Security Basics for Support Technicians

A Complete 10-Chapter Technical Support Course

Topics covered:

Recognizing phishing, vishing & smishing · social engineering beyond phishing

Verifying identity before acting · safe password resets & account recovery

Recognizing a real incident inside a routine ticket · handling a finding without destroying evidence

Confidentiality & data handling · session & workstation security

Capstone: three fresh tickets — phishing, account compromise, in-person social engineering

Exercises: 30 hands-on scenarios with worked solutions

Format: A4 · Dark-theme code examples

Philip Osztromok · Generated with Claude

Table of Contents

CHAPTER 1 OF 10

Security Is Everyone's First Job: What This Course Covers

Security Basics for Support Technicians

Chapter 1 · Security Is Everyone's First Job: What This Course Covers

Every course in this subject so far has assumed the problem in front of you is real — a disk genuinely failing, a service genuinely slow, a connection genuinely broken. This course covers a different category entirely: situations where the "problem" itself is a deliberate attempt to manipulate you, the support technician, into doing something you shouldn't. Attackers know that a busy person trying to be helpful is often an easier target than the system they're actually trying to reach.

What Every Prior Chapter in This Subject Quietly Assumed

Each Technical Support course so far has treated one specific thing as a given. This course exists because none of those things can always be assumed:

COURSE	WHAT IT QUIETLY ASSUMES
log1	The log entries in front of you are complete and trustworthy
netdiag1	The connectivity problem being reported is genuine, not staged
perfdiag1	"Slow" reflects a real resource constraint, not a distraction
appdiag1	A 5xx error is an application bug, not an attacker probing for one
incident1	Whoever filed the ticket is genuinely who they say they are
remote1	Whoever is asking for remote access is legitimately entitled to it

This course is specifically about the situations where that last assumption breaks down — where the "user" isn't who they claim to be, or the urgent request is deliberately shaped to make you skip a step you'd normally take.

Not the Same Ground as This Site's Security Courses

This site already has a Security subject — but its courses assume a very different reader, sitting in a very different seat:

EXISTING SECURITY COURSE	WHAT IT ACTUALLY TEACHES
owasp1	Writing application code that resists common web attacks
auth1	Designing a secure authentication and session system
xss1 / sqli1	Specific attack classes as they appear inside application code
crypto1	Cryptographic primitives and how to use them correctly
pentest1	Actively testing a system's own defenses from the outside

None of that requires writing or reading a line of code the way those courses do — and this course doesn't ask you to either. The skill here is entirely judgment, exercised in real time, usually over a phone call, a chat window, or a ticket queue, with someone on the other end actively trying to move faster than your judgment can keep up.

What This Course Actually Covers

Four areas, built out across the next eight chapters:

- **Recognizing manipulation aimed at you** (Chapters 2–3) — phishing across email, SMS, and voice, and social engineering tactics that don't involve a suspicious link at all
- **Verifying identity and handling resets safely** (Chapters 4–5) — confirming who you're really talking to before acting, and the specific, high-risk case of a password reset request
- **Recognizing a real incident and handling it without destroying evidence** (Chapters 6–7) — the moment a "routine" ticket turns out not to be routine, and what to do (and not do) in the minutes right after
- **Confidentiality and session hygiene** (Chapters 8–9) — protecting the sensitive data that passes through support work every day, and locking down your own workstation and sessions

> **THE ONE IDEA TO CARRY THROUGH THIS WHOLE COURSE**
> If a request feels unusually urgent, that urgency is very often the point, not a coincidence. Manufactured time pressure is one of the most reliable tools in a social engineer's own kit, specifically because it works against exactly the instinct that makes a good support technician good — the instinct to help quickly. Chapter 2 covers this directly.

A Request That Isn't Resolved Yet

A message lands in the support queue midafternoon: "*This is urgent — I'm locked out of my account and about to walk into a board meeting in five minutes. I can't verify by phone right now. Please just reset it and send the new password to this email.*"

Nothing about that message is resolved in this chapter — it's deliberately left open. Chapter 5 comes back to this exact request directly, once the identity-verification and password-reset material it depends on has actually been covered.

