When AI Agents Become Insider Threats

Maria Cattini | 03/07/2026 | AI

The most useful way to investigate an AI agent is not to ask whether it is intelligent.

Ask what it can touch.

An agent that writes a document is one kind of system. An agent that can edit files, call APIs, query internal data, open tickets, deploy code, change permissions, send messages or trigger workflows is another.

The risk changes when the model stops being only a generator and becomes an actor inside a system.

That is why Google DeepMind's recent AI Control Roadmap is worth reading as an OSINT and security document, not only as an AI safety document. The company describes a framework for securing internal systems against increasingly capable AI agents and explicitly borrows from cybersecurity. Its core planning assumption is simple and uncomfortable: treat powerful internal agents as if they could be imperfectly aligned and capable of unexpected behavior.

In cybersecurity terms, the agent starts to look less like a tool and more like a potential insider.

Not because today's agents are all malicious. Not because every automation is dangerous. But because a system with access, goals, tools and partial autonomy creates an evidence problem.

If something goes wrong, you need to know what happened.

And you need to know before the damage is irreversible.

The OSINT Question Is About Access

Public discussion about AI agents often focuses on capability.

Can the agent code?
Can it browse?
Can it reason?
Can it complete a task?
Can it use tools?

Those questions matter. But for investigative and defensive work, the first question is more operational:

```
What permissions does the agent have?
```

An AI agent with no external access can still produce bad information. That is a quality and verification problem.

An AI agent with external access can create a security problem:

• it can read information it should not use;
• it can combine data from separate contexts;
• it can trigger actions faster than a human reviewer can follow;
• it can misunderstand an instruction and over-execute;
• it can delete, overwrite, expose or transmit data;
• it can be influenced by malicious inputs from the environment;
• it can make an action look intentional even when it came from a flawed task interpretation.

This is why the unit of analysis should not be "the model" alone.

The unit of analysis is the agentic system:

```
model + tools + permissions + memory + instructions + environment + logs + human over sight
```

If one of these elements is missing from your notes, your picture is incomplete.

Why DeepMind Uses an Insider-Threat Lens

Google DeepMind says its roadmap treats untrusted AI agents as potential insider threats, similar to a rogue employee who already has access inside an organization.

That comparison is useful because it moves the conversation away from vague fear.

An insider-threat model does not start with the assumption that every insider is hostile. It starts with the fact that insiders have privileges. They can already reach systems, documents, credentials, channels, databases or workflows that outsiders cannot reach.

An AI agent with internal tool access creates a similar planning problem.

The question is not only:

```
Will the model refuse harmful requests?
```

The better questions are:

```
Which actions can it take?
Which systems can it reach?
Which actions require approval?
Which actions are logged?
Which actions can be reversed?
Which actions are blocked in real time?
Which actions are only reviewed after the fact?
```

This is the OSINT value of the roadmap: it gives analysts a way to translate AI safety language into observable system behavior.

Detection Is Not the Same as Control

One of the most important distinctions in the DeepMind post is between detection, prevention and

response.

Detection means the system can notice a concerning action, plan or pattern.

Prevention means it can block the action before it happens.

Response means it can contain, reverse, investigate or escalate after something has happened.

These should not be collapsed.

A company saying "we monitor agents" is not the same as saying "we can stop high-risk actions in real time."

For OSINT work, this distinction gives you a simple claim-checking method.

When an organization describes agent safeguards, break the claim into three columns:

Claim	Evidence to look for	Open question
Detection	logs, alerts, classifiers, supervisor model, behavior monitoring	What behaviors are monitored?
Prevention	permission gates, sandboxing, real-time blocking, human approval	Which actions are blocked before execution?
Response	audit trail, rollback, incident review, escalation path	How fast can the organization act?

The point is not to demand perfect control. That does not exist.

The point is to avoid treating a monitoring claim as if it were a containment claim.

The Evidence Log for an AI Agent

If an AI agent is involved in a mistake, leak, unexpected action or security incident, the investigation needs an evidence log.

Not a transcript dump.

A structured log.

