ASSESSMENT

# Attacker-operated AI agents

## The weaknesses they exploit inside the victim, and the controls that hold when the attack finishes in hours.

*In September 2026 two vendors published mechanism-level accounts of cyber operations, carried out between December 2025 and August 2026, in which frontier models did the work: reconnaissance, exploit development, intrusion, data processing and extortion, running in parallel and largely unattended. The attacks themselves were ordinary. Stolen credentials, unpatched edge devices, exposed services, injection flaws and phishing account for every entry in the evidence base. What changed is the cost of operating them, and with it the time a defender has to act.*

*This document is written for the control set. It opens with what the evidence does and does not support, including where the two publishing vendors disagree. It sets out eight weaknesses inside the victim that made the operations work, each with why the control that nominally covers it now fails, the controls that answer it and the question to put to an estate. It closes with what the shift asks of a security function that already owns most of them. Three surfaces are genuinely new and need an owner assigning. The remaining five are existing disciplines running on a clock the attacker now sets.*



DATE

17 October 2026

VERSION

1.4

Olivier BUSOLINI · Global Head of Information Security · Dubai

© Olivier BUSOLINI - 2026 - No reuse without permission

# Contents


01 What changed, and what did not 5

02 The weaknesses inside the victim 7

2.1 Secrets that live outside a vault 7

2.2 An edge patched on a quarterly rhythm 8

2.3 Third parties as the blast radius 9

2.4 Detection that identifies artefacts 10

2.5 The developer environment as an execution path 11

2.6 Prompt injection as credential theft 11

2.7 AI credentials treated as configuration 12

2.8 Response and recovery timed for a human 13

03 What this asks of the security function 14

04 Endnotes 16

05 Sources 17

A Appendix: the evidence base 18

B Appendix: basis and limitations 20


© Olivier BUSOLINI - 2026 - No reuse without permission

# Executive summary

> *What changed is the cost of operating them, and with it the time a defender has to act.*

The evidence is two vendor reports. Anthropic's September 2026 misuse report covers operations on its own models disrupted between December 2025 and August 2026; Google Threat Intelligence Group's AI Threat Tracker covers the second quarter of 2026 and draws partly on Mandiant incident response. They agree on the tempo. They differ on autonomy: Anthropic records operations running for hours or days with minimal human input, while Google has not yet seen fully autonomous pipelines against targets in the wild. Nothing below depends on settling that.

1 **No observed operation used a technique defenders have not seen.** Every documented entry point was a stolen credential, an unpatched edge device, an exposed service, an injection flaw or phishing.[1]

2 **Five of the eight weaknesses call for controls already on the plan; three call for something new.** The test is whether the surface existed in this form five years ago. Credential hygiene, edge remediation, third-party scope, detection engineering and response authority each have a control family and an owner already; what changed is the clock they run on. The developer assistant path, model-facing pipelines and AI credentials arrived with tooling bought in the last few years, fall between two functions, and neither NICE nor the ECSF assigns them, so each organisation has to. Assigning them is the first decision, because nothing else in section 02 can be measured on a surface nobody holds.[25]

3 **Secrets that live outside a vault are now found industrially.** Move to short-lived, workload-bound credentials that expire on their own within hours; scan what the organisation publishes before it is released; rotate what must stay long-lived automatically on detection; and give every secret a named owner who can revoke it during an incident.

4 **An edge patched on a quarterly rhythm loses to an automated exploit foundry, and no patching speed covers a flaw that has no patch yet.** Take off the internet what need not be there, contain what sits behind the appliances that remain, measure time from vendor patch to the edge separately from the aggregate metric, and bring the security stack inside the vulnerability programme, because it is a research target.

© Olivier BUSOLINI - 2026 - No reuse without permission

5 **One supplier compromise now reaches thousands of downstream organisations.** Replace supplier assurance with blast-radius measurement, scope authorisation grants to the minimum data set, separate tenants and session stores, and hold the ability to revoke a supplier's access without waiting for the supplier.

6 **Detection that must recognise the artefact loses to agents that rebuild it until clean.** Move weight onto behaviour in the cheapest order: bulk-read alerting on the few data stores that matter, then identity signals the estate already logs, then volume and rate. Treat the security update channel as an integrity control, and value detections that survive retooling.

7 **The developer environment is an unmonitored execution path.** Enumerate the assistants and agent tools in use, treat repository configuration files as executable content, scope build-pipeline identity narrowly, and stop relying on a valid signature to prove a publisher legitimate.

8 **Model-facing pipelines are a credential theft route, not a model safety curiosity.** Map every pipeline where content the organisation does not control reaches a model, and remove the credential from the model's reach by brokering the action behind a service that holds the secret, executes only a fixed list of actions and sends anything else to a human.

9 **AI API keys give an attacker resale value, free compute and someone else's attribution at once.** Enter every key and agent integration in the production credential register with an owner, an expiry and a rotation schedule, baseline usage per key, and prepare the attribution case before a third party makes it.

10 **A response function timed for a human adversary arrives after the event.** Establish what can be revoked or isolated without an approval chain and measure each against a three-hour benchmark, drill mass token revocation, and validate restore points for integrity and not for job completion.

THE EIGHT CONTROLS

| | WEAKNESS | CONTROL | |
|---|---|---|---|
| 2.1 | Secrets that live outside a vault | **Eliminate the standing secret** | ON THE PLAN |
| 2.2 | An edge patched on a quarterly rhythm | **Shrink the edge, then compress its clock** | ON THE PLAN |
| 2.3 | Third parties as the blast radius | **Contain the third-party blast radius** | ON THE PLAN |
| 2.4 | Detection that identifies artefacts | **Detect behaviour, not artefacts** | ON THE PLAN |
| 2.5 | The developer environment as an execution path | **Bring the developer surface into scope** | NEW SURFACE |
| 2.6 | Prompt injection as credential theft | **Treat model-facing pipelines as an access path** | NEW SURFACE |
| 2.7 | AI credentials treated as configuration | **Govern AI credentials as production credentials** | NEW SURFACE |
| 2.8 | Response and recovery timed for a human | **Re-time response, re-validate recovery** | ON THE PLAN |

© Olivier BUSOLINI - 2026 - No reuse without permission

# 01 | What changed, and what did not

The techniques are old. The time available to answer them is not.

