

Buying an AI SOC?

Ask these 70 questions first.

A practical vendor-evaluation guide

With a companion assessment worksheet

Jessen Kurien

Cybersecurity Leader · Practitioner · Author

Jessen brings 18+ years of cybersecurity experience, including nearly 15 years at Microsoft in leadership and technical roles, SOC leadership at a global MSSP, and a consulting engagement supporting Cisco.

CISM · CISA · ISO/IEC 42001:2023 Lead Implementer · ISO/IEC 42001:2023 Lead Auditor

Version 1.12 · Last reviewed 27 September 2026

Free and open. No sign-up. jessenkurien.com

Self-contained — no prior reading required.

Contents

Buying an AI SOC? Ask these 70 questions first. · Version 1.12. Every entry below is a link.

Cutting through the demo	2
Where to start — ten questions for the first meeting	7
What a good answer sounds like — the ten in detail	8
1. Evidence — can you prove what the agent did?	12
2. Authority — what is it allowed to do?	13
3. Failure — what happens when it is wrong?	15
4. Vendor claims and testing	17
5. Speed and accuracy	18
6. Correlation — does it add detection value, or just sort your queue?	20
7. Supply chain — the software your product is built from	21
8. Ransomware	22
9. Data theft	24
10. Moving between systems	24
11. Cost	25
12. Change management	27
13. Exit	27
Four things to ask for, not ask about	29
Recording the answers	31
What to do with the answers	32
Sources	32
About the author	34
Worksheet — print this page	35
Version history	36

Cutting through the demo

An AI SOC demo can show useful capabilities: an agent reviews an alert, gathers context, explains its assessment and closes a case. Evaluating it for production also requires evidence of how it handles incomplete telemetry, incorrect decisions and operational disruption in your environment.

A demonstration is a starting point. The purchasing decision also depends on failure handling, the permissions an attacker could misuse, the evidence you retain after exit, and the cost of unexpected behaviour. Establish these alongside the product's expected benefits.

These are the seventy questions I would take into an evaluation with the vendor's engineering and account teams. Expect clear answers supported by evidence; acceptable designs will vary by deployment. Record unresolved questions, agree who will answer them, and use the proof of concept to test the claims that matter most.

What you are actually evaluating

"AI SOC" is sold as one category and covers at least four different products. Establish which one is in front of you before you ask anything else, because many of these questions change meaning between them.

Investigation assistant. Summarises, enriches and drafts. A human does everything that changes state. The risk is being wrong in a way an analyst believes.

Automated triage. Decides which alerts close and which escalate, and writes those decisions into your case system. The risk is what it closes.

Acting agent. Holds permissions to change your environment — isolate a host, disable an account, quarantine mail, revoke a session. This is where the Authority and attack-scenario sections earn their place.

Managed service. The vendor's people and the vendor's agents, operating inside your environment. Everything above applies, and the vendor's own staff are now an identity in your tenant.

Cutting across all four is how much the product does on its own:

Level	What it means
Read-only	It looks and reports. Nothing changes.
Recommend	It proposes; a human approves every action.

Level	What it means
Bounded autonomy	It acts within a written policy; a human reviews afterwards.
Autonomous	It acts under an enforced policy without waiting for anyone, and you learn of it from the record. Boundaries still exist; no one is asked in advance.

The top rung is not the absence of limits. A product acting autonomously still operates inside a policy; what changes is that nobody is consulted before it acts. Ask which specific actions require prior human approval, which proceed under enforced policy, and what the product does when an action falls outside both.

Ask the vendor to place their product on that ladder and to say precisely what moves it up a rung. "Configurable" is not an answer on its own — ask who can change it, whether the change is logged, and whether you are told.

Which sections apply to what. Evidence, Vendor claims, Correlation, Supply chain, Cost, Change management and Exit apply to all four. Authority, Failure and the attack scenarios apply in full from automated triage upward and thin out for a read-only assistant. Speed and accuracy applies wherever the product closes or prioritises alerts. For a managed service, press hardest on the vendor's own access and on who does the work after go-live.

Mark a question not applicable in writing, with the reason and the date. "Not applicable, product is read-only, confirmed by the vendor on 14 October" is a record you can revisit when someone proposes enabling autonomy nine months later. In a spreadsheet six months on, an unanswered question and an inapplicable one look identical, and they are not the same thing.

Two questions underneath all seventy

Two questions guide the evaluation: how the product improves your defence, and how you will control the risk if the product itself is compromised. Evaluate both with the vendor.

The first is **what does your agent do about this attack?** That is the question the product was built to answer, and the demo is its answer.

The second is **what does your agent become if the attacker reaches it first?** An AI SOC product holds some of the most dangerous permissions in your company. It can disable accounts, isolate hosts, quarantine mail, revoke tokens and in some cases change your detection content. If any part of it is manipulated, the attacker does not need to reach production. Your SOC reaches production on their behalf.

There is a specific reason this is sharper in a SOC than anywhere else: **your telemetry is attacker-influenced content.** Alert fields carry filenames, URLs, email subjects, command lines, ticket text and user-supplied strings. Someone who wants to reach your triage agent does not need to breach anything.

They need to generate an alert containing text the agent will read.

Three gaps to resolve before granting authority

Resolve these three gaps before granting operational authority, including automated triage. For a read-only assistant, assess their impact against the narrower scope. Agree the required evidence and retest before proceeding.

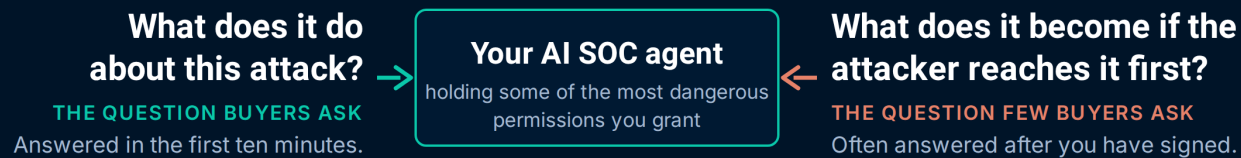
"The model decides." (question 10) If the model is the last thing between a decision and an action, then the safeguards you were shown depend on the model continuing to behave as it did in the demo. That is a tendency, not a boundary, and tendencies change when the model does.

No answer on what it wrongly closed. (question 15) A false-positive rate does not establish how often real incidents were wrongly closed. Ask for that measure, its methodology and the process used to identify missed incidents.

Evidence available only in the vendor's console. (question 1) Without an independent activity record, investigations depend on access to the vendor's console. Establish how evidence remains available during an outage and after the contract ends.

Those three become critical rows when you record the answers, along with three more the rest of this piece argues for: the permission boundary (question 7), the off-switch (question 8) and the liability position (question 22).

The demo answers one of these. The deal turns on the other.



Jessen Kurien · jessenkurien.com · AI SOC vendor evaluation guide

Figure 1 — The two questions underneath all seventy.

Three gaps to resolve before granting authority

Resolve these gaps before enabling automated triage or response.

01 · QUESTION 10

“The model decides.”

WHY IT MATTERS

The safeguards rest on the model's judgement, not on a boundary it cannot cross. That is a tendency, and tendencies change.

02 · QUESTION 15

“We do not measure what we wrongly closed.”