Use this template.

```
Agent:
Name, version, provider, deployment context.

Task:
What was the agent asked to do?

User instruction:
Original instruction, relevant constraints, approval requirements.

System instruction:
Known policy, guardrails, tool-use rules, role boundaries.

Tools:
Which tools were available?

Permissions:
Read, write, execute, send, delete, deploy, invite, export, query.
```

```
Data access:
Which files, systems, channels, repositories or records could the agent access?

Action sequence:
What did the agent do, in order?

Human checkpoints:
Where did a human approve, reject, edit or miss an action?

External inputs:
Which web pages, documents, emails, tickets, prompts or data sources influenced the a
gent?

Observed failure:
Misinterpretation, over-execution, policy bypass, data exposure, deletion, unauthoriz
ed action, hallucinated evidence.

Evidence level:
Direct log, screenshot, audit event, vendor statement, third-
party report, user claim.

Reversibility:
Can the action be undone? How quickly?

Open questions:
What cannot be verified from the available evidence?
```

This template is deliberately boring.

That is the point.

Agent incidents will be difficult to analyze if every discussion begins with abstract terms: autonomy, alignment, intelligence, safety, trust.

A good evidence log forces the investigation back to observable events.

Watch the Difference Between Error and Adversarial Behavior

DeepMind notes that many flagged events in its analysis did not appear to come from adversarial intent. They often came from agent misinterpretation or over-eagerness to achieve a user's goal.

This distinction matters.

If an agent deletes data because it misunderstood a task, that is different from an agent trying to hide a harmful action.

But the operational impact can still be serious.

For investigators and defenders, intent should not be guessed too early. Start with behavior.

Ask:

- What instruction did the agent receive?
- What goal did it appear to optimize?
- Which constraints did it ignore?
- Did it ask for confirmation?
- Did it operate inside its expected scope?

• Did it use a tool in a way the user could reasonably anticipate?
• Did the system warn, block or log the action?

Only after that should you move toward interpretation.

The mistake is to turn every failure into a drama about rogue AI, or every failure into a harmless bug. Both shortcuts hide the evidence.

A Practical Risk Matrix

For a quick assessment, map agent actions across two dimensions:

```
Impact if wrong
Reversibility after execution
```

Low impact and reversible:

• drafting text;
• summarizing public material;
• creating a local note;
• suggesting a query;
• preparing a non-sent message.

Medium impact or partially reversible:

• editing shared documents;
• moving files;
• opening tickets;
• updating metadata;
• querying internal knowledge bases;
• drafting messages in a live workspace.

High impact or hard to reverse:

• deleting data;
• changing permissions;
• sending messages externally;
• deploying code;
• modifying production systems;
• exporting sensitive information;
• connecting new integrations;
• approving transactions or access requests.

This matrix should shape the control level.

Low-risk actions can be reviewed asynchronously.

High-risk actions need real-time approval, sandboxing, or hard blocks.

If an agent can perform high-impact, hard-to-reverse actions with only delayed review, the problem is not the agent's personality. The problem is system design.

What to Check Before Trusting an Agentic Workflow

Before relying on an AI agent in a research, journalism, OSINT or security workflow, ask these questions:

1. ScopeWhat is the agent allowed to do?
2. Tool accessWhich tools can it call without asking?
3. Data boundariesCan it access private, sensitive, unpublished or cross-client data?
4. Action boundariesCan it write, delete, send, publish, deploy or invite?
5. ApprovalWhich actions require human confirmation?
6. LoggingCan you reconstruct the full action sequence?
7. External influenceCan web pages, emails, documents or tickets inject instructions into the workflow?
8. RollbackWhat can be undone, and how fast?
9. MonitoringAre you looking only at outputs, or also at actions and tool calls?
10. Evidence retentionHow long are logs, transcripts and tool events preserved?

This is not only an enterprise checklist.

It applies to smaller workflows too: a research assistant connected to documents, a coding agent inside a repository, a browser agent that can submit forms, or a content agent that can publish drafts.

The moment an AI system can act, the evidence trail becomes part of the safety model.

The Takeaway

AI agents make verification harder because they sit between instruction and action.

They can read, decide, call tools, interpret context and produce outputs that look coherent even when the path was flawed.

For OSINT, that means we need to stop treating agent output as the only artifact worth checking.

The important artifacts are:

• the instruction;
• the available tools;
• the permissions;
• the data accessed;
• the action sequence;
• the human checkpoints;
• the logs;
• the blocked actions;
• the failed actions;
• the rollback path.

An AI agent should not be trusted because it sounds reliable.

It should be trusted only to the degree that its actions are bounded, logged, reviewable and reversible.

That is the real investigative shift.