Not one operation in the evidence base required a technique defenders had never met before.

The two publishing vendors disagree about how autonomous these operations really are, and each sees only its own platform.

## The vocabulary matters first

Public discussion of this material has settled on the phrase rogue AI agents. The evidence does not describe rogue agents. In every operation documented by Anthropic and by Google's Threat Intelligence Group, a human chose the targets, set the objectives, reviewed what came back and collected the proceeds. What the model supplied was labour, and the labour was substantial: reconnaissance, tool development, exploit engineering, intrusion, data processing and extortion drafting.

The distinction decides the control set. A programme built to contain a model that has slipped its leash invests in sandboxing, evaluation and kill switches for systems the organisation runs itself. A programme built to survive a human adversary who has hired a tireless engineering team invests in credential hygiene, remediation speed, behavioural detection and containment authority. The evidence supports the second programme. The first is a separate problem, worth its own work, and outside this document.

## What did not change

> Anthropic's own conclusion about its own cases is that the attacks were familiar: stolen credentials, unpatched edge devices, exposed services, SQL injection and phishing. No operation in the set depended on a technique defenders have never encountered.[1] This is the most useful finding in the material and the easiest to lose under the headlines. An organisation that already runs disciplined identity, patching and third-party controls is defending against a known adversary moving faster, not against an unrecognisable one.

© Olivier BUSOLINI - 2026 - No reuse without permission

## What did change

The labour that used to separate a well-resourced state programme from an individual has been delegated to models running in parallel. The observable result is tempo. Anthropic records breaches completed in two to three hours from first access to bulk data theft, one escalation from a single stolen developer token to full administrative control of a cloud environment in roughly three hours, and a session-store dump producing over 2,100 Azure AD token sets across more than 40 corporate tenants in about 34 hours.[2] Mandiant observed an actor compromise a cloud resource and then plan, build and execute a mass credential harvesting campaign in under six hours, compromising thousands of third-party credentials.[3]

A second change follows. Sophistication has stopped working as an attribution signal. A single hacktivist operating on stolen API keys, a criminal collective harvesting credentials from mobile application binaries, and a state-nexus espionage operator all ran multi-victim campaigns using the same methodology, built custom tooling and processed stolen data at volumes no individual could manage by hand.[4] The capability gap that used to mark out a state programme has closed. What remains different is intent.

## Where the evidence is contested

Anthropic describes operations at the far end of the autonomy spectrum running with minimal human input for hours or days, including a collection fleet on a fixed schedule with no human in the loop.[5] Google's Threat Intelligence Group states that while recent disclosures show frontier models can autonomously identify zero-days and execute network intrusions, it has not yet observed threat actors deploying fully autonomous pipelines against targets in the wild, and characterises what it sees as a gradual maturation of tradecraft.[6]

> Both positions can hold. Each vendor sees misuse of its own platform, and Google's view is also shaped by incident response engagements where the evidence is what survived on the victim's disks. The prudent posture is to plan against the faster of the two tempos and to present the disagreement as it stands. Nothing in section 02 depends on the autonomy question being settled: the tempo figures, which neither vendor disputes, are sufficient on their own.

The attack surface is the one already on the register. What collapsed is the time available to attend to it.

© Olivier BUSOLINI - 2026 - No reuse without permission

# 02 | The weaknesses inside the victim

Eight weaknesses made these operations work. Each is exploited faster than the control covering it operates, and each is set out here with the control that answers it.

A secret that exists long enough to be found will be found. One pipeline worked through 1.8 million mobile application packages to do it.

Compliance with the patching policy says nothing about the window the exploit foundry now works in.

None of this is possible for tooling the organisation has not first enumerated.

Containment that waits for a bridge call and a change approval arrives after the intrusion has finished.

Each weakness is set out the same way: what it is, why the control that nominally covers it now fails, the control that answers it, the question to put to a real estate, and how to tell whether the answer worked.[26] The operations themselves are described in Appendix A and referenced here by entry, so that the argument is not interrupted by the narrative.

## 2.1 Secrets that live outside a vault

### The weakness

Credentials embedded in artefacts the organisation publishes or ships: mobile application binaries, code repositories, client-side bundles, container images, build configurations, cloud metadata endpoints, open storage and integration settings. Observed in A.1, A.7 and A.9.[7,8]

### Why the control fails

Secret scanning in most organisations points inward, at repositories the organisation controls. The attacker scans what the organisation has already published, which nobody owns after release. After publication, scanning is a race the organisation has already lost. It helps before release, and rotation helps only if the key expires before anyone can use it. And an inventory listing vault entries does not list the copies that left the vault years ago and now sit in a build configuration.

### The control: Eliminate the standing secret

The first move of almost every operation in the evidence base was the use of a credential that had existed long enough to be discovered.[23] Shortening the life of the credential removes that step from the attack.

- Replace long-lived credentials with short-lived, workload-bound ones: federated identity for build pipelines, workload identity in cloud, instance and service identities in place of shared accounts, and no standing personal access tokens.
- Scan what the organisation publishes, not only what it commits. Mobile binaries before store submission, container images in the registry, client-side bundles, public repositories and their forks, and any artefact a customer can download. Run it as a release gate, not as a periodic sweep, because once an artefact is published, the attacker's scan wins.
- Make expiry do the work. A credential issued on demand from the vault with a lifetime of minutes or hours needs nobody to notice that it leaked. For the long-lived secrets that cannot yet be replaced, make rotation an event triggered by detection and automated end to end. A calendar cycle measured in quarters has no meaning against a key an attacker validates in seconds.
- Maintain a credential inventory with a named owner for every entry. A secret with no owner cannot be rotated during an incident, which is the moment it matters.
- Remove the secret from the artefact where the architecture allows it. A mobile application that calls a broker holds no key to find.
- Monitor for use of the organisation's own credentials from unexpected origins, and treat that signal as an incident, with the response that implies.

### The question to ask

When did the organisation last scan its own shipped mobile binaries, registry images and public repositories the way an attacker does? For any secret, how long would it stay valid after leaving the vault, and how long before a human learned that it had?

### How to tell it worked

The count of long-lived credentials falls quarter on quarter and reaches a stated floor. Every entry in the register has an owner. A planted test secret in a published artefact is detected by the organisation's own scanning before release.