WHAT THIS COURSE WON'T GIVE YOU

A memorized script or your specific employer's own formal security policy — those differ by organization, and neither substitutes for the other. What this course builds instead is transferable judgment: the pattern-recognition to notice something's off, even the first time you see a specific version of it.

Hands-On Exercises

EXERCISE 1

Using this chapter's own reasoning, explain why support technicians are described as a "primary target" for social engineering rather than simply one target among many.

View solution

EXERCISE 2

Using the first comparison table, identify which specific assumption from an existing Technical Support course this new course exists to question, and explain why that assumption can't always be trusted.

View solution

EXERCISE 3

Explain why the urgent password-reset request in this chapter is left deliberately unresolved, and name the specific chapter that returns to it.

View solution

CHAPTER 1 QUICK REFERENCE

- This course covers **human-facing** security judgment for support work — no coding, distinct from `owasp1`/`auth1`/`xss1`/`sqli1`/`crypto1`/`pentest1`
- It exists to question one assumption every other Technical Support course quietly makes: that the person you're helping is genuinely who they claim to be
- Four areas ahead: recognizing manipulation, verifying identity/resets, recognizing and handling a real incident, confidentiality/session hygiene

- **Core idea:** manufactured urgency is a tool, not a coincidence — treat it as a signal, not a reason to skip a step
- Next: Chapter 2, recognizing phishing across email, SMS, and voice

CHAPTER 2 OF 10

Recognizing Phishing: Email, SMS & Voice

Security Basics for Support Technicians

Chapter 2 · Recognizing Phishing: Email, SMS & Voice

Phishing is the most common way an attacker reaches a support technician directly, and it now comes through three genuinely different channels — email, text message, and phone calls — each with its own specific tells. This chapter builds the concrete pattern-recognition Chapter 1's tip-box promised: manufactured urgency is the constant across all three, but the surface details differ enough that each channel deserves its own look.

Email Phishing

The classic channel, and still the most common. The red flags rarely show up one at a time — a genuine phishing email usually stacks several at once:

- **Sender domain mismatch** — a display name that looks legitimate ("IT Support Desk") hiding a domain that isn't
- **Urgency and consequence** — "within 24 hours," "your account will be suspended," "immediate action required"
- **A generic greeting** — "Dear User" instead of an actual name, since bulk phishing rarely has one
- **A link whose visible text doesn't match its real destination** — the single most reliable technical tell, and the easiest to check

```
-- Raw header, sender display name vs. actual domain --
From: "IT Support Desk" <it-support@paypa1-secure-verify.com>
To: helpdesk@company.com
Subject: URGENT: Your account will be suspended in 24 hours

-- Raw HTML source of the "reset your password" button --
<a href="http://mycompany-secure-login.ru/reset.php">
  https://mycompany.com/reset-password
</a>
```

The display text of that link reads exactly like the real password-reset page — that's deliberate. The actual destination, visible by hovering over the link (or reading its raw HTML source, as above) rather than clicking it, is a completely different domain. This gap between what a link *says* and where it actually *goes* is worth checking on every single link in a message that asks you to log in or reset anything.

SMS Phishing ("Smishing")

The same manipulation, compressed into a text message. Smishing relies on the format itself working in the attacker's favor — a text message has no visible "From" domain to inspect, sender IDs are trivially spoofed, and shortened links (`bit.ly/...`, `t.co/...`) hide their real destination by design, not just by accident.

- "Your package couldn't be delivered — confirm your address: [link]"
- "Unusual sign-in detected. Verify your identity now or your account will be locked: [link]"
- A sender ID that looks like a short internal extension or an official-looking company name, spoofable with no special tools

> **WHY SHORTENED LINKS ARE WORSE HERE, NOT JUST INCONVENIENT**
>
> On email, a suspicious full URL is at least visible before you click. A shortened link gives you nothing to inspect at all until you've already followed it — treat any shortened link in an unsolicited text as unverifiable by default, not just "worth a second look."