WHY IT MATTERS

A high closure rate is a volume, not a verdict. Closing more does not show better judgement unless somebody checks.

03 · QUESTION 1

“The evidence is only in our console.”

WHY IT MATTERS

Independent records support investigations during outages and after the contract ends. Confirm how you will retain access.

Verify the answer with evidence: inspectable enforcement, an exported case, a run on your own alerts, an injected-alert test you watch.
Critical rows 10, 15 and 1 on the worksheet, alongside 7, 8 and 22.

Jessen Kurien · jessenkurien.com · AI SOC vendor evaluation guide

Figure 2 — Three gaps to resolve before granting authority, and the critical rows they map to.

Where to start — ten questions for the first meeting

Seventy questions is a procurement project. Ten is a meeting. Use these ten to structure an initial meeting and identify the evidence needed for a deeper evaluation. The numbering matches the full list, so you can move between the two.

#	The question	What it separates
1	Which events do you record, and can I get a copy into my own log system?	Whether you can ever investigate the product
7	What account does it use, what can it reach, and who at your company can use it?	The size of the permission you are granting
8	Can I switch it off in one action, and how long before it stops working?	Whether containment is real
10	What enforces the rules between the model deciding and the action happening?	Enforced control
15	How often does it close a real incident as benign, and how did you measure that?	Whether they measure the failure that hurts
16	What can it do if an attacker plants instructions in an alert it reads?	Whether they have modelled themselves as a target
25	Can I run the evaluation myself, in my environment, on my data?	Whether the numbers survive contact
30	Does it detect anything itself, or act on alerts my tools already generated?	What you are actually paying for
58	Is pricing per alert, per user, per unit of AI usage, or flat, and whose account is billed?	Who absorbs a bad week
69	What do I keep if I leave?	Evidence available after exit

Then use the rest in three passes.

Technical evaluation, with your architects and detection engineers: Evidence, Authority, Failure, Supply chain and Change management in full. These decide whether the product is safe to install, not whether it is good.

Proof of concept: Vendor claims and testing, Speed and accuracy, Correlation, the attack scenarios, and the four artifacts at the end. These decide whether it works in your environment.

Procurement and legal: question 22 on liability, the Cost section, and Exit. Bring these people in before the technical evaluation finishes, not after.

What a good answer sounds like — the ten in detail

For the ten above, here is what separates an answer you can accept from one that only sounds reassuring. Several architectures can satisfy each of these; what follows is the shape of a credible answer rather than a single correct one. Record the demonstration and your judgement of it separately — a vendor can evidence something perfectly and still give you an answer you should refuse.

1 · Events into your own log system

Acceptable: names which event types the product emits, offers a documented schema, and describes a supported path into a SIEM or object store you control, at a stated frequency. *Ask for:* the field list or schema, plus one real example of each event type. *Warning sign:* "we can build an integration for that" with no existing customer running it, or export available only as a CSV download from their interface. *Test:* land a day of events in your own store during the proof of concept and answer a real question from them without opening the vendor's console.

7 · The account, and who at the vendor can use it

Acceptable: the permission set in writing with a reason for each, named break-glass roles on the vendor side, an approval step for them, and their access visible in your own logs. *Ask for:* the role definition or application manifest, and their internal access-approval process. *Warning sign:* one broad administrative role because "it makes deployment simpler", or support access to your tenant that is mentioned in the contract and logged nowhere you can see. *Test:* read the permissions out of your own identity platform rather than out of their document.

8 · The off-switch

Acceptable: one documented action, a propagation time they are willing to measure with you, and a clear account of what happens to work already in flight and to any child agents. *Ask for:* the runbook, and a witnessed execution rather than a screenshot. *Warning sign:* "just disable the integration", with no answer on tokens already issued, actions already queued, or agents the main agent has spawned. *Test:* a timed revocation in your tenant with an active session and a queued action, and check both stopped.

10 · What enforces the boundary between the model and the action

Acceptable: names the component that enforces it — a policy engine, typed permissions on the action API, a signed configuration, an approval service — shows you its current state, and can show that changes to it are versioned and attributable. *Ask for:* the current configuration, a diff from last month, and the list of people who can change it. *Warning sign:* "the model is trained to refuse that", "there are

guardrails", "it's in the system prompt". Those describe a tendency, not a boundary. *Test*: attempt a disallowed action during the proof of concept and confirm the refusal comes from the enforcing component, not from the model's judgement on the day.

15 · What it wrongly closed

Acceptable: a stated denominator, a stated period, a named adjudicator, and a described process for discovering that a closure was wrong. *Ask for*: the methodology and the raw counts, not the percentage. *Warning sign*: a false-positive rate quoted to one decimal place with no false-closure figure beside it, or "no customer has reported one", which measures reporting rather than accuracy. *Test*: seed cases with known outcomes into your evaluation set and count what it closed.

16 · An attacker planting instructions in an alert

Acceptable: describes what the agent could reach before anyone noticed, names the control that bounds it, and can point to their own testing. *Ask for*: their test results or red-team findings on this specific path. *Warning sign*: treating it as hypothetical, or answering entirely in terms of input filtering — filtering is worth having and is not the control. *Test*: one alert with planted instructions, run in their sandbox with their knowledge, while you watch.

25 · Running the evaluation in your environment

Acceptable: a defined proof of concept with scope, duration, cost, data handling and exit, run on your alerts with your analysts scoring the results. *Ask for*: the plan, and a reference customer who ran one and will talk to you. *Warning sign*: evaluation only on their curated dataset, only in their environment, or a trial that requires production permissions from day one. *Test*: the proof of concept itself, built as described later in this piece.

30 · Whether it detects or only triages

Acceptable: a straight statement of which headline numbers come from detection the product performs and which from alerts your existing sensors generated. *Ask for*: the provenance of each metric in the deck. *Warning sign*: a time-to-detect improvement measured from the moment an alert reached their product. *Test*: during the proof of concept, check whether any finding originated with the product rather than with something you already owned.

58 · Pricing and whose account is billed

Acceptable: a named unit, a clear statement of whose account absorbs usage, and defined behaviour when a ceiling is reached. *Ask for*: a sample invoice or usage report from a comparable customer, with identifiers removed. *Warning sign*: "it depends on usage" with no unit economics, or consumption billed to your cloud account with no cap and no alert. *Test*: measure spend per case through the proof of concept and extrapolate to a bad week rather than an average one.

69 · What you keep if you leave

Acceptable: named export formats for detection logic, evidence and case history, a stated window after termination, and no additional fee. *Ask for:* an actual export from a comparable tenant, in the format you would receive. *Warning sign:* "we'll work with you at the time", or an export that arrives as PDF, or only through a professional services engagement. *Test:* run the export during the proof of concept. The end of a contract is the worst moment to discover what it contains.