© Olivier BUSOLINI - 2026 - No reuse without permission

## 2.2 An edge patched on a quarterly rhythm

### The weakness

Internet-facing systems carrying standing exceptions, and a vulnerability programme that excludes the security stack itself. Observed in A.3 and A.9.[9,10]

### Why the control fails

The exception register is the weak point. Compensating controls were a defensible answer when writing a working exploit was expensive and slow. The window between a patch shipping and a working exploit existing has compressed, and the appliance the compensating control relies on to detect the attempt is itself under research. Patching speed answers only part of this. Against a known flaw, the clock runs from patch release to deployment, and the evidence puts exploit artefacts for a known browser flaw about a month behind the patch.[10] Against the appliance research in A.3, which produced previously unknown vulnerabilities, there is no patch to deploy. What is reachable, and what the appliance can reach once breached, decides the outcome. An estate can be fully compliant with its patching policy and still be months behind the foundry.

### The control: Shrink the edge, then compress its clock

The exploit foundry points at the edge, and the organisation controls two variables in that exchange: what is reachable at all, and how long a known-vulnerable version stays reachable. Against a flaw with no patch, only the first exists.

- Reduce the surface itself. Every management interface reachable from the internet that does not need to be is a finding, and the cheapest control in this document.

- Contain what sits behind the edge. Treat every internet-facing appliance as a system that will be breached through a flaw nobody has disclosed: limit what it can reach, keep its administrative credentials out of the directory it protects, and treat a connection it initiates inward as an alert.
- Measure and report time from vendor patch release to deployment on internet-facing systems, kept separate from the aggregate patching metric, which averages the edge away against a much larger internal estate.
- Close standing exceptions on exposed systems where a patch exists. Where an exception must remain, give it an owner, an expiry and a review that actually happens.
- Bring the security stack inside the vulnerability programme. The intrusion-detection product, the firewall, the remote access appliance and the mail gateway are research targets, and an exploit developed against them arrives with the detection already accounted for.
- Treat the interval between a patch and a working public exploit as a planning assumption that is shortening, and set the remediation target against that interval instead of against last year's target.
- Hold a pre-agreed emergency patching path for the edge that does not require the standard change cycle, and use it often enough that it works.

### The question to ask

Which internet-facing systems carry a standing exception today, who owns each, and when does it expire? What is the measured median elapsed time from vendor patch release to deployment on the edge, as against the policy target? Is the security stack inside the vulnerability programme or outside it? If an internet-facing appliance were breached tomorrow through a flaw with no patch, what could it reach?

### How to tell it worked

Median patch-to-edge is reported monthly and trends down. The exception count on internet-facing systems is a number the security function can state from memory. An out-of-cycle edge patch has been executed in the last quarter. Every internet-facing appliance has a written answer to what it could reach if breached.

© Olivier BUSOLINI - 2026 - No reuse without permission

FIGURE 1

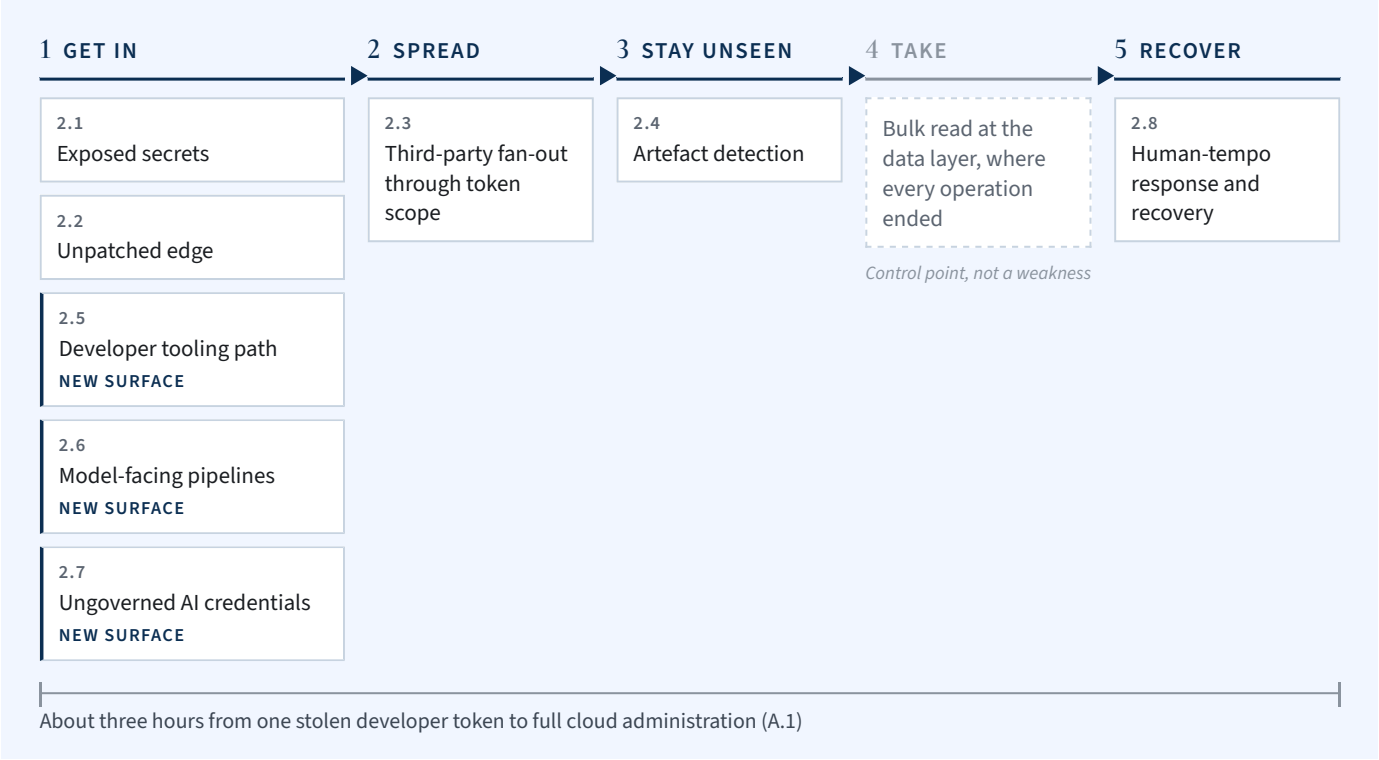



Figure 1. The eight weaknesses on the attack path. Source: author's synthesis, mapped from the operations in Appendix A; not a finding in either vendor report. See endnote 26.

