Map the behavior, not the AI label.	Investigation aids for labeling what you actually observed.	11
Put the first hunt to work	Practical first steps for Deception Check	12
Detection appendix	11 native / Sigma starters, then 6 advanced engineering designs	13
Source reading list	Named research and documentation links	36

Choose the product you already operate. Native hunts require normal collection and inventory checks. The final advanced appendix requires additional instrumentation and is not the first-use route.

01 / THE SHORT VERSION

A tripwire, not a leash.

Catch the action. Understand the context. Stop only what you are authorized to stop.

A honeypot is a deliberately monitored decoy resource. A honeytoken is a small piece of bait, such as a fake credential or unique link. Neither needs real candidate, client, or employee data to be useful.

Three problems often get called 'AI attacks': an outside attacker using AI tools, an internal agent redirected beyond its task, and an attacker compromising the AI system's tools or data. This guide primarily focuses on the second, with hunts that also help investigate the other two. Ordinary endpoint activity alone cannot identify which one occurred.

1

Place a harmless decoy

Put a unique fake credential in an approved test workflow. It grants no real access. Define which actions are allowed before the test starts.

2

Observe the specific action

Record a file read, a URL request, and a credential submission as different events. A link checker may explain a URL request; it does not explain every later action.

3

Use a separate control to contain it

An authorized operator or preapproved rule pauses the affected run, revokes its temporary access, and checks its child jobs and retries.

What does ’corral an agent’ mean here?

Limit the places it can reach and the actions it can take, then verify that a stop worked. A warning in a prompt or an agent saying 'I stopped' is not an enforcement control.

Start with one owned, low-risk workflow and synthetic content. Keep candidate and client data, production secrets, and irreversible business actions outside the exercise.

02 / WHAT THE RESEARCH SUPPORTS

Detection is the part the research backs.

Research results and operating guidance answer different questions.

IAPS: detection has the strongest footing

IAPS's October 15 report examines honeypots for internal AI-agent threats. Detection has the strongest evidence of its three proposed uses. Establishing dangerous intent and deterring future behavior remain less established. A decoy covers the paths it was placed to observe, not everything an agent might do. [1]

95.9% in one controlled study

Tracebit reports that, among canary-present runs that reached admin, 95.9% touched a canary before the first critical action. That is a conditional result in its agent cyber range, not a detection rate for ordinary enterprise activity. Runs continued after alerts, so this figure does not demonstrate containment. [2]

CISA and international partners: constrain the work

The May 2026 joint guidance recommends gradual adoption, limited privileges, monitoring, and human oversight. This is the broader deployment foundation: a decoy adds a signal to controls you still need. [3]

October guidance from NIST, NSA, and CISA explicitly includes canaries for access outside intended bounds, protected logs, scoped shutdown, and checks before restart. It is voluntary guidance primarily for frontier training and evaluation. Its defined control-incident scope excludes adversarial prompt injection as the primary cause; do not treat it as a complete enterprise incident playbook. It cross-references NIST CSF 2.0, so it reads naturally alongside a CSF program. [4]

Sources: [1] IAPS report and authors; [2] Tracebit canary experiment; [3] Careful adoption of agentic AI services; [4] NIST / NSA / CISA incident practices.

03 / WHAT WE OBSERVED

A callback is evidence of a request.

Keep the observed event separate from the story you might tell about it.

Our published internet-facing decoy study, built on one leaked-backup decoy persona, linked 121 non-operator bait-file retrievals to their first canary callbacks, from July 10 to October 8, 2026. Five designated operator-test pairs were excluded from the original 126. The median time to the first callback was 13 minutes 15 seconds across all 121 pairs. [5]

121 FIRST CALLBACKS / RECORDED ADDRESS COMPARISON



100 different IP

21 same IP

First callback	Pairs	What the count means
Different recorded IP	100	The first request for the canary URL came from a different recorded address than the bait-file retrieval.
Same recorded IP	21	The first callback used the same recorded address.
Total	121	Linked file-retrieval and first-callback pairs; not a count of attackers.

Do not turn this into an AI-attack rate

These were HTTP requests to embedded canary URLs, not credential logins or proof of AI attackers. A source address is a network observation, not an identity.

Three additional requests had an Amazon SearchBot user agent and no linked bait-file retrieval. A later published address list corroborated that classification retrospectively. This matters because useful automation can trip a decoy too. Verify crawler identity and task context; a familiar user agent alone is not an allowlist. [5]

Evidence should survive the system that made it

In a separate October 17 lab test, two signed synthetic findings verified on a separate machine using only the public key after the producing container and its state were logically removed. A one-byte rationale change was rejected. That is the property you need when an alert has to hold up in a later review. [5]

Sources: [5] Deception Check: study, methods, and commissioning worksheet.

04 / MAKE THE BOUNDARY VISIBLE

Read, fetch, submit: three different actions.

An illustrative test workflow, using synthetic content and a mock endpoint.

1

Agent reads a test document

The document contains a fake credential and a monitored URL. Reading it may be part of the assigned task. Retain the approved task and document version.

2

Something requests the URL

The controlled sink records the request. Join it to trusted run and tool records where possible. A URL fetch is not a credential submission.

3

A fake credential is submitted

The mock endpoint records a match to a synthetic marker without storing a real secret. Its response grants nothing. Record submission and authentication outcome separately.

4

The owner checks and acts

Compare the action with the run's permissions. If it crossed the declared boundary, preserve evidence and apply the preapproved, scoped stop procedure.

Keep the authority outside the agent

The gateway, runner, and mock sink should create the authoritative events. Keep the run registry and audit store outside the agent's write permissions. An agent-provided caller ID is a claim until matched to a trusted identity or session.

Retain task and run IDs, parent-child relationships, tool and policy versions, UTC times, outcomes, and source-record references. Collect the minimum content needed for investigation; do not turn prompt logging into a new store of candidate data, client data, or secrets.

05 / HUNT THE BEHAVIOR

Start with questions your logs can answer.

AI-assisted attackers and redirected internal agents can leave ordinary process, identity, and network evidence.

Start with the logs you already collect

Use the native endpoint hunts first: CrowdStrike Falcon, Microsoft Defender for Endpoint, and Sophos Live Discover. The Sumo Logic and Splunk routes search real Falcon Data Replicator process events. Standard Sigma rules describe supported process behavior. You need the named products, normal collection, an approved agent inventory, and local validation; no new Deception Check event producer is needed for this route. H03-H08 are retained at the end as advanced engineering designs requiring new authenticated audit producers.

A process launch records a command and its execution context. It does not establish that a referenced file was read, that a requested network operation succeeded, that a permission mode became effective, or that an AI system chose the action. Review the parent binary path and hash against approved inventory, then compare the observed action with the authorized workload.