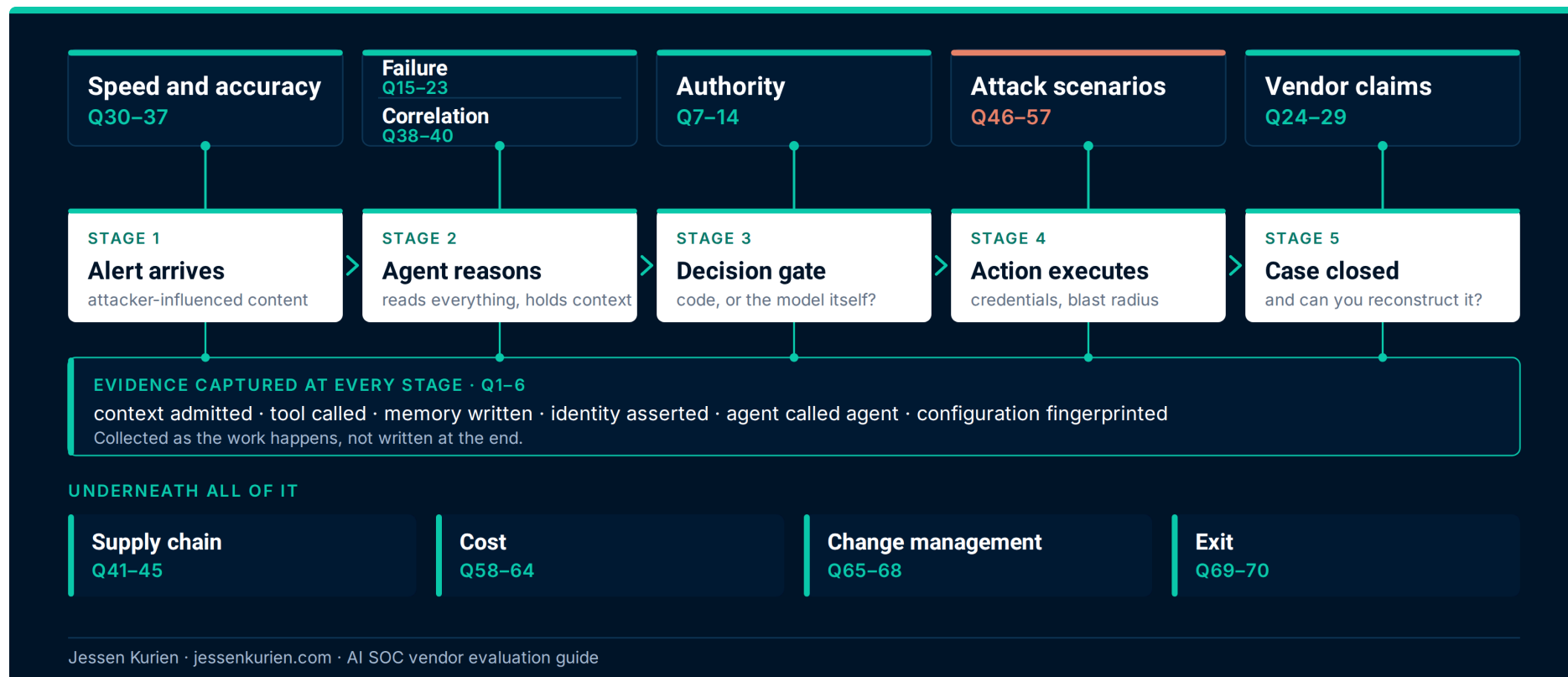


Figure 3 — Where each question applies to the product. Five stages of an AI SOC product, with the question groups that assess each one, the evidence that has to be captured across all of them, and the four groups that sit underneath the whole thing.

1. Evidence — can you prove what the agent did?

Start here, because every other answer depends on it. An independent activity record lets your team verify behaviour and investigate incidents without relying solely on the vendor's console.

1. Which events do you record, and can I get a copy into my own log system?

Six events carry most of the value — context admission, tool call, memory write, identity assertion, agent-to-agent call and configuration fingerprint — within a wider set of sixteen event families that covers the agent activity worth logging across the platforms surveyed. [5] It is a working taxonomy, not a proof of completeness: a product may emit something none of the sixteen anticipates, which is itself worth asking about. Ask which of these the product emits, and then ask the harder half of the question: can those events land in your SIEM or data lake, in a documented schema, without an export button and a spreadsheet.

2. Do you log the tool call itself, or only the conclusion the agent reached?

A tool call is the agent doing something — running a query, disabling an account, pulling a file. An outcome summary is useful, but it does not replace a record of the actions taken. When you are investigating six months later, the narrative is the agent's account of itself and the tool calls are the evidence. If only one of those survives, you have a story rather than a record.

3. Can I reconstruct a closed alert six months from now without logging into your console — and could the record have been changed in the meantime?

This is retention, portability and integrity in one question. Ask what the retention period is, whether it is configurable, what it costs to extend, and whether the records are complete enough on their own that an investigator who has never seen the product could follow what happened. Then ask who can alter or delete an activity record, whether the product itself can, and how an investigator would know it had happened. Undetected changes to a record undermine its evidential value.

4. What do you not log?

Most products have gaps, and a vendor who names theirs is telling you they have thought about it. The common omissions are worth listening for: reasoning steps, retries, failed tool calls, content the agent read but did not act on, and anything the agent did while an error was being handled.

5. How long from the agent acting to me being able to search for it — seconds, minutes, or next day?

Evidence you cannot query during an incident is evidence for the post-mortem. If the agent's activity reaches your logging platform on a nightly batch, then for the entire duration of an incident your most privileged automated actor is invisible to the people running the response.

6. Do your records carry the same identifiers as my own logs?

User, host, case identifier, timestamp, request identifier. Without shared fields you cannot join the product's activity to your own telemetry, which means you cannot answer the only question that matters during an incident: was this action ours, and what did it touch. Without shared identifiers, reconstructing the activity becomes slower and less reliable.

2. Authority — what is it allowed to do?

An AI SOC product is an identity in your environment with permissions attached. Assess its permissions as carefully as its detection quality. Detection quality determines the value you get. Authority determines the damage you absorb.

7. What account does the product use inside my environment, what can that account reach, and who at your company can use it?

Ask for the actual permission set, in writing, not a description of it. Then ask why each permission is needed — the gap between what a product requests at install and what it actually uses is usually large, and usually never revisited. The second half of the question is the one people forget: which of the vendor's own staff can read your data, change your configuration or trigger an action, how that access is approved, whether it is logged on your side as well as theirs, and whether you are told after the fact. For a managed service this is not a side question; it is the main one.

8. Can I switch that account off in one action, and how long before it actually stops working?

Disabling an account and revoking its access are not the same event, and the gap between them is your real containment window. Microsoft documents a default access-token lifetime of 60 to 90 minutes on the Microsoft identity platform, and two hours for some clients in tenants without Conditional Access; where continuous access evaluation applies, a disabled account is a critical event that propagates in near real time, though Microsoft still notes latency of up to fifteen minutes. Your platform and configuration will differ. Measure the actual containment window with the vendor in your tenant, using an active session. Published timings are a starting point, not a substitute for that test. [1] [2]

9. What can it do without a human approving it — and is that something I configure, or something you promise?

There is a real difference between "the agent will not do that" and "the agent cannot do that". The first is a behaviour, and behaviours change when models change. The second is a permission boundary. Ask which one you are being offered, and whether you can see it in a settings page.