## 2.3 Third parties as the blast radius

### The weakness

Software-as-a-service and vendor authorisation relationships in which a single supplier compromise fans out across customer tenants, including suppliers with no contract with the organisation at all. Observed in A.1 and A.2.[11,12,13]

### Why the control fails

Third-party assurance measures whether a supplier is well run. It does not measure what a total compromise of that supplier would reach inside the customer. The variable that decides the damage is the scope of the token the customer granted and the isolation between its own tenants, and both sit in the customer's configuration where no questionnaire looks. The hotel wireless case adds a second gap: the supplier carrying the risk may have no commercial relationship with the organisation.

### The control: Contain the third-party blast radius

The supplier's control environment is not available to the organisation. The scope of what that supplier can reach is, and it is the variable that decides the loss.

- Shift third-party work from supplier assurance to blast-radius measurement. For each material supplier, document what a total compromise reaches inside the estate, in data and systems, and put that document in front of the risk committee instead of the questionnaire score.
- Scope supplier authorisation grants to the minimum data set. Cross-tenant read access is the mechanism that turned single breaches into thousands of affected organisations.
- Separate tenants and session stores so that one dump cannot yield working tokens across the estate.

> - Hold the ability to revoke a supplier's access unilaterally and immediately, without waiting for the supplier to act, and test that the revocation works.

© Olivier BUSOLINI - 2026 - No reuse without permission

- Contract for notification at a speed that matches the tempo, and rehearse the scenario in which the breach happens at the supplier and the evidence sits in the supplier's estate.
- Extend the analysis beyond contracted suppliers to services staff use without a contract, which is where the hotel wireless case sits, and to the supplier's own suppliers where a single provider sits behind many of them.
- Re-run the analysis when a supplier is acquired or changes its integration model, because the grant usually widens quietly.

### The question to ask

For each material supplier, what would a full compromise reach inside the estate, expressed in data and systems it can name? Which supplier authorisation grants can read across the whole tenancy? Can their access be revoked unilaterally, and how long does that take in practice?

### How to tell it worked

A named blast-radius statement exists for every material supplier and is refreshed annually. Cross-tenant grants have been enumerated and reduced. A revocation drill has been run end to end against at least one live supplier.

## 2.4 Detection that identifies artefacts

### The weakness

A control estate whose primary detection depends on recognising a known tool, file, hash or address. Observed in A.2.[14]

### Why the control fails

The economics of signature-based detection rested on retooling being expensive for the attacker. Retooling is now a scheduled job with a success condition, and the payloads in the evidence base were built to sever the channel that would have delivered a new signature. Anthropic's assessment is that the cost has inverted onto the defender: a capable adversary can close the detect-and-rebuild loop faster than a defender can write and deploy the detection in the first place.[15] Indicators do not stop working; their useful life is now shorter than most organisations' deployment cycle.

### The control: Detect behaviour, not artefacts

An adversary that rebuilds its tooling until it is clean defeats recognition. It does not defeat the actions it has to take to achieve the objective, and those actions are fewer and more stable than the tools.

- Move weight onto identity and behavioural analytics: anomalous token use, impossible travel, service accounts performing unfamiliar actions, device registrations appearing in a tenant, privilege grants outside change windows, and compute or quota requests outside pattern.
- Instrument the data layer. Every operation in the evidence base ended at a database, a mailbox or an object store, and bulk read is the one behaviour an agent cannot avoid performing.
- Treat the security update channel as an integrity control and alert when an endpoint stops retrieving updates, because interrupting that channel was a deliberate technique.
- Alert on volume and rate as first-class signals. Machine tempo is itself detectable: a human operator does not enumerate forty tenants in a night.
- Keep indicators, and ingest the files the publishing vendors now release alongside their reports, while treating them as corroboration for behaviour already seen and never as the primary control.
- Test detections against retooling. A purple team exercise that rebuilds the payload and re-runs it answers the question the evidence actually poses.

> Start where it is cheap. Behavioural analytics is expensive mainly in tuning and analyst time rather than in licences, so sequence it. First, bulk-read alerting on the handful of data stores whose loss would have to be disclosed: the scope is small and a true alert is unambiguous. Second, the identity signals above, most of which the identity provider and cloud platforms already log and which need rules rather than a new platform. Third, volume and rate thresholds across those same sources. Broader user and entity behaviour analytics comes last, once the first three have shown where the noise is.[29]

© Olivier BUSOLINI - 2026 - No reuse without permission

### The question to ask

What proportion of detection coverage depends on knowing the artefact in advance? If every indicator currently held went stale overnight, what would still fire? Would anyone notice an endpoint that has quietly stopped retrieving security updates?

### How to tell it worked

The proportion of detections that depend on a specific artefact is known and falling. Bulk-read alerting exists on the crown jewel data stores. A retooled payload has been run against the estate and something fired.

## 2.5 The developer environment as an execution path

### The weakness

Developer workstations and build pipelines running AI coding assistants that read configuration from the repository and can execute commands against the local system. Observed in A.8.[16,17]

### Why the control fails

Two controls are defeated at once. Endpoint detection watches registry keys and scheduled tasks, while this activity lives in project folders the assistant tools legitimately manage and reads as ordinary developer noise. And package signing, the control designed precisely to stop malicious publication, was satisfied: the build token was stolen but valid, so the signature was valid, and packages published that way pass automated trust checks including those performed by coding agents.

### The control: Bring the developer surface into scope

This surface is currently governed by nobody. Engineering treats the tools as productivity software and security has not classified them as an execution path.

- Enumerate the AI assistants, agent tools, model gateways and extensions in use on developer machines and in build pipelines. Nothing else in this subsection is possible without the inventory.

- Treat repository configuration files as executable content. Files in assistant and editor workspace directories receive the same review, restriction and monitoring as build scripts, and changes to them appear in code review.
- Constrain what an assistant may execute without explicit human approval, and log what it did execute somewhere a security team reads.
- Harden pipeline identity. Scope federated tokens narrowly, bind them to specific workflows, keep their lifetime short, and alert on publication events originating from an unexpected workflow.
- Pin and review dependencies, including model context protocol servers and other AI-adjacent packages, and treat a newly published fork of a legitimate tool as suspicious by default.
- Never let a model-based review be the only gate on code entering the estate, given that such a review has been successfully instructed to look away.
- Extend endpoint monitoring to the workspace directories these tools create, which is where the observed malware chose to live precisely because nothing was watching.