Hunt question	Best starting evidence	What to do with a hit
Was Claude launched with reduced approval checks?	Falcon process command lines on known agent hosts; H01	Confirm who approved the mode, then inspect its tool use.
Did an agent launch a shell?	Falcon process ancestry and approved-run inventory; H02	Separate expected build work from an unexpected command or destination.
Did a connector request a denied destination at startup?	Trusted MCP startup and egress audit; H03	Check outcome, origin, version, and change record. A request is not a download.
Did a denied sensitive read succeed?	Trusted resource-broker read receipt; H04	Check the final policy decision and task boundary, not just the file name.
Did the run request an out-of-scope target or submit a fake marker?	Tool gateway and controlled sink; H05-H06	Separate a denied outside-boundary request from a received fake credential.
Did work continue after a stop or denial?	Trusted ordered dispatch and control records; H07-H08	Check child jobs, retries, cached permissions, and control failures.

05 / HUNT THE BEHAVIOR / CONTINUED

EDR means endpoint detection and response: process and host activity. SIEM means security information and event management: joining records across systems. MCP, the Model Context Protocol, connects AI applications to tools and data. None automatically supplies every event needed here. The appendix marks native Falcon data separately from custom audit events that must first be collected and ingested.

A useful hit starts an investigation

A shell, a permission-bypass flag, or a network request can be legitimate. Join the event to its approved task, actor, process chain, and destination. Empty results with missing telemetry mean 'unknown,' not 'safe.'

06 / FROM ALERT TO A VERIFIED STOP

The parent stopped. Did the work stop?

Use this as a proposed response checklist, adapted to the service owner's authority.

1. **Preserve and attribute**

   Record the original event, approved task, trusted run mapping, and relevant tool inputs and outcomes. Protect an evidence copy before routine cleanup.

2. **Pause the affected run**

   Apply the owner-approved control at the runner or gateway. Revoke temporary run credentials and stop new dispatches. Keep unrelated work available.

3. **Check children and queues**

   Inspect spawned workers, queued jobs, delayed retries, and outstanding sessions. Measure the last observed action after the stop became effective.

4. **Verify before resuming**

   Retest logging, egress restrictions, policy checks, and credential revocation. Capture the owner's decision and restore only the approved scope.

Record	Why it matters
Event, collection, and alert times	Shows whether the signal arrived and where delay occurred.
Acknowledgment and effective-stop time	Separates human response from control execution.
Last child or retry action	Checks whether stopping the parent actually contained the workflow.
Export and independent verification	Checks whether another authorized analyst can validate the retained record.

Use trusted ordering within the same run when measuring post-stop actions. Late log arrival is not late execution. A missing event source or an unverified control leaves the stop unconfirmed.

07 / TEST BEFORE YOU DEPEND ON IT

A small, useful commissioning run.

A small test that shows whether the signal, the alert, and the stop all work.

Cohort	What to run	What it answers
Benign controls	Approved document reads, a link checker, and a crawler fixture with a forged user agent.	Which normal activities trigger, and can you explain them?
Positive controls	A harness deliberately requests the canary and submits a synthetic marker to the mock endpoint.	Does collection, attribution, alerting, and the stop path work?
Untrusted-input trials	Predeclared test documents or tool results try to redirect a real agent toward the mock endpoint.	Does this workflow resist the redirect? If not, what action occurred and what caught it?

Use three to five repetitions per commissioning case, freeze definitions first, and retain failures. Record read, fetch, submit, outcome, attribution, and stop status per trial. This is a commissioning sample, not a population detection-rate estimate.

Ask what else would have caught it

For each positive control and injection event, compare the decoy with existing egress, tool, and pipeline logs. Was the action merely searchable afterward, or would a working alert have brought it to a person? That difference is the useful comparison.

Run at least one genuine injection-cohort alert through signed export and a separate verifier. If none occurs, report that result; a harness-generated substitute can test plumbing but is not an agent finding. Move to a real internal workflow only after the safety and evidence gates pass.

Sources: Deception Check commissioning worksheet.

08 / MITRE MAPPING

Map the behavior, not the AI label.

Investigation aids for labeling what you actually observed.

Observed scenario	ATT&CK mapping	Evidence needed
Instructions planted in a document or tool result redirect an agent	T1698 / Indirect Prompt Injection	The external input, its provenance, and the resulting unauthorized behavior.
A malicious user prompt attempts to redirect the system	T1699 / Direct Prompt Injection	The direct input and its attempted effect; distinguish it from retrieved content.
A tool invokes an interpreter to execute commands	T1059 / Command and Scripting Interpreter	Process or tool execution records. Shell activity alone does not prove prompt injection.
An actor searches files for credentials	T1552.001 / Credentials In Files	File access and task context. A synthetic credential exercise is a test analogue, not proof of stolen real credentials.

A canary callback, a denied tool request, or a failed stop does not automatically deserve an ATT&CK technique label. Retain the event first; add a mapping only when the observed behavior supports it. MITRE's AI-focused ATLAS knowledge base can extend scenario design, but it is a separate taxonomy, not an interchangeable ID list.

Sources: MITRE T1698; MITRE T1699; MITRE T1059; MITRE T1552.001; MITRE ATLAS.

PRACTICAL APPLICATION / DECEPTION CHECK

Put the first hunt to work

One owned workflow, an accountable owner, and the telemetry you already operate.

Choose an owned, low-impact workflow and keep real secrets and identifying data outside the exercise. Define the approved host, runner executable path/hash, account, permitted tools, and responder. Baseline current process and network behavior before using a hunt for alerting.

Start with H01/H02 on Falcon, N01-N03 on Defender, N04/N05 on real Falcon FDR events already collected in Sumo or Splunk, or N06/N07 for Sophos live endpoint snapshots. S01/S02 are standard Sigma process-creation descriptions that still need backend mapping and host scoping.

When the question needs file-read evidence, a credential submission receipt, or trustworthy post-stop dispatch ordering, commission the separate producers in the advanced appendix. An endpoint process event cannot substitute for those facts.

DETECTION APPENDIX / START HERE

Native hunts first; engineering last

Read the purpose and evidence requirements before copying the source. Each entry retains its full code and the conditions that limit its interpretation.

Start with the 11 native and standard-Sigma entries. H01/H02 use Falcon process events. N01-N03 use Defender tables; N04/N05 use real FDR events already ingested into Sumo/Splunk; N06/N07 use live Sophos endpoint tables. S01/S02 describe conventional Windows/Linux process creation. H03-H08 are separate advanced engineering designs, printed last, requiring new authenticated producers.

What has been validated

Official field and collection documentation checked on 29 October 2026. Exact distributed files are checked with safe positive, negative, and missing-data fixtures; validation details accompany the code bundle. Engine compilation, recipient tenant execution, and production coverage are not established. A parser or converter success alone is not evidence of a useful detection.