10. Between the AI deciding and the action happening, what actually enforces the rules — and can I inspect it?

This establishes whether an independent control enforces the action boundary. If the model is the last thing that decides, the safeguards rest on its judgement rather than on something it cannot cross. What you are checking is not the existence of a file — a policy engine, a signed configuration, typed permissions on the action API, an approval service and several other designs all qualify. What every acceptable answer has in common is that the constraint lives outside the model, that you can read it, that changes to it are versioned and attributable, and that it holds when tested. Ask which component enforces it, ask to inspect the current state, ask who can change it, and ask what changed last month.

11. Can it change my detection rules, my logging settings, or how long I keep logs — and what stops a change nobody approved?

Tuning detections is legitimate work, and a product that proposes improvements to your rules may be worth having. What is not acceptable is a change that happens without a named approver, a version history and a way back. An agent that can quietly tune down a noisy rule or reduce sampling to save cost reduces your visibility, and the resulting silence looks exactly like an improvement. It is also the last stage of a competent attack. So ask for the specifics: can it propose, or can it apply? Who approves, and is that person independent of the team whose alert volume is being reduced? Is every change versioned, diffable and revertible in one step? And does a change to detection or logging content itself raise an alert somewhere the agent cannot reach?

12. Can it hand work to other agents or create new ones — and does my off-switch stop those too?

Multi-agent products spawn helpers: an enrichment agent, a summariser, a hunter. Ask whether those inherit the parent's permissions, whether they appear in the logs under their own identity, and whether pulling the kill switch stops them or leaves orphaned children still holding credentials.

13. Does it save anything between cases — notes, memory, a knowledge base? Who can read and write that, and can anything learned in my tenant surface in another customer's investigation?

Persistent memory is how a product gets better and also how a wrong conclusion becomes permanent. Ask where it is stored, whether it records the origin of each entry, whether entries expire, and whether an agent that reads untrusted content can write into a store that a more privileged agent later reads. That last one is the delayed attack: the write happens today, the damage next month, and the investigation looks in the wrong fortnight. Then ask the multi-tenant version of the same question: what technically prevents context stored from my environment — hostnames, user names, case text, learned patterns — from influencing an investigation in someone else's. "Logical separation" is a phrase, not a control; ask what enforces it.

14. Can it act on systems outside the scope we agreed, if it decides they are relevant?

Investigations sprawl by nature — an alert on a laptop leads to a file server leads to a domain controller. Ask whether scope is a hard boundary enforced by permissions or a guideline in a prompt, and what the product does when the trail leads somewhere it is not allowed to go.

3. Failure — what happens when it is wrong?

Products in this category are sold on what they catch. The evaluation should be about what they miss and what they break, because those are the costs you will carry and they are rarely on the slide.

15. How often does it close a real incident as benign, and how did you measure that?

Measuring wrong closures is essential to understanding missed incidents and delayed containment. **Ask for the formula in writing**, because two different figures get quoted under the same name:

Figure	How it is calculated
Wrong-closure rate	malicious alerts closed as benign ÷ malicious alerts evaluated
Closed-queue contamination	malicious alerts closed as benign ÷ all alerts closed as benign

The numerator is the same count in both. Only the population underneath changes, and the two populations answer different questions, which is why you want both.

The first measures **judgement**: of the malicious alerts this product looked at, what proportion did it wave through? That is the number to compare between products, and the one to watch after a model change.

The second measures **exposure**: of everything sitting in your closed queue, what proportion was incorrectly closed as benign? That is the number your incident commander needs, because it sizes what the closed pile is hiding.

Which figure is larger depends on the ratio of the two populations in your environment, so neither can be inferred from the other and neither can be compared with anyone else's unless both denominators are stated. Ask for both, with the period, the adjudicator, and the raw counts rather than the percentages — and ask whether the numerator came from an adjudicated sample or from incidents that happened to surface on their own, because the second depends on how hard anyone looked. A vendor who says they do not know is being honest; a vendor with a process — sampled re-review by humans, published results — is telling you something about how they run.

16. Suppose an attacker plants instructions inside an alert your agent reads. What can it do before anyone notices?

Your alerts are full of attacker-supplied text. An attacker who wants to reach your triage agent does not need to breach anything, only to generate an alert containing text the agent will read. Introducing an agent creates this additional attack path. Ask how the vendor has assessed it and which controls limit its impact.

17. If one decision is wrong, how many systems can it touch before something stops it?

Ask for the specific ceilings: isolations per hour, accounts disabled per hour, messages purged per action, and what happens when a limit is reached. A product with no ceilings is one manipulated alert away from becoming a denial-of-service tool pointed at your own estate, operated by your own SOC.

18. Show me a case where the agent got it wrong.

Ask for the case, the investigation and what changed afterwards. The point is not the mistake, it is whether they have a process for finding mistakes at all. If examples cannot be shared, ask for anonymised test findings and the process used to identify and review errors.

19. Has it ever taken an action you did not intend — in a customer environment or in testing?

This is a different question from the last one, and the distinction matters. Getting an answer wrong is a judgement failure. Taking an action nobody sanctioned is a control failure, and control failures tell you about the architecture rather than the model.

20. Walk me through shutting your product down during an incident.

Not disabling the integration — containing the product while it is mid-flight on two hundred cases. Ask what happens to work in progress, whether actions already issued complete or roll back, how long the whole thing takes, and whether anyone has ever done it outside a slide.

21. What happens when your product is the thing the attacker manipulated?

Ask for the containment story and the residual risk they accept. A vendor who has thought about this has an answer ready and will usually volunteer the limits of it. If the answer is unclear, request the relevant threat model, test evidence and incident-response procedure.

22. If it takes a damaging action, who is liable — you or us? Show me the clause.

Ask for the actual contract language rather than a verbal assurance, and read what it excludes. In the agreements I have read, the risk sits with the customer. That may be acceptable, but it should be a decision you made rather than one you discovered afterwards.

23. When your product is down, what happens to the alerts it would have handled?

Do they queue, and for how long? Do they fall through to your analysts, who by then may have reorganised around the product's existence? Does the queue drain in order or by priority when service returns? A triage layer that fails silently and leaves a gap is worse than one you never installed, because you staffed around it.

4. Vendor claims and testing

The numbers in the pitch deck were produced under conditions you are unlikely to replicate. That does not make them false, it makes them uninterpretable until you know how they were made.

24. Is your headline number typical or your best case — what was the spread, across how many environments, over what period, and how many of them look like mine?

Ask for the middle of the distribution and the worst result they have seen, not the average. An average built from one exceptional deployment and nine ordinary ones tells you about the exceptional one. Then ask what those environments were: size, sector, tooling, alert volume, analyst headcount. A product validated across thirty small cloud-native companies is a different proposition for a regulated enterprise with twenty years of accumulated infrastructure, and it will fail in different places.

25. Can I run the evaluation myself, in my environment, on my data?

The most relevant benchmark reflects your environment and operating conditions. Push for a proof of concept on your own historical alerts with your own analysts scoring the results, and treat reluctance as information. Ask what it costs, how long it takes and what you have to give them to make it work.

26. If alerts arrive late or humans respond slowly, how much worse do your numbers get?

Results in this field are usually sensitive to timing assumptions that are rarely stated. Ask what happens to the numbers when an alert lands twenty minutes late, when an approval sits unanswered overnight, or when the enrichment source they depend on is slow. The first thing to break tells you what the product actually relies on.