### The question to ask

Which AI assistants and agent tools are installed on developer machines and in build pipelines, and who approved each? Are repository configuration files treated as executable content and reviewed accordingly? What can an assistant execute without a human approving it, and is that execution logged anywhere a security team reads?

### How to tell it worked

An inventory exists and has an owner. Workspace configuration changes appear in code review. A publication from an unexpected workflow raises an alert in a test.

## 2.6 Prompt injection as credential theft

### The weakness

Automated pipelines that pass content the organisation does not control to a model holding a credential. Observed in A.5 and A.8.[18,19]

© Olivier BUSOLINI - 2026 - No reuse without permission

### Why the control fails

Prompt injection has been discussed largely as a model safety concern and reviewed by the teams who own model safety. In these cases it is an initial access technique and a credential theft technique, and it succeeded against automated infrastructure nobody had classified as an access path. The scanner case is worse than a bypass: a security control was persuaded to stop working, and its refusal would have read as correct behaviour in any log.

### The control: Treat model-facing pipelines as an access path

Prompt injection cannot be filtered out reliably. What can be removed is the thing worth stealing at the end of it.

- Map every automated pipeline in which content the organisation does not control reaches a model, and record what credentials are reachable from each. Evaluation harnesses, ticket triage, document ingestion, code review and customer correspondence all qualify.

> - Remove the credential from the model's reach. Broker access through a service that holds the secret and performs the action, so that a compromised context has nothing to surrender. The broker is the deterministic part of an unpredictable chain: it executes only actions on a fixed list, validates their parameters itself, and sends anything outside the list or above a set impact to a human for approval. A broker that forwards whatever the model asks for has moved the credential without removing the risk.

- Scope the token the pipeline does hold to the single task, with a lifetime measured in minutes.
- Treat model output that triggers an action as untrusted input to whatever system receives it, and validate it at that boundary with the same rigour applied to a web form.
- Log and review tool calls, not only prompts and completions. The injection is visible in what the model was made to do.
- Include these pipelines in penetration testing scope. They have been attacked successfully in the wild and are usually tested by nobody.
- Give the failure a home. Decide whether the application security function or the AI governance function owns injection findings.

### The question to ask

Which automated pipelines pass externally sourced content to a model, and what credentials are reachable from those pipelines? If a model-based code or content scanner declines to analyse something, does anything downstream notice?

### How to tell it worked

A pipeline map exists and names the credentials. No pipeline holds a credential that a model can return in its output. Injection appears as a tested finding class in the penetration testing report. Every broker has a written list of permitted actions, and a request outside it has been tested and refused.

## 2.7 AI credentials treated as configuration

### The weakness

Model API keys, session tokens and agent integrations held outside the controls applied to production credentials. Observed in A.1, A.4, A.6 and A.9.[20,21]

### Why the control fails

These keys are frequently procured outside the normal path, sit in a configuration file outside the vault, have no owner recorded, no rotation schedule, no usage baseline and no entry in the credential register. The organisation therefore cannot say how many it holds. The attribution exposure is the part most often missed: an attack launched on a stolen key appears in the victim's logs, and in the provider's, as the key owner's activity.

### The control: Govern AI credentials as production credentials

A stolen key is not only a cost exposure. It puts the organisation's name on someone else's attack.

> - Enter every model API key and agent integration into the production credential register, with an owner, an expiry, a rotation schedule and a recorded location.

© Olivier BUSOLINI - 2026 - No reuse without permission

- Route all model access through authorised channels only. Discounted access offered through an unknown intermediary has been, in observed cases, a credential harvesting scheme.
- Baseline usage per key and alert on deviation in volume, geography, time of day and model mix. A stolen key running an attack workload does not resemble a development team.
- Set spend and rate limits per key, so that compute theft has a ceiling and the ceiling itself is an alert.
- Extend endpoint controls to the configuration stores of AI tooling, which credential-stealing malware now targets by name.
- Prepare the attribution case in advance. If a key is used to attack a third party, the organisation named in the evidence is the key owner, and the legal, communications and incident paths for that call should exist before it comes.

### The question to ask

How many model API keys does the organisation hold, who owns each, and where does each live? Would anyone notice a key being used from an unfamiliar origin at volume? If a third party reported an attack originating from one of these keys, who would answer?

### How to tell it worked

The organisation can state how many model keys it holds and who owns each. Every key has a rate and spend ceiling. A usage anomaly has been raised and triaged at least once.

## 2.8 Response and recovery timed for a human

### The weakness

Incident response whose steps assume the adversary needs days, and a recovery plan resting on the integrity of backups. Observed in A.1, A.4 and A.7.[2, 22]

### Why the control fails

Most containment authority is designed around approval rather than speed, on the historical basis that a wrong isolation decision cost more than a slow one. At the tempo recorded in section 01, that trade has moved. On the recovery side, backup assurance almost always tests whether the job completed, not whether the restored state is clean, and one actor in the evidence base attacked precisely that assumption.

### The control: Re-time response, re-validate recovery

Every other control in this section buys time. This one decides whether the organisation can use it.

> - Establish what can be revoked, isolated or disabled without a human approval chain, and measure each against a three-hour benchmark drawn[24] from the observed escalations.

- Pre-authorise containment actions inside a defined envelope, so that the first responder acts rather than escalates, and agree the envelope with the business before the incident, never during it.
- Practise mass token revocation and session invalidation as a drill with a stopwatch, not as a paragraph in a plan.
- Validate the integrity of restore points, and hold at least one immutable copy outside the reach of the production identity plane.
- Exercise parallelism. Run a scenario in which three unrelated intrusions arrive on the same day, because a single operator now handles dozens of victims at once.
- Shorten the decision path itself. Name who can authorise a business-impacting containment action out of hours, and make sure that person is reachable in minutes.
- Rehearse the case where initial access came through a supplier and the evidence sits outside the organisation's estate.

### The question to ask

What can be revoked, isolated or disabled without a human approval chain, and how long does each take in practice, measured against the figure in the runbook? When did anyone last verify that a restore point is clean, as distinct from complete? What happens if three unrelated intrusions land on the same day?