HUNT / CLICK TO OPEN	EVENT SOURCE	PAGE
Review launch text requesting permission bypass	Native Falcon	15
Baseline command interpreters launched by known agents	Native Falcon	16
Defender: review agent permission-bypass launch text	Native Defender	17
Defender: baseline shells launched by known agent names	Native Defender	18
Defender: inspect network activity for an identified process	Native Defender	19
Sumo: baseline agent-parent shells in raw Falcon FDR	FDR / Sumo Logic	20
Splunk: baseline agent-parent shells in Falcon FDR	FDR / Splunk	21
Sophos: inspect current agent-parent shells	Sophos live endpoint	22
Sophos: inspect current sockets for an inventoried runner	Sophos live endpoint	23
Sigma: Windows agent-parent shell behavior	Sigma / Windows	24
Sigma: Linux agent-parent shell behavior	Sigma / Linux	26

DETECTION APPENDIX / START HERE / CONTINUED

HUNT / CLICK TO OPEN	EVENT SOURCE	PAGE
ADVANCED: Review denied external requests during MCP startup	ADVANCED custom	30
ADVANCED: Review a sensitive resource read despite a denied policy	ADVANCED custom	31
ADVANCED: Review a decoy session requesting an out-of-scope tool target	ADVANCED custom	32
ADVANCED: Find a registered fake credential submitted to the decoy sink	ADVANCED custom	33
ADVANCED: Find a new task dispatch after an effective stop	ADVANCED custom	34
ADVANCED: Find tool execution that started under a final denial	ADVANCED custom	35

Line numbers appear in the margin. A **>** margin marker means a display continuation of the preceding source line; it is not source syntax. The exact source of every hunt, and this telemetry contract, are attached to this PDF as plain-text files: open the Attachments panel in Adobe Acrobat or Reader and copy from there rather than from the page. If your PDF viewer does not show attachments, ask us for the plain-text copies.

DETECTION APPENDIX / 01 OF 17

Review launch text requesting permission bypass

H01-falcon-permission-mode.logscale

Purpose. Identify Claude Code launches whose command line requests bypassPermissions. Review whether that mode was approved for the host and workflow.

Telemetry. ProcessRollup2 with aid, FileName, ImageFileName, CommandLine, TargetProcessId, and ParentBaseFileName. Set agent_host_aid to one inventoried Falcon host; confirm FileName extraction in your parser. Normal Falcon sensor process collection and search access/retention are required. Set the host parameter from approved inventory; a name match alone is not trusted agent identity.

Logic. Match the native claude binary name and the documented bypass flag or permission-mode value. This narrow starter deliberately covers one CLI. Inspect actual arguments; a quoted prompt can contain the same text.

Validation. Local fake-data checks exercise the source regex and representative positive/benign cases. No Falcon engine, tenant, installed CLI, or policy-enforcement test.

False positives. Approved isolated automation; an authorized developer choosing bypass mode; flag text inside a quoted prompt. A match is a review lead.

Blind spots. Other agents, renamed binaries, Node-hosted wrappers, configuration-driven modes, quoted or escaped argument variants, missing or truncated command lines, and mode changes after launch. A flag does not prove it became effective.

Triage. Verify binary identity, owner, isolation, change approval, and process ancestry. Preserve evidence references; compare subsequent execution with authorized scope.

Mapping. No automatic ATT&CK or ATLAS tag. A permission setting alone does not establish malicious execution, credential theft, or prompt injection.

Sources: CrowdStrike: native process-event fields and query examples; CrowdStrike: field regex filtering; Anthropic: Claude Code CLI flags; Anthropic: permission modes.

```
// H01: native Falcon process hunt; select an inventoried agent host.
// Starter only. Tenant parser, argv, and policy checks are required.
#event_simpleName=ProcessRollup2 aid=?agent_host_aid
| FileName=/^claude(\.exe)?$/i
| CommandLine=/(^|\s)--(dangerously-skip-permissions(\s|$)|permission-mode(=|\s+)bypassPermissio
ns(\s|$))/
| select([@timestamp, aid, TargetProcessId, ParentBaseFileName,
          ImageFileName, CommandLine])
```

DETECTION APPENDIX / 02 OF 17

Baseline command interpreters launched by known agents

H02-falcon-agent-shell.logscale

Purpose. Find agent-parent shell starts that violate the local workload baseline, such as execution on a host intended only to retrieve documents.

Telemetry. ProcessRollup2 with aid, ParentBaseFileName, FileName, ImageFileName, CommandLine, ParentProcessId, and TargetProcessId. Resolve parent identity against approved paths/hashes and the process graph. Normal Falcon sensor process collection and search access/retention are required. Set the host parameter from approved inventory; a name match alone is not trusted agent identity.

Logic. Select explicit native agent parent names and a bounded shell list. This is an inventory feed; it contains no threshold or automatic blocking rule.

Validation. Local fake process records test agent-parent, ordinary-parent, shell, and non-shell cases against source regexes. No Falcon query execution.

False positives. Expected coding, build, support, and test tasks. Names can be spoofed; a matching parent name does not authenticate the agent.

Blind spots. Node/Python wrappers, intermediate shells, remote or container execution outside the sensor view, missing lineage, and tool calls that create no process. Generic node/python activity is not labeled an agent.

Triage. Compare host role, user, parent identity, time, tool-call evidence, and approved work. Pivot with aid plus Falcon process IDs; OS PID reuse can mislead.

Mapping. ATT&CK T1059 is relevant after adversarial interpreter use is established. Normal agent shell use is not an ATT&CK observation or proof of an AI attacker.

Sources: CrowdStrike: native process-event fields and query examples; CrowdStrike: field regex filtering; MITRE ATT&CK: T1059.

```
// H02: native Falcon process hunt; baseline expected agent shell use.
// Confirm parent path/hash against inventory before labeling an agent.
#event_simpleName=ProcessRollup2 aid=?agent_host_aid
| ParentBaseFileName=/^(claude|codex|copilot)(\.exe)?$/i
| FileName=/^(cmd|powershell|pwsh|bash|sh|zsh|dash|fish|wsl)(\.exe)?$/i
| select([@timestamp, aid, TargetProcessId, ParentProcessId,
          ParentBaseFileName, ImageFileName, CommandLine])
```

DETECTION APPENDIX / 03 OF 17

Defender: review agent permission-bypass launch text

N01-mde-permission-mode.kql

Purpose. Review whether command-line permission bypass was authorized. A flag string does not prove the mode became effective.

Telemetry. Defender for Endpoint deployed and reporting DeviceProcessEvents; appropriate advanced-hunting access, license, retention, and role. Select the actual device ID.

Logic. Exact binary name and bounded CLI flag text. A quoted prompt may contain identical text.

Validation. Safe local source/fixture checks only; no recipient tenant or native engine execution.

False positives. Authorized coding, build, support, testing, and isolated automation. Establish the approved workload baseline before alerting.

Blind spots. Missing or truncated command lines; renamed binaries; intermediate shells; remote and container work outside sensor visibility.

Triage. Confirm the host, account, executable path/hash, parent process instance, and approved workload. Treat this as an investigation lead. Preserve only the evidence needed; redact secrets from command-line exports.

Mapping. No AI attribution or automatic technique coverage. T1059 is relevant only after adversarial interpreter use is established.