27. What does the agent do when it does not recognise the user or machine in an alert — act anyway, stop, or escalate?

Unknown assets are the normal state in most enterprises, not the exception. A product that behaves sensibly only when its context is complete will spend a lot of its life in the state you did not test.

28. Can the analyst see why it decided something, at the time, or only afterwards in an audit trail?

Reasoning that surfaces only in an export is not available to the person who has to accept or override the decision in the moment. Ask what the analyst sees on screen, how much of it is the model's own account of itself, and whether the underlying evidence is one click away or three.

29. On our most sensitive systems, does the product do less rather than more? What is still automatic there?

The right answer is that autonomy narrows as consequence rises. Ask how the product knows which of your systems are sensitive, whether that classification comes from you or is inferred, and what remains automatic on a domain controller or a payment system.

5. Speed and accuracy

This is the section vendors are most ready for, which is exactly why the numbers alone decide nothing. Before any of the questions below, establish one rule for the conversation: **for every timing number you quote, name both endpoints**. From attacker activity to detection, from alert creation to acknowledgement, from acknowledgement to action, from action to confirmed containment. These are wildly different numbers and they are routinely presented as the same one.

Hold one more distinction through this whole section. **An alert the product saw and wrongly closed is a different failure from an attack that never produced an alert at all**. The first is visible in a review of closed cases. The second is not visible in that sample and requires separate detection-coverage testing. Accuracy figures in this market are usually drawn from the first alone.

30. Does your product detect anything itself, or does it act on alerts my existing tools generated?

Most products in this category sit downstream of detection. Your endpoint tool or SIEM found the thing. For any claimed improvement in detection time, establish whether the product generates the finding or accelerates handling of an existing alert. Attribute each contribution separately. It also sets a hard limit on every accuracy figure they will show you: a product that only triages cannot see an attack that generated no alert, and no amount of reviewing its closed cases will reveal one.

31. Time to acknowledge and time to triage — what were they before your product, in the same customers, measured the same way?

A before-and-after from different customers measured by different methods is not a comparison. Ask for paired numbers from the same environments, and ask who measured them. Measure the improvement under comparable conditions before using it in the business case.

32. Time to contain — is that the agent issuing the action, or the target system confirming the change took effect?

These are not the same number and the gap between them is where incidents live. A tool reporting success is a claim; the endpoint's own record is evidence. The product may report API acceptance because that event is directly visible to it; target-system confirmation requires additional telemetry. Whether the endpoint was reachable, whether the agent was healthy, whether the change survived a reboot are separate questions, and in my experience few products join the action back to the target system to answer them.

33. Time to resolve — define it.

Case closed, threat removed, or system back in service? Each is a legitimate measure and each produces a wildly different number. Whichever they choose, make sure it is the same one your own reporting uses, or you will spend a year explaining to your board why two dashboards disagree.

34. How many alerts does it correctly dismiss as false alarms, and what is that as a percentage of everything it saw?

Ask for the denominator every time. "Dismissed forty thousand false positives" means nothing without how many alerts it handled, how many were genuinely benign, and how many a human checked. Volume alone does not establish accuracy or operational value.

35. Of the alerts it closed, how many turned out to be real — and what is your process for finding that out?

This is the previous question's shadow and the more important half. For a given level of discrimination, a product tuned to close more will report fewer false positives and miss more real incidents; the trade runs the other way when it is tuned to escalate more. A product can improve both at once by genuinely getting better at telling them apart — which is the thing worth paying for — but you cannot tell which is happening from one number, and the false-positive figure is the one that improves for the wrong reason. Ask *how* they discover a wrong close: sampled re-review by humans, customer reports, incidents traced back to a closed case. If there is no process, the accuracy numbers are unaudited by construction. Keep the answer separate from detection coverage in your notes — this question measures what the product got wrong about what it saw, and says nothing about what it never saw.

36. Does it get faster or more accurate over time in my environment — and if so, from what?

Learning from your analysts' decisions is valuable and raises a data question. Learning from other customers' environments raises a different one. Ask which is happening, whether you can turn it off, and what happens to that learned behaviour when you leave.

37. Which of these numbers will you commit to in the contract, and what happens when they are missed?

The distance between a marketing claim and a service level is the distance between a hope and an obligation. Accuracy commitments depend on scope and measurement conditions. Agree what can be measured and committed to, including availability, latency and support response, with procurement and legal.

6. Correlation — does it add detection value, or just sort your queue?

Assess whether the product improves triage, adds detection coverage, or does both. Each can provide value, but they require different evidence.

38. Does it find relationships across alerts that my rules do not connect?

Same user across three tools, same machine over four days, the same unremarkable behaviour appearing in six places. Most detection logic works within a single product and a short window. Ask for concrete examples of cross-tool, cross-time correlations it produced, and how it decided they were related rather than coincidental.

39. When it finds one, do I get it back as a rule I can keep and run myself?

This is the question worth pressing hardest in the whole section. A product that finds patterns in your environment but will not hand them back as detection logic you own is renting you insights drawn from your own data. Ask what format they come in and whether they survive the end of the contract.

40. Show me a correlation it found in a customer environment that their existing rules missed.

Ask what it took to find, how long the signal had been present, and what the customer did with it afterwards. A good answer is specific and slightly unglamorous. If the answer is unclear, separate the demonstrated triage benefit from any additional detection value that remains unproven.

7. Supply chain — the software your product is built from

An AI SOC product is not one piece of software. It is a model, a runtime, a set of connectors, a tool catalogue and a stack of third-party packages, several of which update themselves. Your vendor assessment covered the company. These questions cover what the company ships.

41. List every third-party component, plug-in and connector your product uses, with versions and publishers.

Ask for it in writing. Check that the inventory is maintained and covers the components actually deployed. Pay attention to connectors published by individuals rather than organisations.

42. Do those update themselves inside my environment? What changed last month?

An integration that updates on its own schedule is a change to your security stack that nobody in your change process approved. Ask whether updates are pulled automatically, whether you can pin them, and whether anyone would notice if a component's behaviour changed.

43. Do you lock those components to a fixed version, or take whatever the registry serves that day?

In September 2025 an npm package called `postmark-mcp` built trust over fifteen releases and then, at version 1.0.16, began silently copying email it handled to an external server. The package impersonated Postmark; Postmark states it had not published its own MCP server on npm before the incident and that its legitimate API and services were unaffected. That distinction matters when you tell this story — the lesson is not that a vendor was breached, it is that a package with a trusted-looking name accumulated fifteen releases of good behaviour before it needed to be malicious, and nothing about its identity changed when it was. Only the version did. [3]

44. If one of those components is compromised, what limits how far it reaches into my environment?

The answer should be about isolation and permissions rather than vetting. Vetting tells you a component was acceptable when you looked. Isolation determines what happens when it stops being acceptable, which is the case you are actually buying protection against.

45. Do you publish a component inventory for the agent itself, including the model and prompt versions?

A software bill of materials — a parts list for the product — is becoming standard, and for an agent it needs two entries that traditional lists omit: which model version is running, and which version of the instructions it is running under. Both change the product's behaviour and neither is usually tracked.