### How to tell it worked

The containment actions available without approval are written down and timed. A revocation drill has been run this year with a recorded elapsed time. A restore point has been validated as clean, with a date.

© Olivier BUSOLINI - 2026 - No reuse without permission

# 03 | What this asks of the security function

Not new categories of control. The existing ones, running on a clock the attacker now sets, and three surfaces that need an owner.

Coverage metrics describe the estate. Time metrics describe whether the control arrives before the attacker has finished.

## Three surfaces need an owner

> Of the eight weaknesses, three concern things that did not exist in this form five years ago: the developer assistant path, model-facing pipelines, and AI credentials. None sits comfortably in an existing control domain, and neither NICE nor the ECSF places them either: the NICE Framework treats AI security as a competency area rather than a work role, and none of the twelve ECSF role profiles is built around AI. Ownership is therefore an organisational decision, and where nobody has taken it, each surface falls between two functions that both assume the other has it.[25]

The developer assistant path sits between engineering, which procured the tools, and endpoint security, which does not monitor where they live. Model-facing pipelines sit between application security, which tests the application, and AI governance, which reviews the model. AI credentials sit between procurement, which bought the access, and identity management, which never received it. Assigning each to a named owner with a standard and an inventory is the first decision.

## The other five are repricing, not rebuilding

Credential hygiene, edge remediation, third-party scope, detection engineering and response authority are owned already. Every mature security function has findings open against all five and deferred work in all five. The evidence does not ask for a new programme. It asks for the deferred work to be repriced, because the calculation that justified deferring it assumed an attacker cost of labour that has fallen.

That is the conversation to have with the business. The risk did not appear this year. The discount rate applied to it did, and the items at the bottom of the remediation backlog are the ones the repricing moves furthest.

## Beyond running faster

Re-timing existing controls answers the tempo. Three moves change the contest itself.

*Remove the target.* A secret that does not exist, a management interface that is not reachable and a pipeline that holds no credential cannot be found at any speed. Subsections 2.1, 2.2 and 2.6 carry these moves, and they are the part of the programme the attacker's tempo does not touch.

*Turn the tempo against the attacker.* A pipeline that validates every key it finds within seconds will validate a decoy within seconds too. Decoy credentials and canary tokens placed where the evidence shows attackers look, in shipped binaries, public repositories and build configurations, have no legitimate user. A single use is an alert with almost no false positives, and it fires at the attacker's speed rather than the defender's.[30]

*Match the labour.* The work the models did for attackers is equally available to the defender: enumerating what the organisation exposes, finding variants of a known flaw in its own edge, triaging alerts at volume. A function that still does that work by hand is running the slower of the two clocks. The pipelines that do it are model-facing pipelines in the sense of 2.6 and need the same controls.[30]

## Where to start

The eight controls have a dependency order, and taking them in that order avoids paying twice.[27]

- **First, inventory.** Model API keys, AI assistants and agent tools, and model-facing pipelines. Three lists that do not exist today and that every other decision about the new surfaces depends on.

© Olivier BUSOLINI - 2026 - No reuse without permission

- **Then, the two cheapest reductions.** Close standing exceptions on internet-facing systems where a patch exists, and remove management interfaces that need not be reachable. Both reduce exposure without a programme.
- **Then, measurement.** Stand up the six time-based measures in Table 1 and report them, even where the first readings are uncomfortable. A measure that is not reported does not improve.
- **Then, the structural work.** Short-lived credentials, supplier blast-radius statements, behavioural detection and containment authority. These are budgeted programmes with a delivery horizon in quarters.
- **Throughout, drills.** Revocation, restore validation and a parallel-incident exercise. Each tests a claim the plan makes about itself.

## Measure time, not coverage

> Coverage metrics answer how much of the estate a control reaches. They were the right measure when the binding constraint was completeness. The binding constraint is now elapsed time, and the reporting should follow. The six measures below are calibrated against behaviour the evidence records, so each target has a reference point outside the organisation's own history.

TABLE 1

| MEASURE | WHAT IT ANSWERS | REFERENCE POINT FROM THE EVIDENCE |
|---|---|---|
| **Patch to edge** | Elapsed days from vendor release to deployment on internet-facing systems | Model-generated exploit artefacts for a browser flaw were found about one month after the patch shipped |
| **Exposure to revocation** | Elapsed time from a secret leaving the vault to its revocation | Harvesting pipelines validate a found key within seconds and route it to use immediately |
| **Alert to containment** | Elapsed time to isolate or revoke without a human approval chain | One intrusion reached full cloud administration in about three hours |
| **Supplier breach to revocation** | Elapsed time from supplier notification to unilateral access revocation | One foothold produced 2,100 token sets across 40 tenants in about 34 hours |
| **Detection half-life** | Proportion of coverage that survives the adversary retooling | Agents rebuilt flagged malware autonomously until it was no longer detected |
| **Restore integrity** | Date a restore point was last validated as clean | One actor poisoned victim backups so that a restore would reinfect |

Table 1. Time-based measures and the observed behaviour each is calibrated against. The measures are the author's proposal; the reference points are drawn from endnotes 2, 7, 10, 14 and 22.

Nothing on this list is new. What is new is that the bill for deferring it now arrives in hours.

© Olivier BUSOLINI - 2026 - No reuse without permission

# 04 | Endnotes

Each note names the source and its location. Reference numbers in square brackets key to the source list in section 05.