Voice Phishing ("Vishing")

The channel most likely to reach a support technician directly, and the hardest to fake-detect on instinct alone, because caller ID can be spoofed just as easily as an email's display name. A vishing call typically impersonates IT, a vendor, or an executive, and leans even harder on urgency than email or SMS, since a live conversation gives the caller room to keep pushing past hesitation in real time.

The specific attack worth knowing by name is the **one-time-code relay**: the caller triggers a genuine password reset or MFA challenge on the real account first, then calls claiming to be verifying your identity, and asks you to "read back the code we just sent you to confirm it's really you."

> **THE CODE IS THE SECURITY CONTROL — NOT PROOF OF WHO YOU ARE**
>
> A one-time code exists specifically to prove that the person completing an action has access to the account. Reading it back to a caller hands them exactly that proof — you are not confirming your own identity to them, you are handing them the credential that confirms *their* access to the account. No legitimate verification process ever needs you to read a code back over the phone. Chapter 4 covers what legitimate identity verification actually looks like.

Red Flags That Appear Across All Three Channels

SIGNAL	WHY IT WORKS ON THE ATTACKER'S SIDE
Manufactured urgency	Pressures you to act before you'd normally stop to verify
A request for credentials or a one-time code	No legitimate process ever needs these read back or forwarded

SIGNAL	WHY IT WORKS ON THE ATTACKER'S SIDE
Branding that's almost, but not quite, right	Close enough to pass a glance, wrong enough to be checkable if you look
Pressure to skip your normal verification step	Directly targets the one safeguard that would otherwise catch the attempt

That last row connects directly back to Chapter 1's own framing: the request that asks you to skip verification "just this once, because it's urgent" isn't an edge case this course has to imagine — it's the single most common shape a real attack actually takes.

REPORT IT, DON'T JUST DELETE IT

A suspicious email or text you simply delete disappears from your own inbox but tells your organization nothing. Reporting it (through whatever channel your organization provides) means the same message reaching a colleague can potentially be caught before they see it at all — turning one near-miss into protection for everyone else.

Hands-On Exercises

EXERCISE 1

Using the email example in this chapter, explain specifically why the link's visible text being different from its actual destination is described as "the single most reliable technical tell."

View solution

EXERCISE 2

Explain why the one-time-code relay attack tricks the victim into believing they're confirming their own identity, when they're actually doing the opposite.

View solution

EXERCISE 3

Explain why a shortened link in an unsolicited text message is treated as unverifiable by default, rather than just "worth a second look" the way a full URL in an email is.

View solution

CHAPTER 2 QUICK REFERENCE

- **Email:** check sender domain (not just display name) and the real destination of every link, not its visible text
- **SMS:** spoofable sender IDs, shortened links hide their destination by design — treat as unverifiable by default
- **Voice:** caller ID is spoofable; never read back a one-time code to anyone who calls asking for it — that code proves access, it doesn't prove identity
- Shared thread across all three: manufactured urgency, requests for credentials/codes, near-right branding, pressure to skip normal verification
- Report suspicious messages through your organization's own channel rather than only deleting them
- Next: Chapter 3, social engineering tactics that don't rely on a suspicious link or call at all

CHAPTER 3 OF 10

Social Engineering Beyond Phishing: Pretexting, Impersonation & Baiting

Security Basics for Support Technicians

Chapter 3 · Social Engineering Beyond Phishing: Pretexting, Impersonation & Baiting

Chapter 2 covered phishing's three channels — but every one of them relied on a message: an email, a text, a call. This chapter covers the tactics that don't need a message at all — a fabricated backstory, a face pretending to be someone it isn't, or a lure left somewhere it's likely to be found. None of these show up in a typical "spot the phishing email" training, which is exactly why they're worth their own chapter.

Pretexting: The Fabricated Backstory

Pretexting is an invented scenario, built specifically to make a request feel routine and legitimate before the actual ask arrives. The backstory does the persuading; the request itself is often small enough to seem reasonable on its own.

Example: a caller identifies themselves as the manager of a new hire starting Monday, explains the new hire's laptop hasn't arrived yet, and asks for "just a temporary login" so the new hire can start onboarding paperwork from a personal device today. Every individual detail — a manager, a new hire, a shipping delay — is plausible in isolation. The backstory's job is to make the actual request (creating access outside the normal process) feel like a minor administrative favor rather than what it actually is.