8. Ransomware

The scenario section exists because a general question about capability produces a general answer. Naming the attack forces the vendor to describe specific behaviour, and specific behaviour is something you can evaluate.

46. Walk me through what your agent does in the first ten minutes of a live encryption event. Which of those actions are automatic?

Ask for the sequence, with the automatic steps marked. This is the scenario where automation is most valuable and most dangerous at the same time, and the shape of the answer tells you how much the vendor has thought about consequence rather than speed.

47. Can it cut a production server off the network — and who decided that was allowed?

Isolating an encrypting file server is often the right call and occasionally a bigger outage than the ransomware. Ask whether the product knows which of your systems cannot be isolated without an approval, where that list comes from, and who maintains it once you are live.

48. What happens to your triage and your costs at fifty thousand alerts an hour?

Ransomware generates alert storms, and an agent that reasons about every alert individually is the component most likely to fall over exactly when you need it. Ask what degrades first, whether there is a batching or sampling mode, and what it costs to run through a storm.

49. What can its account do to backups and the recovery environment – see them, or change them?

Separate the two. Read-only visibility into backup status is often genuinely useful during a ransomware investigation: knowing whether last night's job completed, and which systems are recoverable, changes the containment decision. The ability to modify, expire, unlock or delete a backup is different. There are legitimate reasons a product might hold a narrow version of it — triggering an out-of-band snapshot before containment, for instance — but it should be explicitly approved, scoped to the specific operation, logged where the product cannot reach, and never a side effect of a general integration role. Ask which permissions the account actually holds against the backup platform, whether read-only is enforced by design or merely by current configuration, and what would have to happen for that to change. Then ask the same question about the recovery environment itself.

9. Data theft

50. Where is your agent allowed to send data — a short approved list, or anywhere not explicitly blocked?

Default-deny with a named allowlist is the answer to push for. A blocklist means every destination nobody thought of is permitted, and the destinations nobody thinks of are the ones an attacker chooses.

51. Does our alert data, case content or evidence leave our environment? Which fields?

Ask for the field list rather than the assurance. Alert content contains hostnames, usernames, file paths, email subjects and sometimes the contents of documents. Where it goes, who can read it there, how long it stays and whether it is used for anything besides serving you are four separate questions.

52. If it is manipulated, how much data can it move out before something stops it?

This is the volume version of the blast radius question. Ask about rate limits, per-case ceilings and whether anything alerts on unusual outbound volume from the product itself — which is a control almost nobody has, because the product is the trusted thing.

53. Does it fetch external content while investigating — images, links, lookups?

Fetching an attacker's URL is itself a way for data to leave, and it needs no exfiltration channel of its own. This is how CVE-2025-32711 worked — an AI command injection in Microsoft 365 Copilot, published in June 2025, scored 7.5 by NIST and 9.3 by Microsoft, and classed as CWE-74. [4] The researchers who reported it named it EchoLeak: the user never opened the message, the assistant retrieved it on their behalf, and the data left inside a request for an image in the answer. [6]

10. Moving between systems

54. Does it use separate accounts per environment, or one account spanning production, corporate and cloud?

One identity that reaches everything is convenient to deploy and is also one of the shortest paths an attacker will find in your estate. Ask how the product is segmented, and whether segmentation is available on your licence tier or is a feature of the enterprise edition.

55. Can it pivot from an alert in one business unit into another?

Relevant for anyone with subsidiaries, regional separation, regulated units or a recent acquisition. The agent's willingness to follow a trail is a feature until the trail crosses a boundary that exists for legal reasons.

56. Can it call other agents, and do those inherit its permissions?

If the answer is yes without qualification, then the permissions of your most privileged agent are effectively the permissions of your least careful one. Ask whether the origin of the original request travels with the call, so you can answer afterwards who actually asked for an action.

57. What is the shortest path from your agent to full administrator rights — have you drawn it?

Vendors who have done this exercise say so immediately and usually have a diagram. If the answer is "that is not possible", ask which architectural controls support that claim and how they were tested. Draw it yourself during the proof of concept.

11. Cost

Cost belongs in a security evaluation for a reason that is not budgetary. In an agent product, spend is a function of behaviour — so an attacker who can influence behaviour can influence your bill, and your bill becomes a signal you can monitor. The other reason is that the licence is rarely the largest number: the people who keep the thing running usually are.

58. Is pricing per alert, per user, per unit of AI usage, or flat? If usage-based, is it billed to your account or mine?

Whose account the usage lands in determines who absorbs a surprise. It also determines who has an incentive to make the agent efficient, which quietly shapes how the product behaves over the life of the contract.

59. What does it cost to triage one alert — typically, and on the expensive ones?

Ask for the middle of the range and the worst five per cent, because the distribution has a long tail. A handful of pathological alerts can cost more than a normal week, and those are exactly the alerts an attacker would choose to generate.

60. Who pays when the agent gets stuck in a loop and makes four hundred tool calls on one alert?

Agents loop. They retry, re-reason and re-enrich, and a single confusing alert can cost more to triage than an ordinary week. Find out whose account is billed for that and whether there is a ceiling, because an attacker who works out how to generate expensive alerts has found a way to run up your bill or exhaust your quota without touching a single system.

61. Is there a spending ceiling per alert or per day, and when it is hit does it stop, degrade, or keep spending?

All three are defensible answers and they have very different consequences. Stopping is safe and creates a gap; degrading is sensible if you are told it is happening; continuing is fine until the month you find out otherwise. What you cannot accept is not knowing which one your product does.

62. Does the cost change when you change models?

Vendors move between models for quality and for margin. If your unit cost is a function of a choice they make without telling you, your budget is a variable they control. Ask for notice and for the right to stay on a known configuration.

63. Can I see spend per case as live data I can alert on, or only on the monthly invoice?

This is the security question hiding in the cost section. A sudden change in what the agent spends per case is one of the earliest signals that it is behaving differently — looping, being manipulated, or working much harder on a category of alert than it used to. Month-end billing tells you about it five weeks late.

64. Who on my team operates this after you have gone — integrations, approvals, exceptions, and investigating the agent itself?

Someone maintains the connectors when an API changes. Someone approves the actions the policy does not cover, handles the exceptions, tunes the suppressions, and works out what happened when the agent behaves oddly. Ask the vendor for the staffing their existing customers actually use, in hours per week rather than in adjectives, and ask whether that person needs to be your best analyst — because if it does, you have moved your scarcest resource rather than freed it.

12. Change management

65. Which model version is running, and what happens when you change it?

A floating model name means the component making decisions in your environment changes on someone else's schedule. Ask whether you can pin a version, how long old versions are supported, and whether you are told before a change or after.

66. How often do you ship updates, and can the agent's behaviour change without telling us?

There is a real difference between a release that adds a feature and a release that changes judgement. Ask whether release notes distinguish the two, and whether behaviour changes can arrive outside the release cycle — which they can, whenever the model behind the product is updated.

67. When you ship an update, can it reset or override tuning my team did?

Suppressions, exceptions, auto-close rules, asset classifications. Teams invest months in this and it is rarely versioned. Ask what is preserved across updates, what is not, and whether you get a diff of what changed.

68. Do you re-run a benchmark after each update and publish the difference? Can I stay on the previous version, and has anyone ever rolled back?