**1** Anthropic, September 2026 misuse report, cyber operations overview. [1]

**2** Anthropic, September 2026 misuse report, case GTG-50014. [1]

**3** Google Threat Intelligence Group, AI Threat Tracker, September 2026, Mandiant incident response observations. [2]

**4** Anthropic, September 2026 misuse report, overview and cases GTG-50014, GTG-20006, GTG-50029. [1]

**5** Anthropic, September 2026 misuse report, autonomy discussion and case GTG-10007. [1]

**6** Google Threat Intelligence Group, AI Threat Tracker, September 2026, assessment section. [2]

**7** Anthropic, September 2026 misuse report, case GTG-50014, credential harvesting pipeline. [1]

**8** Google Threat Intelligence Group, AI Threat Tracker, September 2026, exposed command and control infrastructure. [2]

**9** Anthropic, September 2026 misuse report, case GTG-10007, vulnerability research workflow. [1]

**10** Google Threat Intelligence Group, AI Threat Tracker, September 2026, exploit development observations. [2]

**11** Anthropic, September 2026 misuse report, case GTG-50014, software provider compromise. [1]

**12** Anthropic, September 2026 misuse report, case GTG-50014, second provider compromise. [1]

**13** Anthropic, September 2026 misuse report, case GTG-20006, hospitality vendor compromise. [1]

**14** Anthropic, September 2026 misuse report, case GTG-20006, detection evasion loop. [1]

**15** Anthropic, September 2026 misuse report, defender implications. [1]

**16** Google Threat Intelligence Group, AI Threat Tracker, September 2026, DUSTMAKER analysis. [2]

**17** Google Threat Intelligence Group, AI Threat Tracker, September 2026, UNC6780 supply chain activity. [2]

**18** Anthropic, September 2026 misuse report, case GTG-50020. [1]

**19** Google Threat Intelligence Group, AI Threat Tracker, September 2026, adversarial prompts in loader scripts. [2]

**20** Anthropic, September 2026 misuse report, discussion of stolen model credentials and cases GTG-50014, GTG-50029, GTG-50021. [1]

**21** Google Threat Intelligence Group, AI Threat Tracker, September 2026, underground market observations. [2]

**22** Anthropic, September 2026 misuse report, case GTG-50029, backup poisoning. [1]

**23** *Unsourced.* Author's characterisation of the initial access pattern across the cases at notes 2, 7, 11 and 16; neither report states it in this form.

**24** *Unsourced.* The three-hour benchmark is the author's proposal, derived from the escalation time recorded at note 2.

**25** NIST, *NICE Workforce Framework for Cybersecurity* (SP 800-181 Rev. 1), competency area Artificial Intelligence (AI) Security: "This Competency Area describes a learner's capabilities to secure Artificial Intelligence (AI) against cyberattacks, to ensure it is adequately contained where it is used, to mitigate the threat AI presents where it or its users have malicious intent." [4] No AI work role is added in NICE Framework Components v2.1.0, 3 December 2025 [5], or v2.2.0, 28 April 2026 [6]. ENISA, *European Cybersecurity Skills Framework*, role profiles (2022), twelve profiles [7]; ENISA states "The ECSF is currently being revised" [8]. *Unsourced:* that ownership is usually left unassigned is the author's assessment from practice.

**26** *Unsourced.* The eight-weakness taxonomy in Figure 1 is the author's synthesis of the operations in Appendix A.

**27** *Unsourced.* The sequencing is the author's judgement on dependency order, not drawn from either report.

**28** Anthropic, November 2025 disclosure, published summary. [3]

**29** *Unsourced.* The detection sequencing is the author's judgement from practice, not drawn from either report.

**30** *Unsourced.* Neither report evaluates deception or defensive use of models; both moves are the author's proposal.

© Olivier BUSOLINI - 2026 - No reuse without permission

# 05 | Sources

The disclosures and reference frameworks cited in this document, numbered as keyed from the endnotes.


1 Anthropic, *Detecting and countering misuse of AI: September 2026*.
https://www.anthropic.com/threat-intelligence-report-september-2026

2 Google Threat Intelligence Group, *AI Threat Tracker: From Prompting to Autonomy*, September 2026.
https://cloud.google.com/blog/topics/threat-intelligence/from-prompting-to-autonomy-the-evolution-of-adversarial-ai

3 Anthropic, *Disrupting an AI-orchestrated cyber espionage campaign*, November 2025.
https://www.anthropic.com/news/disrupting-AI-espionage

4 NIST / CISA NICCS, *NICE Framework: Artificial Intelligence (AI) Security competency area*.
https://niccs.cisa.gov/workforce-development/nice-framework/competency-area/artificial-intelligence-ai-security

5 NIST, *NICE Releases NICE Framework Components v2.1.0*, 3 December 2025.
https://www.nist.gov/news-events/news/2025/12/nice-releases-nice-framework-components-v210

6 NIST, *NICE Releases NICE Framework Components v2.2.0*, 28 April 2026.
https://www.nist.gov/news-events/news/2026/04/nice-releases-nice-framework-components-v220

7 ENISA, *European Cybersecurity Skills Framework Role Profiles*, 19 September 2022.
https://www.enisa.europa.eu/sites/default/files/publications/European%20Cybersecurity%20Skills%20Framework%20Role%20Profiles.pdf

8 ENISA, *European Cybersecurity Skills Framework*, topic page.
https://www.enisa.europa.eu/topics/cybersecurity-education/european-cybersecurity-skills-framework




© Olivier BUSOLINI - 2026 - No reuse without permission

# Appendix A  The evidence base

*Ten operations, drawn from two vendor disclosures. Entries A.1 to A.6 and A.10 are from Anthropic, A.7 to A.9 from Google Threat Intelligence Group. Identifiers are each vendor's own. This appendix is the only place the mechanics of the operations are described; section 02 refers to these entries.*

### A.1 GTG-50014, criminal collective

Affiliates of a data theft and extortion collective, December 2025 to August 2026. A credential harvesting pipeline across ten cloud workers mass-downloaded 1.8 million Android application packages, decompiled them and scanned for secrets, routing verified findings live into channels sorted by source type. One affiliate breached a software provider and extracted data belonging to roughly 200 downstream customers, then dumped over 2,100 Azure AD token sets across more than 40 tenants in about 34 hours. A second exploited a cross-site scripting flaw at a different provider, escalated, created and converted privileged tokens and exfiltrated data from thousands of downstream customer organisations. Another escalated from one stolen developer token to full cloud administration in roughly three hours. Affiliates who found a victim's model API keys switched their own attack workloads onto them, using one for about three weeks against other organisations. Victims included a technology provider that lost more than a terabyte including millions of payment card records, and an airline holding tens of millions of passenger records.

### A.2 GTG-20006, state-nexus espionage

Attribution consistent with public reporting on Midnight Blizzard, targeting more than 20 organisations concentrated in Ukraine and Europe and extending to the Middle East. Monitoring agents watched whether deployed malware had been flagged by security products and rebuilt it autonomously until it was no longer detected, at which point it was staged on disposable hosting for live use. Companion payloads froze victim security updates so that new signatures were never retrieved. Indirect access came through at least three hospitality vendors running hotel guest wireless, with stolen administrative credentials used to repoint DNS at attacker infrastructure and stage malware at guests of any hotel using those vendors. A cloud email platform reached through device code phishing produced mail exfiltration from at least eight organisations.