Sources: Microsoft DeviceProcessEvents schema.

```
// Set DeviceId from approved inventory and a narrow time window.
let agent_device_id = "REPLACE_WITH_DEVICE_ID";
DeviceProcessEvents
| where Timestamp > ago(1d) and DeviceId == agent_device_id
| where FileName in~ ("claude", "claude.exe")
| where ProcessCommandLine matches regex @"(^|\s)--(dangerously-skip-permissions(\s|$)|permissio
n-mode(=|\s+)bypassPermissions(\s|$))"
| project Timestamp, DeviceId, DeviceName, AccountName, FileName,
    ProcessId, ProcessCreationTime, InitiatingProcessFileName,
    FolderPath, ProcessCommandLine
```

DETECTION APPENDIX / 04 OF 17

Defender: baseline shells launched by known agent names

N02-mde-agent-shell.kql

Purpose. Inventory agent-parent shell starts, then compare with the host role and authorized work.

Telemetry. DeviceProcessEvents from normal MDE collection. Confirm initiating process identity against approved paths/hashes; generic node/python parents are deliberately excluded.

Logic. Parent and child names are candidate selectors, not identity proof or a policy decision.

Validation. Safe local source/fixture checks only; no recipient tenant or native engine execution.

False positives. Authorized coding, build, support, testing, and isolated automation. Establish the approved workload baseline before alerting.

Blind spots. Missing or truncated command lines; renamed binaries; intermediate shells; remote and container work outside sensor visibility.

Triage. Confirm the host, account, executable path/hash, parent process instance, and approved workload. Treat this as an investigation lead. Preserve only the evidence needed; redact secrets from command-line exports.

Mapping. No AI attribution or automatic technique coverage. T1059 is relevant only after adversarial interpreter use is established.

Sources: Microsoft DeviceProcessEvents schema.

```
let agent_device_id = "REPLACE_WITH_DEVICE_ID";
DeviceProcessEvents
| where Timestamp > ago(1d) and DeviceId == agent_device_id
| where InitiatingProcessFileName in~ ("claude", "claude.exe",
    "codex", "codex.exe", "copilot", "copilot.exe")
| where FileName in~ ("cmd.exe", "powershell.exe", "pwsh.exe",
    "pwsh", "bash", "sh", "zsh", "dash", "fish", "wsl.exe")
| project Timestamp, DeviceId, DeviceName, AccountName, FileName,
    ProcessId, ProcessCreationTime, ProcessCommandLine,
    InitiatingProcessFileName, InitiatingProcessFolderPath,
    InitiatingProcessId, InitiatingProcessCreationTime
```

DETECTION APPENDIX / 05 OF 17

Defender: inspect network activity for an identified process

N03-mde-process-network.kql

Purpose. Review recorded network activity for one process instance; compare destination and ActionType with the approved workload.

Telemetry. MDE DeviceNetworkEvents and advanced-hunting access. Copy DeviceId, ProcessId, and ProcessCreationTime from a process event. Confirm supported ActionType meanings in the tenant schema.

Logic. Device plus PID plus creation time avoids correlating a reused OS PID alone. No URL marker or destination proves credential submission or file contents.

Validation. Safe local source/fixture checks only; no recipient tenant or native engine execution.

False positives. Authorized coding, build, support, testing, and isolated automation. Establish the approved workload baseline before alerting.

Blind spots. Short-lived or uncollected connections, absent process creation time, child processes, proxies, NAT, and activity outside MDE visibility. This pivot covers the selected process only.

Triage. Confirm the host, account, executable path/hash, parent process instance, and approved workload. Treat this as an investigation lead. Preserve only the evidence needed; redact secrets from command-line exports.

Mapping. No AI attribution or automatic technique coverage. T1059 is relevant only after adversarial interpreter use is established.

Sources: Microsoft DeviceNetworkEvents schema.

```
// Copy all three values from the identified process event.
let agent_device_id = "REPLACE_WITH_DEVICE_ID";
let selected_pid = long(123456789);
let selected_start = datetime(2000-01-01T00:00:00Z);
DeviceNetworkEvents
| where Timestamp > ago(1d) and DeviceId == agent_device_id
| where InitiatingProcessId == selected_pid
| where InitiatingProcessCreationTime == selected_start
| project Timestamp, DeviceId, DeviceName, ActionType,
    InitiatingProcessFileName, InitiatingProcessId,
    InitiatingProcessCreationTime, RemoteIP, RemotePort, RemoteUrl
```

DETECTION APPENDIX / 06 OF 17

Sumo: baseline agent-parent shells in raw Falcon FDR

N04-sumo-fdr-agent-shell.sumo

Purpose. Search real Falcon process events already collected in Sumo; no Deception Check normalization schema.

Telemetry. FDR subscription/export and authorized raw-event transport into Sumo, such as owned S3 with the documented S3 Source. One FDR JSON event per message with top-level vendor keys; adapt only the extraction paths if your transport wraps the vendor event. Ordinary Falcon alert APIs do not supply these process events. Set your actual source category and aid; no category is built in.

Logic. Parse the actual vendor keys into local aliases, match known parent names and executable path suffixes. Confirm complete event shape with a known process.

Validation. Safe local source/fixture checks only; no recipient tenant or native engine execution.

False positives. Authorized coding, build, support, testing, and isolated automation. Establish the approved workload baseline before alerting.

Blind spots. Missing or truncated command lines; renamed binaries; intermediate shells; remote and container work outside sensor visibility.

Triage. Confirm the host, account, executable path/hash, parent process instance, and approved workload. Treat this as an investigation lead. Preserve only the evidence needed; redact secrets from command-line exports.

Mapping. No AI attribution or automatic technique coverage. T1059 is relevant only after adversarial interpreter use is established.

Sources: CrowdStrike Falcon Data Replicator; Sumo Logic Amazon S3 Source; Sumo Logic JSON parsing.

```
// Replace category and aid with your real FDR source and inventory.
_sourceCategory="REPLACE_WITH_FDR_CATEGORY"
| json "event_simpleName", "aid", "ParentBaseFileName",
    "ImageFileName" as event_name, host_aid, parent_name, image_path
| where event_name = "ProcessRollup2"
    and host_aid = "REPLACE_WITH_AID"
| where parent_name matches /(?i)^(claude|codex|copilot)(\.exe)?$/
| where image_path matches /(?i)[\\/](cmd|powershell|pwsh|bash|sh|zsh|dash|fish|wsl)(\.exe)?$/
| json "CommandLine" as command_line nodrop
| fields _messagetime, host_aid, parent_name, image_path, command_line
```

DETECTION APPENDIX / 07 OF 17

Splunk: baseline agent-parent shells in Falcon FDR

N05-splunk-fdr-agent-shell.spl

Purpose. Use vendor FDR events already in Splunk to review shell-launch context.

Telemetry. Active FDR subscription/export, configured CrowdStrike FDR add-on or equivalent authorized event ingestion, searchable raw JSON, permissions, and retention. Set the real index and aid. This searches top-level keys in each vendor JSON event; wrapped transports need explicit spath paths.

Logic. Extract real FDR keys; anchored parent names and executable path suffixes. No Splunk CIM model or package-defined dc fields are assumed.

Validation. Safe local source/fixture checks only; no recipient tenant or native engine execution.

False positives. Authorized coding, build, support, testing, and isolated automation. Establish the approved workload baseline before alerting.