The question is no longer only:

```
What did the AI say?
```

It is:

```
What did the AI do, what could it have done, and how do we know?
```

The most useful way to investigate an AI agent is not to ask whether it is intelligent.

Ask what it can touch.

An agent that writes a document is one kind of system. An agent that can edit files, call APIs, query internal data, open tickets, deploy code, change permissions, send messages or trigger workflows is another.

The risk changes when the model stops being only a generator and becomes an actor inside a system.

That is why Google DeepMind's recent AI Control Roadmap is worth reading as an OSINT and security document, not only as an AI safety document. The company describes a framework for securing internal systems against increasingly capable AI agents and explicitly borrows from cybersecurity. Its core planning assumption is simple and uncomfortable: treat powerful internal agents as if they could be imperfectly aligned and capable of unexpected behavior.

In cybersecurity terms, the agent starts to look less like a tool and more like a potential insider.

Not because today's agents are all malicious. Not because every automation is dangerous. But because a system with access, goals, tools and partial autonomy creates an evidence problem.

If something goes wrong, you need to know what happened.

And you need to know before the damage is irreversible.

The OSINT Question Is About Access

Public discussion about AI agents often focuses on capability.

Can the agent code?
Can it browse?
Can it reason?
Can it complete a task?
Can it use tools?

Those questions matter. But for investigative and defensive work, the first question is more operational:

```
What permissions does the agent have?
```

An AI agent with no external access can still produce bad information. That is a quality and verification problem.

An AI agent with external access can create a security problem:

- it can read information it should not use;
- it can combine data from separate contexts;
- it can trigger actions faster than a human reviewer can follow;
- it can misunderstand an instruction and over-execute;
- it can delete, overwrite, expose or transmit data;
- it can be influenced by malicious inputs from the environment;
- it can make an action look intentional even when it came from a flawed task interpretation.

This is why the unit of analysis should not be "the model" alone.

The unit of analysis is the agentic system:

```
model + tools + permissions + memory + instructions + environment + logs + human over sight
```

If one of these elements is missing from your notes, your picture is incomplete.

Why DeepMind Uses an Insider-Threat Lens

Google DeepMind says its roadmap treats untrusted AI agents as potential insider threats, similar to a rogue employee who already has access inside an organization.

That comparison is useful because it moves the conversation away from vague fear.

An insider-threat model does not start with the assumption that every insider is hostile. It starts with the fact that insiders have privileges. They can already reach systems, documents, credentials, channels, databases or workflows that outsiders cannot reach.

An AI agent with internal tool access creates a similar planning problem.

The question is not only:

```
Will the model refuse harmful requests?
```

The better questions are:

```
Which actions can it take?
Which systems can it reach?
Which actions require approval?
Which actions are logged?
Which actions can be reversed?
Which actions are blocked in real time?
Which actions are only reviewed after the fact?
```

This is the OSINT value of the roadmap: it gives analysts a way to translate AI safety language into observable system behavior.

Detection Is Not the Same as Control

One of the most important distinctions in the DeepMind post is between detection, prevention and response.

Detection means the system can notice a concerning action, plan or pattern.

Prevention means it can block the action before it happens.

Response means it can contain, reverse, investigate or escalate after something has happened.

These should not be collapsed.

A company saying "we monitor agents" is not the same as saying "we can stop high-risk actions in

real time."

For OSINT work, this distinction gives you a simple claim-checking method.

When an organization describes agent safeguards, break the claim into three columns:

Claim	Evidence to look for	Open question
Detection	logs, alerts, classifiers, supervisor model, behavior monitoring	What behaviors are monitored?
Prevention	permission gates, sandboxing, real-time blocking, human approval	Which actions are blocked before execution?
Response	audit trail, rollback, incident review, escalation path	How fast can the organization act?

The point is not to demand perfect control. That does not exist.

The point is to avoid treating a monitoring claim as if it were a containment claim.

The Evidence Log for an AI Agent

If an AI agent is involved in a mistake, leak, unexpected action or security incident, the investigation needs an evidence log.

Not a transcript dump.

A structured log.

