

Answer Key — Two Scenario Walkthroughs

Admin / self-check reference

The point is not perfect alignment with the key. The point is that your reasoning uses the same categories: evidence model for red, volume-and-burst for blue, content class as a secondary signal in both.

Red scenario — mid-engagement triage

The captured NTLM hash. Do not paste it. A hash captured during an engagement is a credential derived from the client's environment, and pasting it into any AI turns a timestamped entry in a provider log into the "last known location" of that credential — outside the scope of the engagement's contract, outside any data-handling agreement, and permanently linked to whatever account submitted the prompt. The right workflow is one of two options: either describe the hash abstractly ("32-hex-character value, no salt, no prefix, format looks like X") and ask the AI to identify the type from the description, or run an offline tool like `hashid` / `hashcat --identify` and skip the AI entirely. Offline is cheaper and safer. Once the hash is in a provider log, it's discoverable — subpoena, breach, internal investigation, any of them. That cannot be un-done.

The partial Kerberos ticket. Absolutely never paste. A Kerberos ticket is not just a credential, it carries domain, realm, principal name, timestamps, and encryption metadata — every one of those is a client identifier in addition to being a credential. There is no version of "just to check" that makes this safe. If you need to understand the ticket's structure, use offline tooling (`klist` , `impacket` 's parsers, `ticketer`) against the binary locally. If you need to ask the AI how Kerberos tickets are structured, do so with a generic example from documentation — never with the real one.

The list of internal hostnames. Do not paste verbatim. Internal hostnames are high-signal client identifiers; even one line of `app-prod-01.corp.client-name.internal` identifies the engagement and the client simultaneously. The safe workflow is to abstract the topology instead: describe it as "three-tier environment: public web tier →

application tier → database tier, over internal DNS, with hostnames following a `{role}-{env}-{n}` convention" and ask the AI to reason about attack paths against that generic description. The abstraction is enough for the AI to be useful and not enough to identify the client.

The next-step plan. Safe to AI-assist if, and only if, every engagement-identifying context is stripped before the prompt is written. Write against a framing like "hypothetical target with three-tier topology, NTLMv2 hashes recovered, partial Kerberos material available, objective: lateral movement to the database tier" — no real names, no real hostnames, no real ticket fragments, no actual captured material. The AI's output will still be useful because the reasoning transfers from the hypothetical to the real environment in your head, not in the prompt.

If the personal account already has any engagement content. The account is, from that point forward, part of the engagement's evidence footprint. It is no longer a "personal" account for the purposes of this engagement — it has been contaminated. The correct response is the same as persona contamination in Lesson 05: stop using it for engagement work immediately, switch to a properly scoped engagement persona on an enterprise zero-retention tier, and treat the personal account as burned with respect to this engagement. Do not try to "clean" it by deleting the prompts — deletion in the UI does not mean deletion in the provider's logs, and the legal posture is already set the moment the paste happened.

Blue scenario — DLP telemetry triage

Most concerning: Engineer A. 12 file uploads in 30 minutes is a burst pattern, and burst patterns are the highest-value signal in this telemetry set. Humans doing normal development work upload files gradually across the day; a 12-file burst inside a 30-minute window is consistent with either intentional exfil or a panic-debug-after-incident pattern (someone has a problem, they are trying to shove the whole offending directory into an AI to get help fast). Both possibilities warrant investigation *now*, not tomorrow. Next actions: pull the file names and classes from DLP metadata, identify the source directory, check whether any file matches sensitive data classes (source code from restricted repos, customer data, credentials, configuration with secrets), and check whether the engineer has had an incident or production page in that window. Open a direct conversation with the engineer before HR or legal — the most likely explanation is still honest panic, and you want to confirm or rule it out before escalating.

Engineer B — moderate concern. 8 code blocks spread over 6 hours is within the envelope of normal developer usage of a coding assistant. The volume alone is not the story. The question is content class: were the pastes code from private repositories? Did they contain internal hostnames, API keys, or customer identifiers? Did they match source files the company considers sensitive IP? Pull the classes from DLP metadata, not the content itself. If the classes come back clean, this is a soft conversation — "hey, remember the approved-tools policy, here are the sanctioned options" — not a ticket-style escalation. If the classes surface private repo matches, escalate proportionally.

Engineer C — low concern on volume. One 40-line prompt is a normal pattern for asking a question. The volume does not justify further investigation by itself. Still check the content class from DLP metadata to rule out that the 40 lines happened to be the crown jewels. Almost always this is benign: a question about a library, a shell one-liner, a small reasoning prompt. Note the result in the triage log and move on.

General next step for all three: pull the content class from DLP metadata rather than the content itself. Classes — "source code from repo X", "customer data", "credentials", "configuration file", "generic text" — are enough to decide escalation without the defender team reading the actual content. Match the classes against the approved-tool policy, and escalate only on a policy violation or a sensitive class hit. This is both the right privacy posture for the engineers and the right investigative posture for the team: you are looking for signals that justify deeper examination, not running content-level surveillance on every prompt.

Escalation trigger. The line between a soft conversation and a formal incident is a policy violation involving a sensitive class, or a refusal to cooperate with the soft conversation. Volume burst plus sensitive class = formal incident. Sensitive class alone = formal conversation with a paper trail. Volume burst without sensitive class = direct conversation with the engineer, logged but not escalated. Neither = note and move on. Consistency matters more than severity: publish the trigger thresholds inside the security team so that every analyst applies the same rules.