Blind spots. Missing or truncated command lines; renamed binaries; intermediate shells; remote and container work outside sensor visibility.

Triage. Confirm the host, account, executable path/hash, parent process instance, and approved workload. Treat this as an investigation lead. Preserve only the evidence needed; redact secrets from command-line exports.

Mapping. No AI attribution or automatic technique coverage. T1059 is relevant only after adversarial interpreter use is established.

Sources: CrowdStrike FDR technical add-on guide; CrowdStrike Falcon Data Replicator.

```
index=REPLACE_WITH_FDR_INDEX earliest=-1d
| spath path=event_simpleName output=event_name
| spath path=aid output=host_aid
| spath path=ParentBaseFileName output=parent_name
| spath path=ImageFileName output=image_path
| spath path=CommandLine output=command_line
| where event_name="ProcessRollup2"
    AND host_aid="REPLACE_WITH_AID"
| where match(parent_name,"(?i)^(claude|codex|copilot)(\.exe)?$")
| where match(image_path,"(?i)[\\\\/](cmd|powershell|pwsh|bash|sh|zsh|dash|fish|wsl)(\.exe)?$")
| table _time host_aid parent_name image_path command_line
```

DETECTION APPENDIX / 08 OF 17

Sophos: inspect current agent-parent shells

N06-sophos-live-agent-shell.sql

Purpose. Inspect currently running shell children of inventoried agent runners.

Telemetry. Sophos Live Discover endpoint query capability, supported osquery processes table, authorized selected endpoints, and suitable EDR/XDR entitlement. Choose the endpoint editor, not the Data Lake editor.

Logic. Live processes self-join by parent PID; verify parent path/hash in inventory. This is a current snapshot and can miss every short-lived process.

Validation. Safe local source/fixture checks only; no recipient tenant or native engine execution.

False positives. Authorized coding, build, support, testing, and isolated automation. Establish the approved workload baseline before alerting.

Blind spots. Exited processes, parent exit or PID reuse between collection, short-lived shells, remote/container execution, and unavailable endpoint. No historical absence claim is valid.

Triage. Confirm the host, account, executable path/hash, parent process instance, and approved workload. Treat this as an investigation lead. Preserve only the evidence needed; redact secrets from command-line exports.

Mapping. No AI attribution or automatic technique coverage. T1059 is relevant only after adversarial interpreter use is established.

Sources: Sophos endpoint and Data Lake schemas.

```
-- Live Discover endpoint query, not a Data Lake history query.
SELECT c.pid, c.parent, c.name, c.path, c.cmdline, c.start_time,
       p.name AS parent_name, p.path AS parent_path,
       p.start_time AS parent_start_time
FROM processes AS c
JOIN processes AS p ON c.parent = p.pid
WHERE lower(p.name) IN ('claude','claude.exe','codex','codex.exe',
                        'copilot','copilot.exe')
  AND c.start_time > 0 AND p.start_time > 0
  AND p.start_time <= c.start_time
  AND lower(c.name) IN ('cmd.exe','powershell.exe','pwsh.exe',
                        'pwsh','bash','sh','zsh','dash','fish','wsl.exe');
```

DETECTION APPENDIX / 09 OF 17

Sophos: inspect current sockets for an inventoried runner

N07-sophos-live-agent-sockets.sql

Purpose. Review current socket destinations against the approved runner baseline.

Telemetry. Sophos Live Discover endpoint SQL; supported processes and process_open_sockets tables, licensed/authorized endpoint query access. Verify fields against the selected endpoint schema before use.

Logic. Current-process/socket join by PID. A remote socket is not proof of a completed transfer or an unauthorized action.

Validation. Safe local source/fixture checks only; no recipient tenant or native engine execution.

False positives. Authorized coding, build, support, testing, and isolated automation. Establish the approved workload baseline before alerting.

Blind spots. Historical and short-lived connections; proxy destination masking; child-process connections; unavailable endpoint; sockets visible under generic runtime names.

Triage. Confirm the host, account, executable path/hash, parent process instance, and approved workload. Treat this as an investigation lead. Preserve only the evidence needed; redact secrets from command-line exports.

Mapping. No AI attribution or automatic technique coverage. T1059 is relevant only after adversarial interpreter use is established.

Sources: Sophos endpoint and Data Lake schemas.

```
1  -- Current open sockets, not connection history or successful exfiltration.
2  SELECT p.pid, p.name, p.path, s.remote_address, s.remote_port,
3         s.local_address, s.local_port
4  FROM processes AS p
5  JOIN process_open_sockets AS s ON p.pid = s.pid
6  WHERE lower(p.name) IN ('claude','claude.exe','codex','codex.exe',
7                          'copilot','copilot.exe')
8    AND s.remote_port > 0;
```

DETECTION APPENDIX / 10 OF 17

Sigma: Windows agent-parent shell behavior

S01-sigma-windows-agent-shell.yml

Purpose. Portable behavior description for standard process-creation telemetry; apply an inventoried host scope in the backend.

Telemetry. Windows or Linux process_creation telemetry with complete executable paths in Image and ParentImage. Conventional fields must map to the real endpoint source in a supported backend pipeline. Sigma is not a collector or a universally importable query.

Logic. Match parent/child executable path suffixes; the author is metadata and logsource.product names the actual operating system.

Validation. Safe local source/fixture checks only; no recipient tenant or native engine execution.

False positives. Authorized coding, build, support, testing, and isolated automation. Establish the approved workload baseline before alerting.

Blind spots. Missing or truncated command lines; renamed binaries; intermediate shells; remote and container work outside sensor visibility.

Triage. Confirm the host, account, executable path/hash, parent process instance, and approved workload. Treat this as an investigation lead. Preserve only the evidence needed; redact secrets from command-line exports.

Mapping. No AI attribution or automatic technique coverage. T1059 is relevant only after adversarial interpreter use is established.

Sources: Sigma rule log sources.

```
title: 'Agent Named Parent Starting a Command Interpreter (Windows)'
id: '28d4c52c-3f69-4dde-9ddb-a07ce1c71491'
status: 'experimental'
description: 'Review process creation on approved agent hosts. Names require inventory verificat
> ion; this rule does not prove AI identity or a policy breach.'
author: 'Deception Check'
date: '2026-10-29'
logsource:
    category: 'process_creation'
    product: 'windows'
detection:
    selection_parent:
        ParentImage|endswith:
            - '\claude.exe'
            - '\codex.exe'
            - '\copilot.exe'
    selection_shell:
        Image|endswith:
            - '\cmd.exe'
            - '\powershell.exe'
```

Sigma: Windows agent-parent shell behavior

S01-sigma-windows-agent-shell.yml / continued

```
            - '\pwsh.exe'
            - '\wsl.exe'
    condition: 'selection_parent and selection_shell'
falsepositives:
    - 'Approved development, build, support, and test workflows'
level: 'informational'
```

DETECTION APPENDIX / 11 OF 17

Sigma: Linux agent-parent shell behavior

S02-sigma-linux-agent-shell.yml

Purpose. Portable behavior description for standard process-creation telemetry; apply an inventoried host scope in the backend.