Impersonation: Claiming to Be Someone Specific

Where pretexting invents a scenario, impersonation claims a specific identity — a named coworker, a known vendor contact, an executive, or even IT itself calling about IT's own systems. The more specific and verifiable-sounding the claimed identity, the more it discourages a target from questioning it.

- **Executive impersonation** — a request that name-drops urgency and authority together ("this is [CFO name], I need this handled before my flight")
- **Vendor impersonation** — someone claiming to represent a company you already do business with, asking for account details "to process a routine update"
- **IT impersonation** — perhaps the most direct threat to a support technician specifically, since it targets the exact trust relationship coworkers have with the support desk itself

- **Physical impersonation (tailgating)** — wearing a delivery uniform, a contractor badge, or simply carrying a box and walking confidently through a badge-locked door behind someone who holds it open

That last one is a preview of Chapter 9's own physical-security material — worth flagging here because it's a form of impersonation, not a separate category of threat.

Baiting: The Enticing Lure

Baiting offers something the target wants — curiosity, a free item, an exclusive download — with a malicious payload attached to the offer. The classic version is physical: a USB drive labeled `Payroll_2026_Confidential.xlsx` left in a car park, breakroom, or elevator, counting on someone's curiosity to plug it into a work machine. The digital version is the same idea online — a "free" tool, plugin, or download that quietly installs something unwanted alongside it.

> **A FOUND USB DRIVE IS NOT A LOST-AND-FOUND ITEM**
> Never plug an unknown USB device into any work machine to "see who it belongs to." That instinct is precisely the mechanism the attack is built to exploit — hand it to your security or IT team unopened instead.

Quid Pro Quo: Something for Something

A close cousin of baiting, framed as an even trade rather than a free gift. An attacker calls around an organization claiming to be IT support offering a "free security upgrade," and asks whoever picks up to run a specific command or temporarily disable a specific protection in exchange. Most employees who answer won't take the offer — but the attacker only needs one who does.

Comparing the Four Tactics

TACTIC	CORE MECHANISM	WHAT MAKES IT WORK
Pretexting	An invented backstory precedes the actual request	The request feels routine by the time it arrives
Impersonation	Claiming a specific, real identity or role	Discourages the target from questioning someone they think they know
Baiting	An enticing lure carries the payload	Curiosity or self-interest overrides caution
Quid pro quo	An offer framed as a fair trade	Feels reciprocal rather than one-sided, lowering suspicion

A Request That's Left Open Again

A call comes in: "This is Facilities — we've got a delivery emergency and need the loading dock badge reader temporarily disabled for the next twenty minutes." It's a blend of pretexting (the delivery-emergency backstory) and impersonation (claiming to be Facilities). Whether it's legitimate or not can't be determined from the call alone — and that's exactly the point.

THE RULE THIS CHAPTER IS BUILDING TOWARD

If you can't verify a claim through a channel the caller themselves doesn't control — calling Facilities back at a known internal number, rather than trusting whatever number they called from — you can't verify it at all. Chapter 4 covers exactly what that kind of independent verification looks like in practice.

Hands-On Exercises

EXERCISE 1

Using the new-hire example, explain what job the fabricated backstory is doing, separate from the actual request being made.

View solution

EXERCISE 2

Explain why IT impersonation is described as "perhaps the most direct threat to a support technician specifically," using this chapter's own reasoning about trust.

View solution

EXERCISE 3

Explain why calling a claimed caller back at a known internal number is a meaningfully different verification step than trusting the number they called from, and why this chapter says a claim can't be verified any other way.