This is the question where the risk is hardest to see, because a bad update rarely announces itself. It might close more of what it should escalate, and everyone congratulates the tool on the quieter queue. It might escalate more of what it used to close, and your analysts absorb the difference without anyone recording why. Or it might keep the same volumes and make different wrong decisions inside them, which is the version nobody catches at all. None of those three produce an alert. A vendor who re-runs a regression set against the same cases after every update, and publishes the delta, has built the control most likely to catch any of them.

13. Exit

69. What do I keep if I leave — detection logic, evidence, case history?

Ask for the export format, the completeness and the timeline. Anything generated in your environment from your data has a reasonable claim to being yours, and the moment to establish that is before you sign rather than during the thirty-day wind-down.

70. What happens to my data — how long you keep it, how it is deleted, who your subprocessors are, where it is processed, and whether any of it trains your models?

Training is the question everyone asks and only one part of the answer. Ask what "data" covers — alerts, case content, analyst decisions, corrections — and whether opting out of training changes the product's behaviour or its price. Then ask the lifecycle questions that matter just as much: the retention period, what deletion actually means and how it is confirmed, which subprocessors touch the data, and which countries it is processed in. Finish with what happens to anything already learned when you leave, which is the part most agreements are quiet about.

Four things to ask for, not ask about

Use artifacts to verify the answers. Agree with the vendor which records, configurations and demonstrations will establish that the product meets your requirements.

The policy that governs what the agent may do — enforced outside the model, inspectable, and versioned. Ask who can edit it, ask for a diff from last month, and ask them to demonstrate the enforcement rather than hand you the document. Several architectures can satisfy this; what you are checking is that the constraint exists somewhere you can see and that it holds when tested.

One closed case exported exactly as it would arrive in your logging platform, including every tool call. Not a screenshot, not a PDF summary. The record as your investigators would receive it.

A run against a representative set of your own alerts, showing every place the product disagreed with your analysts. The disagreements are where the exercise earns its keep; the agreements mostly tell you the product handles what your team already handles. How to build that set is below.

One alert with hidden instructions planted in the subject line, run in their test environment while you watch, with their knowledge. Agree the test scope and observe which controls detect or block the attempted instruction.

Designing the proof of concept

Five hundred alerts is a number, not a method. What decides whether the exercise is worth anything is that the set is representative and that both sides agree the success criteria before the first alert is processed. Afterwards, every result is negotiable.

Build the set from five kinds of case, not one.

- Alerts you confirmed malicious, where the outcome is known.
- Alerts you confirmed benign, including the noisy recurring ones that make up most of your volume.
- Ambiguous cases where your own analysts disagreed. These are the most informative and the most commonly left out, because they are the hardest to score.
- Cases with incomplete telemetry — a missing log source, an integration that was down, an asset nothing in your inventory recognises.
- Cases where the human response was slow: an approval that sat overnight, a handover mid-incident.

Then test the things a replay of good days will never show you.

- Prompt injection, in a controlled environment, with the vendor's knowledge.
- A containment decision that turns out to be wrong — what the rollback is, how long recovery takes, and what breaks for the business while it happens.

- A model or configuration change, followed by a rerun of the same set, to see whether the product's judgement moved and whether anyone would have noticed.

Measure two different things, and do not let them be reported as one. Alerts the product saw and wrongly closed, and malicious activity that never reached it. Re-reviewing closed alerts measures the first. The second has to be tested against known attacker behaviour — a purple-team exercise, or a replay of techniques you know work in your environment — because it is by definition absent from the queue the product has already filtered.

Agree in writing, before you start: what counts as a correct close, who adjudicates a disagreement, what failure rate ends the trial, who pays for the compute, and what happens to your data when it is over.

Recording the answers

An evaluation falls apart at the point where four people remember four different versions of what the vendor said. Keep one row per question, in one place, filled in during the meeting rather than afterwards.

Field	What goes in it
Question	The number and the question, so two evaluations can be compared
Vendor response	What they actually said, in their words, not your summary of it
Evidence	The artifact, document, export or demonstration they provided — or "none offered"
Test result	One of the four outcomes below — whether it was shown
Acceptable to us	Yes, no, or not yet decided — whether what was shown is good enough
Gap	What is missing, and what would close it
Owner	The named person on your side who resolves it, and by when

The four outcomes, and what separates them

- **Demonstrated.** You saw it work, on your data or in a test you controlled. A verbal answer is not a demonstration, however confident.
- **Partially demonstrated.** The capability exists but you saw it in their environment, on their data, or with a caveat that has not been tested — for example a control that works today but is not enforced by permissions.
- **Not demonstrated.** Asked and not shown. This includes "we can do that" with nothing behind it, and it includes questions the vendor deferred and never came back to.
- **Not applicable.** The question does not apply to this product type. **Record the reason and the date**, as set out at the start. If you mark a critical row not applicable, judge that decision acceptable explicitly — otherwise the gate below cannot close either way.

Keep "shown" and "acceptable" apart. They are different findings and a worksheet that conflates them will mislead you. A vendor can produce a liability clause instantly, in writing, unambiguous — fully demonstrated — and the clause can place every consequence on you. A vendor can demonstrate a real off-switch that takes forty minutes to propagate. In both cases the evidence is excellent and the answer is not acceptable. Record the demonstration and the judgement in separate columns, and let a critical row pass only when both are satisfied.

Mark a handful of rows critical, and keep them out of the average. Before the evaluation starts, agree which controls you are not willing to go without. Mine are the same six named earlier: questions 1, 7, 8, 10, 15 and 22 — getting evidence out, the permission scope, the off-switch, what enforces the boundary, measuring what the product wrongly closed, and the liability position. Those rows produce a pass or a fail, and are excluded from any score. Everything else can be weighted and totalled however your procurement process prefers. Without that separation, a product with an unacceptable permission boundary wins on convenience features, and the scorecard will have provided the justification.

A worksheet with all seventy rows, the priority flags, the outcome lists and the critical-control gate already built accompanies this piece: **AI SOC Vendor Evaluation Worksheet v1.9**, free to download. One workbook per vendor — the summary reads the evaluation tab in the same file, so copying tabs inside one workbook will quietly leave every summary reading the first vendor's answers.

What to do with the answers

Write them down, and put them in the security exhibit of the contract rather than in a spreadsheet that expires with the evaluation. Work with procurement and legal to turn material assurances into clear obligations, with agreed scope, evidence and remedies.

Use these questions to establish the product's value, the controls it needs and the gaps your organization would accept. Record those decisions, assign owners and confirm how your team will observe, bound, investigate, contain and reverse its actions. The decision is whether the operational value justifies the residual risk, and whether your team can manage that risk.

Sources

Sources 1–5 checked 26 September 2026; source 6 added and read 27 September 2026. Every dated fact, figure and named incident above is sourced here. The list is grouped by publisher rather than by order of citation, so the first marker you meet in the text is [5]. The rest of the piece is either a question or a judgement formed from my own practice — where you read a generalisation about how products, vendors, markets or contracts usually behave, treat it as what I have seen rather than as survey data, and test it against your own. Where a detail comes from the researchers who reported an incident rather than from the record cited, the text says so.