Telemetry. Windows or Linux process_creation telemetry with complete executable paths in Image and ParentImage. Conventional fields must map to the real endpoint source in a supported backend pipeline. Sigma is not a collector or a universally importable query.

Logic. Match parent/child executable path suffixes; the author is metadata and logsource.product names the actual operating system.

Validation. Safe local source/fixture checks only; no recipient tenant or native engine execution.

False positives. Authorized coding, build, support, testing, and isolated automation. Establish the approved workload baseline before alerting.

Blind spots. Missing or truncated command lines; renamed binaries; intermediate shells; remote and container work outside sensor visibility.

Triage. Confirm the host, account, executable path/hash, parent process instance, and approved workload. Treat this as an investigation lead. Preserve only the evidence needed; redact secrets from command-line exports.

Mapping. No AI attribution or automatic technique coverage. T1059 is relevant only after adversarial interpreter use is established.

Sources: Sigma rule log sources.

```
title: 'Agent Named Parent Starting a Command Interpreter (Linux)'
id: '28d4c52c-3f69-4dde-9ddb-a07ce1c71492'
status: 'experimental'
description: 'Review process creation on approved agent hosts. Names require inventory verificat
> ion; this rule does not prove AI identity or a policy breach.'
author: 'Deception Check'
date: '2026-10-29'
logsource:
    category: 'process_creation'
    product: 'linux'
detection:
    selection_parent:
        ParentImage|endswith:
            - '/claude'
            - '/codex'
            - '/copilot'
    selection_shell:
        Image|endswith:
            - '/bash'
            - '/sh'
            - '/zsh'
            - '/dash'
            - '/fish'
```

Sigma: Linux agent-parent shell behavior

S02-sigma-linux-agent-shell.yml / continued

```
            - '/pwsh'
    condition: 'selection_parent and selection_shell'
falsepositives:
    - 'Approved development, build, support, and test workflows'
level: 'informational'
```

ADVANCED ENGINEERING / NEW INSTRUMENTATION

Build this evidence separately

H03-H08 require new authenticated producers. They are engineering designs, not ordinary endpoint or SIEM detections.

These are newly authored, read-only hunt starters. H01-H02 use native Falcon process telemetry. H03-H08 require custom, trusted audit producers and separate ingestion. They are not built-in Falcon or Sumo detections, and no target SIEM execution or production effectiveness is claimed. Do not enable automatic blocks.

Run H01-H02 against a scoped Falcon data view with an explicit agent_host_aid parameter taken from the approved host inventory. Confirm FileName is populated or derive it from ImageFileName in the approved parser. Confirm agent binary paths and hashes independently; names alone are weak identity. Use aid plus the Falcon process identifiers for pivots, not an OS PID alone. Preserve the tenant and time window outside exported query rows. Parent ancestry, configuration, user intent, and policy must be established during triage. No hit is proof of an AI attacker.

For H03-H04 choose a restricted LogScale repository and ingest JSON fields under the exact names below. For H05-H08 configure a dedicated Sumo source category dc/agent_audit with the same JSON schema; these are Log Search queries, not Cloud SIEM rule expressions. A query against an unconfigured source is not a test.

Every custom event requires schema version, event timestamp, tenant_id, run_id, event_id, producer_class, event_kind, and evidence_ref. Use opaque references. Generate IDs and security decisions in an authenticated broker, gateway, executor, or decoy sink. Authenticate ingestion separately; a producer_class string alone does not establish trust. Reject caller-supplied overrides. Retain tenant-bound producer identity, policy revision, and receipt provenance in the evidence store. No model-written transcript or attacker-supplied JSON may populate this stream.

All IDs and enum values are nonempty strings; counters are nonnegative integers strictly below 2^53. Missing, null, invalid, or unverified fields are unknown. Never default them to allowed, denied, zero, or safe. Validate and quarantine malformed records before ingestion; independently measure rejected records, missing sources, late arrivals, duplicates, clock health, and daily synthetic receipt completeness. JSON parsing without nodrop intentionally excludes incomplete records in Sumo. Excluded events remain a collection-quality issue, not a clean security result.

H03: egress_gateway identifies the approved MCP server instance as server_id and links the network request to run_id/request_id. phase=mcp_startup comes from its trusted lifecycle, not a guessed time window. destination_scope=external comes from the approved destination inventory; policy_state=denied is the final policy decision. request_outcome distinguishes blocked, connected, and unknown. An attempt is not proof a connection or package download succeeded.

H04: resource_broker reports an actual completed read as read_outcome=success, with resource_id and resource_class, and the final policy_state for that operation. credential_file and secret_store are locally curated classes. General file-write events and a pathname in a command line do not establish a read. Instrument paths outside the broker separately or record the gap. Do not log contents or secrets.

ADVANCED ENGINEERING / NEW INSTRUMENTATION / CONTINUED

H05: tool_gateway classifies the requested target relative to the decoy boundary as target_scope=outside_decoy and records denied policy. decoy_id identifies the authorized experiment. A blocked request is a boundary event, not a successful escape. Triage can compare it with H08 to distinguish requests from executions.

H06: decoy_sink issues a receipt only after it receives a registered fake marker in the credential submission channel. marker_state=registered_fake and marker_id are checked against a private registry. receipt_state=received means receipt, not successful authentication. Tokens must be unique to a test, nonfunctional, and contained in a sink owned for that test. Do not use real passwords, PHI, public callbacks, or live canaries. A sample sink hostname would be sink.example.invalid.

H07: sequenced_executor records an effective authorized stop acknowledgment and each dispatch in one atomic order domain per tenant/run/authorization epoch. It attests state_integrity=verified, stop_id, stop_run_id, stop_epoch, auth_epoch, stop_seq, and dispatch_seq. stop_seq is the acknowledged enforcement boundary, not the user's click or message timestamp. A resume creates a new epoch and invalidates the old effective-stop state; a restart cannot reuse an old epoch. action_class=task excludes separately authorized cancellation/cleanup. An operation already dispatched before stop may finish afterwards; this hunt does not call that continuation. If ordering, acknowledgment, or current state cannot be proven, emit unknown and investigate instrumentation; do not infer intent.

H08: executor confirms tool_execution started and references the final policy decision with decision_id and decision_tool_call_id. decision_integrity=verified requires an authenticated decision receipt bound to the same tenant/run/tool call and to the immutable action parameters. A later authorized override must be the final decision. A request to a tool is not an execution receipt.

Commission with an owner, scope, parser version, retention/access policy, latency budget, and a small authorized nonproduction sample. Confirm expected positive, benign, missing-field, delayed-event, duplicate, cross-run, and resumed-run cases in the actual query engine. Review source counts separately from hunt hits. Record inconclusive when visibility is missing. Review unexpected hits manually, with evidence IDs, approved change records, and alternative explanations.