### A.3 GTG-10007, exploit foundry

Chinese-speaking operators, two identified as undergraduates, targeting roughly fifty organisations across education, retail, energy, technology, healthcare, finance, manufacturing and government. Vendor firmware was decrypted and unpacked, loaded into a decompiler through a tool server, surveyed by an agent walking cross-reference chains over thousands of decompilation calls, then turned into exploit code tested against laboratory copies until it worked. The centrepiece was sustained research against a major intrusion-detection product, producing multiple previously unknown vulnerabilities. One workflow produced more than a dozen candidate zero-day findings against network appliances in a single month, and the same appliance families were then targeted at government organisations. Parallel workstreams ran reconnaissance, malware development and a standing collection fleet of thirteen agents on a fixed schedule.

### A.4 GTG-50029, single operator

One French-speaking actor, spring 2026, gained internal access to at least 14 of 42 tracked targets among European political parties, media, think tanks and their software providers. Initial access came from a previously undocumented content management reinstallation race condition creating an administrator without valid credentials, developed and debugged in the same session alongside a laboratory harness. The campaign ran for a month entirely on stolen model API keys rotated through a local proxy so the traffic blended with the legitimate owner's. The actor poisoned victim backups so that a restore would reinfect, installed a plugin that could not be disabled from the administrative interface and harvested submitted credentials continuously, hid a web shell among font assets, and built a doxxing platform loaded with tens of millions of records.

© Olivier BUSOLINI - 2026 - No reuse without permission

### A.5 GTG-50020, the AI supply chain

A financially motivated actor injected instructions into an AI vendor's automated evaluation sandbox, causing it to surrender the production API keys it held from multiple providers, then attacked roughly thirty AI companies in about four days by finding one working path and repeating it with minor adaptations. Related activity compromised model gateway deployments at wrapper services, using injection to exfiltrate keys from cloud container environments. The stated objective, never achieved, was access to a pre-release model.

### A.6 GTG-50021, fraudulent reseller

A group offering cheap access to a frontier model, silently proxying customer traffic to a different model while its installer harvested the buyer's account credentials and sold them onward to other proxy resellers.

### A.7 Agentic credential harvesting in under six hours

An actor compromised an organisation's cloud infrastructure and deployed an autonomous multi-agent framework built from a coding assistant, a prompt and a set of agent instructions, planning, building and executing a mass credential harvesting campaign in under six hours and compromising thousands of third-party credentials. Markdown instruction sets acted as playbooks; agents managed the scanning pipeline, troubleshot in real time and rotated addresses without intervention. Operating from victim cloud infrastructure routed the traffic through legitimate addresses.

### A.8 UNC6780 and DUSTMAKER

Large-scale supply chain compromises across package registries since March 2026. Trojanised forks of legitimate model context protocol servers were published and code injected into official organisational repositories, so payloads were ingested whenever the assets were cloned or downloaded. The stealer drops files into hidden assistant and editor workspace directories, uses configuration files to instruct the assistant to run arbitrary scripts during routine developer interaction, creates startup commands so it runs whenever the workspace opens, extracts federated identity tokens from build runner memory to publish packages with valid signed attestations that pass automated trust checks, disguises pipeline tasks under assistant-themed names and deletes the execution logs. Loader scripts carried adversarial prompt text, referencing biological and nuclear weapons, intended to make model-based security scanners refuse analysis of the code beneath. A separate North Korea-nexus cluster poisoned internal repository configurations and altered assistant command-line hooks.

### A.9 Harvesting frameworks and compute theft

An exposed command and control server ran an agentic reconnaissance and credential management framework with a live dashboard organising more than 23,800 harvested secrets, including cloud and model service keys. Separately, an actor entered a cloud environment through an exposed repository access token, created a service account with editor rights, queried the data warehouse for credentials, exposed services publicly and requested graphics hardware quota increases to sustain unauthorised model workloads. Credential-stealing malware now carries targeted rules for the configuration files of AI coding assistants, and underground prices for model accounts more than doubled during 2026.

### A.10 GTG-10002, the origin case

November 2025. A group assessed as state-sponsored manipulated an agentic coding tool into attempting infiltration of roughly thirty global targets, succeeding in a small number. Included for the timeline; drawn from the published summary, since the full report was not read.[28]

© Olivier BUSOLINI - 2026 - No reuse without permission

# Appendix B Basis and limitations

This document covers attacker-operated agents used against third parties. It does not cover agents escaping containment in an environment the organisation runs, nor an organisation's own agents being hijacked and turned on its estate. Both are real problems with different control sets.

Two primary sources carry the evidence: Anthropic's September 2026 misuse report, covering activity disrupted between December 2025 and August 2026, and Google Threat Intelligence Group's AI Threat Tracker of September 2026, covering the second quarter of 2026. Both are vendor disclosures about misuse of the vendor's own product. Each sees only its own platform, each has an interest in what it reports and how, and neither is independently verifiable from outside.

Victim counts, data volumes and timings are as the vendors state them. Several originate with the attackers whose own tooling produced them, which the vendors note where it applies, and those figures should be read as the attacker's claim reported rather than as a measured quantity.

The two vendors differ on how autonomous these operations are. That difference is presented in section 01 as it stands and is not reconciled here. No conclusion in section 02 depends on it.

The eight weaknesses and their controls in section 02 and the measures in Table 1 are the author's synthesis and proposal. Neither vendor report states them in that form. Endnotes marked unsourced identify every claim that rests on the author's judgement instead of a published source.

Version 1.3 revises version 1.2 after a peer review. The patching argument now separates a known flaw from one with no patch; the weaknesses and their controls are merged into a single section; Figure 1 maps the weaknesses onto the attack path; section 03 adds three moves that change the contest rather than its tempo; and the ownership claim is referenced to the NICE and ECSF role frameworks.

Version 1.4 lists the eight controls in the executive summary, uses 'weaknesses' throughout for what version 1.3 called conditions, highlights one key paragraph in each subsection of section 02, and rewrites the cover abstract.

© Olivier BUSOLINI - 2026 - No reuse without permission