Use this template.

```
Agent:
Name, version, provider, deployment context.

Task:
What was the agent asked to do?

User instruction:
Original instruction, relevant constraints, approval requirements.

System instruction:
Known policy, guardrails, tool-use rules, role boundaries.

Tools:
Which tools were available?

Permissions:
Read, write, execute, send, delete, deploy, invite, export, query.

Data access:
Which files, systems, channels, repositories or records could the agent access?

Action sequence:
What did the agent do, in order?

Human checkpoints:
Where did a human approve, reject, edit or miss an action?

External inputs:
Which web pages, documents, emails, tickets, prompts or data sources influenced the a
```

```
gent?

Observed failure:
Misinterpretation, over-execution, policy bypass, data exposure, deletion, unauthoriz
ed action, hallucinated evidence.

Evidence level:
Direct log, screenshot, audit event, vendor statement, third-
party report, user claim.

Reversibility:
Can the action be undone? How quickly?

Open questions:
What cannot be verified from the available evidence?
```

This template is deliberately boring.

That is the point.

Agent incidents will be difficult to analyze if every discussion begins with abstract terms: autonomy, alignment, intelligence, safety, trust.

A good evidence log forces the investigation back to observable events.

Watch the Difference Between Error and Adversarial Behavior

DeepMind notes that many flagged events in its analysis did not appear to come from adversarial intent. They often came from agent misinterpretation or over-eagerness to achieve a user's goal.

This distinction matters.

If an agent deletes data because it misunderstood a task, that is different from an agent trying to hide a harmful action.

But the operational impact can still be serious.

For investigators and defenders, intent should not be guessed too early. Start with behavior.

Ask:

- What instruction did the agent receive?
- What goal did it appear to optimize?
- Which constraints did it ignore?
- Did it ask for confirmation?
- Did it operate inside its expected scope?
- Did it use a tool in a way the user could reasonably anticipate?
- Did the system warn, block or log the action?

Only after that should you move toward interpretation.

The mistake is to turn every failure into a drama about rogue AI, or every failure into a harmless bug. Both shortcuts hide the evidence.

A Practical Risk Matrix

For a quick assessment, map agent actions across two dimensions:

```
Impact if wrong
Reversibility after execution
```

Low impact and reversible:

• drafting text;
• summarizing public material;
• creating a local note;
• suggesting a query;
• preparing a non-sent message.

Medium impact or partially reversible:

• editing shared documents;
• moving files;
• opening tickets;
• updating metadata;
• querying internal knowledge bases;
• drafting messages in a live workspace.

High impact or hard to reverse:

• deleting data;
• changing permissions;
• sending messages externally;
• deploying code;
• modifying production systems;
• exporting sensitive information;
• connecting new integrations;
• approving transactions or access requests.

This matrix should shape the control level.

Low-risk actions can be reviewed asynchronously.

High-risk actions need real-time approval, sandboxing, or hard blocks.

If an agent can perform high-impact, hard-to-reverse actions with only delayed review, the problem is not the agent's personality. The problem is system design.

What to Check Before Trusting an Agentic Workflow

Before relying on an AI agent in a research, journalism, OSINT or security workflow, ask these questions:

1. ScopeWhat is the agent allowed to do?
2. Tool accessWhich tools can it call without asking?
3. Data boundariesCan it access private, sensitive, unpublished or cross-client data?
4. Action boundariesCan it write, delete, send, publish, deploy or invite?
5. ApprovalWhich actions require human confirmation?
6. LoggingCan you reconstruct the full action sequence?
7. External influenceCan web pages, emails, documents or tickets inject instructions into the

workflow?
8. RollbackWhat can be undone, and how fast?
9. MonitoringAre you looking only at outputs, or also at actions and tool calls?
10. Evidence retentionHow long are logs, transcripts and tool events preserved?

This is not only an enterprise checklist.

It applies to smaller workflows too: a research assistant connected to documents, a coding agent inside a repository, a browser agent that can submit forms, or a content agent that can publish drafts.

The moment an AI system can act, the evidence trail becomes part of the safety model.

The Takeaway

AI agents make verification harder because they sit between instruction and action.

They can read, decide, call tools, interpret context and produce outputs that look coherent even when the path was flawed.

For OSINT, that means we need to stop treating agent output as the only artifact worth checking.

The important artifacts are:

• the instruction;
• the available tools;
• the permissions;
• the data accessed;
• the action sequence;
• the human checkpoints;
• the logs;
• the blocked actions;
• the failed actions;
• the rollback path.

An AI agent should not be trusted because it sounds reliable.

It should be trusted only to the degree that its actions are bounded, logged, reviewable and reversible.

That is the real investigative shift.

The question is no longer only:

```
What did the AI say?
```

It is:

```
What did the AI do, what could it have done, and how do we know?
```