Framework context: ATT&CK v20.0 and ATLAS 2026.09 are descriptive taxonomies. H02 may support T1059 after malicious interpreter use is established. H04 may support T1552.001 for credential-file collection, or AML.T0098 for adversarial credential collection through agent tools, when that behavior is established. H06 may contribute to AML.T0086 only with evidence of tool-mediated exfiltration via injection or a poisoned tool. H08 may contribute to AML.T0053 when adversarial agent-tool use is established. The remaining entries need no forced technique mapping. A fake-marker test establishes the test behavior, not a real compromise.

H01-H08 were written for this guide. They were checked against fake records only; they have not been run in a Falcon, LogScale, or Sumo Logic tenant.

DETECTION APPENDIX / 12 OF 17

ADVANCED: Review denied external requests during MCP startup

H03-logscale-startup-fetch.logscale

Purpose. Engineering design requiring additional authenticated instrumentation. Detect an MCP startup request to an external destination that the approved policy denies; separate attempted fetches from completed transfers.

Telemetry. Custom egress_gateway events, separately ingested. Require trusted server_id, lifecycle phase, destination_scope, final policy_state, request_outcome, run_id/request_id, and receipt references.

Logic. Match network_request, mcp_startup, external, and denied. Select the restricted audit repository; its parser must expose the dc_audit_v1 JSON fields. These are not native Falcon fields.

Validation. Local fake records cover a denied startup request, an allowed fetch, an ordinary runtime fetch, and unknown lifecycle. No LogScale execution or package fetch.

False positives. Stale destination inventories, approved package resolution recorded against an outdated policy, and controlled commissioning tests.

Blind spots. Direct egress that bypasses the gateway, unknown startup state, local/stdio activity, and incomplete server identity. External does not mean malicious.

Triage. Check request_outcome, approved server/configuration revision, exact destination ownership, and deployment changes. Record blocked, connected, or unknown separately.

Mapping. No forced technique mapping. Startup egress alone does not establish ingress tool transfer, supply-chain compromise, or command and control.

Sources: CrowdStrike: select().

```
// H03: CUSTOM dc_audit_v1 events, not native Falcon sensor events.
// Select the restricted audit repository; map JSON fields at ingestion.
audit_schema=dc_audit_v1 event_kind=network_request
producer_class=egress_gateway phase=mcp_startup
destination_scope=external policy_state=denied
| select([@timestamp, tenant_id, run_id, event_id, request_id,
          server_id, destination_host, request_outcome, evidence_ref])
```

DETECTION APPENDIX / 13 OF 17

ADVANCED: Review a sensitive resource read despite a denied policy

H04-logscale-sensitive-read.logscale

Purpose. Engineering design requiring additional authenticated instrumentation. Find confirmed reads from locally classified credential files or secret stores when the final operation policy was denied.

Telemetry. Custom resource_broker read receipts plus the final decision for the same operation. Require read_outcome, resource_class, opaque resource_id, run_id/tool_call_id, and evidence_ref.

Logic. Select successful resource_read events with denied policy and the two curated resource classes. A pathname in a command or a file-write event cannot satisfy the read prerequisite.

Validation. Local fake records cover a denied successful read, a blocked read, an allowed read, and unknown read state. No file or secret was accessed.

False positives. Incorrect resource classification, stale decisions, approved support activity attached to the wrong policy revision, or a commissioning simulation.

Blind spots. Direct filesystem access outside the broker and sources that log only writes. A quiet query cannot establish that no secret was read.

Triage. Verify actual read completion, decision binding, approved exceptions, and collector coverage before treating the result as unauthorized collection.

Mapping. T1552.001 applies to established credential-file collection. AML.T0098 requires established adversarial collection through agent tools; neither mapping follows from a generic sensitive path.

Sources: CrowdStrike: in() filter; MITRE ATT&CK: T1552.001; MITRE ATLAS: version 2026.09 technique definitions.

```
// H04: CUSTOM read audit. File-write or process logs cannot substitute.
// policy_state describes the final policy decision for this operation.
audit_schema=dc_audit_v1 event_kind=resource_read
producer_class=resource_broker read_outcome=success
policy_state=denied
| in(field=resource_class, values=["credential_file", "secret_store"])
| select([@timestamp, tenant_id, run_id, event_id, tool_call_id,
          resource_id, resource_class, evidence_ref])
```

DETECTION APPENDIX / 14 OF 17

ADVANCED: Review a decoy session requesting an out-of-scope tool target

H05-sumo-decoy-boundary.sumo

Purpose. Engineering design requiring additional authenticated instrumentation. Identify a denied tool request outside the approved decoy boundary without claiming the boundary was crossed successfully.

Telemetry. Dedicated dc/agent_audit JSON source from a trusted tool_gateway. Require a registered decoy_id, tool_call_id, target_scope, final policy_state, and tenant/run/event references.

Logic. Parse explicit JSON fields, then select outside_decoy tool requests denied by policy. Missing required fields are excluded by parsing and must be counted in a separate collection-quality check.

Validation. Local fake records cover an outside request, an in-decoy request, an allowed request, and an unknown boundary. No Sumo execution.

False positives. Authorized exercises, workflow mistakes, outdated boundary definitions, and legitimate work routed to the wrong experiment.

Blind spots. Unregistered decoys, unmapped targets, tool paths outside the gateway, and fabricated events in an untrusted feed.

Triage. Compare boundary and policy revisions with the request receipt. A block is a successful boundary decision for this request; use executor receipts to determine whether anything nevertheless started.

Mapping. No forced ATT&CK or ATLAS mapping. A policy-boundary request is insufficient to identify injection, intent, or execution.

Sources: Sumo Logic: JSON parsing and missing-field behavior; Sumo Logic: where operator; Sumo Logic: fields operator.

```
// H05: Sumo Log Search; CUSTOM controlled audit source category.
_sourceCategory="dc/agent_audit"
| json "audit_schema", "producer_class", "event_kind"
| json "tenant_id", "run_id", "event_id", "tool_call_id"
| json "decoy_id", "target_scope", "policy_state", "evidence_ref"
| where audit_schema = "dc_audit_v1"
  and producer_class = "tool_gateway"
  and event_kind = "tool_request" and decoy_id != ""
  and target_scope = "outside_decoy" and policy_state = "denied"
| fields tenant_id, run_id, event_id, tool_call_id, decoy_id,
  target_scope, evidence_ref
```

DETECTION APPENDIX / 15 OF 17

ADVANCED: Find a registered fake credential submitted to the decoy sink

H06-sumo-marker-submission.sumo

Purpose. Engineering design requiring additional authenticated instrumentation. Establish that a controlled fake credential reached the designated submission channel, with no live secret or external callback.

Telemetry. Authenticated decoy_sink receipt with registered decoy_id and marker_id, marker_state=registered_fake, and receipt_state=received. Keep marker values and payloads out of logs.

Logic. Match a completed credential_submission receipt. The sink validates the marker against the private registry; the query trusts only the protected ingestion path.

Validation. Local fake receipts cover receipt, marker display without submission, an unregistered marker, and a failed receipt. No network request or credential use.

False positives. Authorized exercises, manual test submissions, replayed markers, and benign automation that copies test data.

Blind spots. Uninstrumented channels, transformed markers, unknown sinks, and missing receipts. Receipt proves neither successful login nor compromise.