1. Microsoft, *Access tokens in the Microsoft identity platform* — default access-token lifetime assigned as a random value between 60 and 90 minutes, and two hours for some clients in tenants that do not use Conditional Access. <https://learn.microsoft.com/en-us/entra/identity-platform/access-tokens>
2. Microsoft, *Continuous access evaluation* — a deleted or disabled account is one of the critical events evaluated; the goal is near real-time response, with latency of up to fifteen minutes possible because of event propagation. <https://learn.microsoft.com/en-us/entra/identity/conditional-access/concept-continuous-access-evaluation>
3. Postmark, *Information regarding the malicious postmark-mcp package* — the package impersonated Postmark, introduced a backdoor at version 1.0.16 that BCC'd handled email to an external server, and Postmark states it had not published its own MCP server on npm before the incident. <https://postmarkapp.com/blog/information-regarding-malicious-postmark-mcp-package>
4. NVD, *CVE-2025-32711* — "Ai command injection in M365 Copilot allows an unauthorized attacker to disclose information over a network." Published 11 June 2025. CVSS 3.1 base score 7.5 (NIST) and 9.3 (Microsoft as CNA), CWE-74. The record carries the classification and the scores; the name and the mechanism come from source 6. <https://nvd.nist.gov/vuln/detail/CVE-2025-32711>
5. Jessen Kurien, *The Defender's Guide to AI Agents*, Chapter 5 — the six events to log first, the sixteen event families, the correlation spine, and what each platform actually emits. The sixteen families are a working taxonomy drawn from the platforms surveyed there, not a claim to cover every event any agent estate can produce. <https://www.jessenkurien.com/ai-agent-logging/>
6. Pavan Reddy and Aditya Sanjay Gujral, *EchoLeak: The First Real-World Zero-Click Prompt Injection Exploit in a Production LLM System*, arXiv:2509.10540v1, 6 September 2025 — records that researchers at Aim Security disclosed and named EchoLeak in June 2025, and describes the exfiltration step: "As soon as Copilot returns its answer, the client interface (e.g., Outlook or Teams) automatically fetches the external image URL included by the attacker — achieving data exfiltration without any user clicks." <https://arxiv.org/abs/2509.10540>

The recommendations about contract language — the liability clause, and attaching answers to the security exhibit — are how I would approach it as a practitioner. They are not legal advice. Have your own procurement and legal teams review anything you intend to rely on.

About the author

Jessen Kurien is a cybersecurity leader, practitioner and author. Jessen brings 18+ years of cybersecurity experience, including nearly 15 years at Microsoft in leadership and technical roles, SOC leadership at a global MSSP, and a consulting engagement supporting Cisco. At Microsoft, he worked predominantly in security operations. He was part of the founding team of the Microsoft Threat Intelligence Center (MSTIC), where his work included APT and nation-state investigations. He contributed early detections to Microsoft Sentinel and later led detection engineering in Microsoft Defender XDR. He holds CISM and CISA, and is a Certified ISO/IEC 42001:2023 Lead Implementer & Lead Auditor (Artificial Intelligence Management System).

This guide can be used independently.

If you want to go further, *The Defender's Guide to AI Agents* covers defending agents once they are running: the surfaces an agent exposes, the stages of an agent attack, what to log, what to detect, and a ninety-day plan. It is free and open access, at <https://www.jessenkurien.com/defenders-guide-ai-agents/>

Reuse permissions

Copyright © 2026 Jessen Kurien. All rights reserved except as expressly permitted below.

You may download and read this guide, share links to it, and circulate unchanged copies within your organization, provided the author attribution and this notice remain intact.

Public republication, distribution of copies outside your organization, modified editions and resale require prior written permission from Jessen Kurien, except where applicable law permits the use. These terms do not limit fair use or other applicable copyright exceptions.

The separate editable Excel worksheet, including the questions reproduced in that file, is licensed under Creative Commons Attribution 4.0 International (CC BY 4.0). That licence does not apply to this guide as a whole.

Permissions: <https://www.jessenkurien.com/contact/>

Worksheet — print this page

The ten questions for a first meeting. Fill it in during the conversation, not afterwards. Shaded rows are critical controls — they pass or fail. The full seventy-row version is the companion Excel worksheet.

Vendor _____ Date _____ Present _____

Q	Question	What they said, and what they showed	Shown?	Acceptable?
1	Which events do you record, and can I get them into my own log system?			
7	What account does it use, what can it reach, and who at your company can use it?			
8	Can I switch it off in one action, and how long before it stops working?			
10	What actually enforces the rules between the model deciding and the action?			
15	How often does it wrongly close a real incident, and how did you measure that?			
16	What can it do if an attacker plants instructions in an alert it reads?			
25	Can I run the evaluation myself, in my environment, on my data?			
30	Does it detect anything itself, or act on alerts my tools already generated?			
58	Is pricing per alert, per user, per unit of AI usage, or flat? Whose account is billed?			
69	What do I keep if I leave — detection logic, evidence, case history?			

Shown: demonstrated / partial / not shown / n-a. **Acceptable:** yes / no / undecided. Different findings — a vendor can evidence something perfectly and the answer still be one you refuse. The sixth critical control, question 22 on liability, sits with procurement rather than the first meeting.

Version history

Version	What changed
1.0	First complete draft: seventy questions in thirteen sections, each with what the question is really testing, plus three figures.
1.1	Added the scope section defining four product types and the autonomy ladder; the ten-question starting set and the three evaluation passes; the decision record; the proof-of-concept design; sources for every dated claim. Five coverage gaps folded into existing questions without adding any.
1.2	Worksheet reliability: the critical gate, completion and decision readiness separated into three indicators; an acceptance column distinct from evidence; question 15 made a critical control; one workbook per vendor. Question 15's two measures restated as explicit formulas.
1.3	The Critical column validated, closing a route by which a failed control could be hidden. Question 35 qualified. Page specification corrected to the current question numbering.
1.4	Question 15's two populations defined precisely and the reason for wanting both stated. Figure 3 aligned with the prose. Dates aligned; worksheet download linked.
1.5	Question 15's exposure measure reworded so a past misclassification is not described as a live incident; a duplicated request removed.
1.6	Retitled and released as a guide: a designed PDF with cover, contents, running heads, a print-ready worksheet page and this version history, alongside the web guide and the Excel worksheet. No change to the questions.
1.7	Positioned as a standalone guide. The cover no longer describes it as a companion to another book, and the closing says the guide can be used independently. The Defender's Guide remains a cited source and is offered as further reading.
1.8	Production pass on the PDF: page breaks reflowed so no page carries a stray fragment; the three figures and the worksheet moved to full sideways pages at readable size; clickable contents, bookmarks and live source links added.
1.9	Correction pass. The three figures now sit on genuine landscape pages with upright content instead of rotated portrait ones, and no heading is left stranded at a page foot. Question 1's logging taxonomy is qualified as a working set rather than a complete one, and the EchoLeak name and mechanism gain their own source. Two malformed formulas in the worksheet's decision-readiness cells corrected — they returned errors in Excel.
1.10–1.12	Editorial tone and author-introduction revisions. Microsoft, global MSSP and Cisco consulting context added. Questions and worksheet unchanged.

This guide is revised when a source is re-checked or a claim is corrected, not only when the text grows. The version and review date on the cover tell you which one you are holding.