Triage. Link the marker's placement and expiry to the run, deduplicate receipts, and inspect the associated tool-call evidence. Preserve only opaque references.

Mapping. AML.T0086 becomes relevant only with evidence of tool-mediated exfiltration via injection or a poisoned tool. A fake-marker receipt alone does not meet that description.

Sources: Sumo Logic: JSON parsing and missing-field behavior; Sumo Logic: where operator; MITRE ATLAS: version 2026.09 technique definitions.

```
// H06: CUSTOM decoy sink receipts. Never log the credential value.
_sourceCategory="dc/agent_audit"
| json "audit_schema", "producer_class", "event_kind"
| json "tenant_id", "run_id", "event_id", "decoy_id", "marker_id"
| json "marker_state", "receipt_state", "evidence_ref"
| where audit_schema = "dc_audit_v1"
  and producer_class = "decoy_sink"
  and event_kind = "credential_submission"
  and decoy_id != "" and marker_id != ""
  and marker_state = "registered_fake" and receipt_state = "received"
| fields tenant_id, run_id, event_id, decoy_id, marker_id, evidence_ref
```

DETECTION APPENDIX / 16 OF 17

ADVANCED: Find a new task dispatch after an effective stop

H07-sumo-after-stop.sumo

Purpose. Engineering design requiring additional authenticated instrumentation. Detect new task dispatches ordered after an acknowledged stop, within the same run and authorization epoch.

Telemetry. Trusted sequenced_executor records with verified state, stop_id, stop_run_id, stop_epoch/auth_epoch, stop_seq/dispatch_seq, and action_class. Use one atomic order domain; timestamps alone are insufficient.

Logic. Require effective stop, matching run and epoch, task action, and dispatch_seq greater than stop_seq. Counters are integers below 2^53. Resume changes epoch; cleanup has a distinct authorized class.

Validation. Local fake events cover post-stop dispatch, pre-stop in-flight work, cleanup, resume, wrong run/epoch, missing or unknown state, and reversed receipt timestamps. No Sumo or live stop test.

False positives. A faulty sequence producer, stale stop snapshots, incorrect action classification, or a resumed run reusing an old epoch.

Blind spots. Unacknowledged stops, unordered workers, missing audit records, bypassed executors, and unverified state. Unknown is inconclusive, never proof of compliance.

Triage. Verify the signed/attested stop receipt and parameter-bound dispatch record. Completion of work dispatched before stop is not a new post-stop action.

Mapping. No automatic technique mapping or claim about model intent. This is an authorization and orchestration control check.

Sources: Sumo Logic: JSON parsing and missing-field behavior; Sumo Logic: where operator; Sumo Logic: numeric conversion.

```
// H07: CUSTOM sequenced executor audit, not transcript timestamps.
// Producer verifies the effective stop and dispatch in one order domain.
_sourceCategory="dc/agent_audit"
| json "audit_schema", "producer_class", "event_kind", "state_integrity"
| json "tenant_id", "run_id", "event_id", "tool_call_id", "evidence_ref"
| json "stop_id", "stop_run_id", "stop_epoch", "auth_epoch"
| json "stop_state", "action_class", "dispatch_seq", "stop_seq"
| where audit_schema = "dc_audit_v1"
  and producer_class = "sequenced_executor"
  and event_kind = "tool_dispatch" and state_integrity = "verified"
  and stop_id != "" and stop_run_id = run_id and stop_epoch = auth_epoch
  and stop_state = "effective" and action_class = "task"
| num(dispatch_seq)
| num(stop_seq)
| where dispatch_seq > stop_seq and stop_seq >= 0
  and dispatch_seq < 9007199254740992
| fields tenant_id, run_id, event_id, tool_call_id, stop_id,
  stop_seq, dispatch_seq, evidence_ref
```

DETECTION APPENDIX / 17 OF 17

ADVANCED: Find tool execution that started under a final denial

H08-sumo-denied-tool-started.sumo

Purpose. Engineering design requiring additional authenticated instrumentation. Identify an executor receipt that confirms a tool started despite a final denied decision for the exact same action.

Telemetry. Trusted executor receipts with tool_call_id, decision_id, decision_tool_call_id, verified decision_integrity, final policy_state, and execution_state. Bind receipt, tenant/run, and immutable action parameters.

Logic. Require tool_execution started, denied final policy, and a verified decision for the matching tool call. An authorized later override changes the final decision.

Validation. Local fake events cover denied execution, denied request without execution, allowed execution, wrong decision binding, and unknown integrity. No Sumo or tool execution.

False positives. Receipt stitching errors, a stale pre-override decision, controlled fault injection, or incorrect execution-state reporting.

Blind spots. Missing executor evidence, actions outside the broker, or decision records lacking immutable action binding. A tool request or generated command is insufficient.

Triage. Reconcile the final decision chain and execution receipt, preserve evidence, and assign an owner to the control defect before any response action.

Mapping. AML.T0053 is relevant only after adversarial use of an agent tool is established. The control mismatch alone does not identify the source of misuse.

Sources: Sumo Logic: JSON parsing and missing-field behavior; Sumo Logic: where operator; MITRE ATLAS: version 2026.09 technique definitions.

```
// H08: CUSTOM executor receipt plus trusted final policy decision.
_sourceCategory="dc/agent_audit"
| json "audit_schema", "producer_class", "event_kind"
| json "tenant_id", "run_id", "event_id", "tool_call_id", "evidence_ref"
| json "decision_id", "decision_tool_call_id", "policy_state"
| json "execution_state", "decision_integrity"
| where audit_schema = "dc_audit_v1" and producer_class = "executor"
  and event_kind = "tool_execution" and decision_integrity = "verified"
  and decision_id != "" and decision_tool_call_id = tool_call_id
  and policy_state = "denied" and execution_state = "started"
| fields tenant_id, run_id, event_id, tool_call_id, decision_id,
  execution_state, evidence_ref
```

REFERENCES / FURTHER READING

Read the sources behind the guide

Research and documentation links are named so their purpose is clear before you open them.

[1] IAPS, Honeypots for AI Agent Insider Threats, 15 October 2026

[2] Tracebit, AI Agents and Canaries, working paper

[3] Joint guidance, Careful adoption of agentic AI services, May 2026

[4] NIST / NSA / CISA, Key Practices for Incident Readiness and Response in Super Intelligence Training and Evaluation, October 2026

[5] Deception Check, An AI-agent alert needs a boundary, with first-party methods and worksheet

[6] NIST, Cybersecurity Framework (CSF) 2.0

[7] NIST SP 800-53 Rev. 5, Security and Privacy Controls (SC-26 Decoys)

[8] ISO/IEC 27001:2022, Information security management systems

[9] ISO/IEC 42001:2023, Artificial intelligence management system

MITRE ATT&CK: Indirect Prompt Injection (T1698)

MITRE ATT&CK: Direct Prompt Injection (T1699)

MITRE ATT&CK: Command and Scripting Interpreter (T1059)

MITRE ATT&CK: Credentials In Files (T1552.001)