View solution

CHAPTER 3 QUICK REFERENCE

- **Pretexting:** a fabricated backstory makes the real request feel routine
- **Impersonation:** claiming a specific real identity, including physical tailgating through a badge-locked door
- **Baiting:** an enticing lure (a found USB drive, a "free" download) carries the payload
- **Quid pro quo:** a request framed as a fair trade, not a one-sided ask

- None of these four rely on a suspicious link, text, or call asking for a code — phishing training alone doesn't cover them
- A claim can only be verified through a channel the claimant doesn't control — never the contact info they themselves provided
- Next: Chapter 4, verifying identity before acting on a request

CHAPTER 4 OF 10

Verifying Identity Before Acting on a Request

Security Basics for Support Technicians

Chapter 4 · Verifying Identity Before Acting on a Request

Chapter 3 established the principle: a claim can only be verified through a channel the claimant doesn't control. This chapter turns that principle into a practical, repeatable process — and, just as important, one that scales with how risky the request actually is. Not every request needs the same amount of scrutiny, and treating a low-stakes question with the same suspicion as a password reset just trains people to route around your process rather than follow it.

Match Verification Effort to Risk, Not to Suspicion

Verification isn't a single on/off switch — it scales with what's actually being asked for:

RISK TIER	EXAMPLE REQUEST	APPROPRIATE VERIFICATION
Low	"How do I connect to the guest Wi-Fi?"	None needed — no sensitive access or data is at stake
Medium	"What's the status of my ticket #4821?"	A basic check — matching name/employee ID against the ticket itself
High	Password reset, access grant, disabling a security control	Independent callback, out-of-band confirmation, or manager sign-off — not skipped for any amount of urgency

The Chapter 1 reset request and Chapter 3's badge-reader call both sit firmly in the high tier — which is exactly why neither could be resolved on the spot in the chapters that introduced them.

Callback Verification, Done Properly

Chapter 3 established the core rule; here's what it looks like in practice. Never call back a number the requester provided — not the number they're calling from, not a number they read out over chat. Instead:

1. End or pause the current contact
2. Independently look up the person or department's number — an internal directory, an intranet page, a previously saved contact — never anything supplied in this specific conversation

3. Call that number and confirm the request directly

Applied to Chapter 3's badge-reader call: rather than acting on the caller's claim, or even calling back whatever number showed on caller ID, the technician looks up Facilities' own listed extension and calls that instead. If the person who actually answers has no idea what "delivery emergency" is being referred to, the original call is confirmed as fabricated — without ever having disabled anything.

Out-of-Band Confirmation

A close relative of callback verification, used when a phone call isn't the channel in question. If a request arrives over chat, confirm it through a different channel entirely — a known email address, an in-person check, or a separate messaging platform the requester doesn't control the identity behind. The principle is the same: never confirm a claim using only the same channel the claim arrived on, since that channel may itself be compromised or spoofed.

Why "Security Questions" Are Weaker Than They Feel

INFORMATION THE REQUESTER PROVIDES ABOUT THEMSELVES ISN'T PROOF

An employee ID, a date of birth, a "mother's maiden name" — these feel like verification because they're personal, but they're often guessable, previously leaked in an unrelated breach, or simply findable through the pretexting techniques Chapter 3 covered. None of it is proof of identity on its own; it only proves the requester knows a fact, which an attacker who's done their homework may also know.

Knowledge-based checks like these are fine as one layer among several for medium-risk requests — they should never be the *only* check standing in front of a high-risk one.

Worked Example: Applying the Tiers

A chat message arrives: "Hey, it's Dana from Accounting, can you tell me what my temp password is? I'm locked out and have a deadline in ten minutes." This is a high-risk request (a credential) wearing a low-risk tone (a casual, friendly chat message). The risk tier is set by *what's being asked for*, not by how the request is phrased — so it gets high-tier verification regardless of how routine or friendly it sounds: an out-of-band callback to a number looked up independently, not a reply in the same chat thread.

TONE IS NOT A RISK SIGNAL — THE REQUEST ITSELF IS

A casual, friendly, or even mildly impatient tone tells you nothing about whether a request is legitimate. Judge the risk tier from what's actually being requested, and let that — not how it's phrased — decide how much verification it gets.

Hands-On Exercises

EXERCISE 1

Explain why treating every request with the same level of scrutiny, regardless of risk tier, is described as counterproductive rather than simply "extra safe."

View solution

EXERCISE 2

Using the Dana from Accounting example, explain why the request is classified as high-risk even though its tone is casual and friendly.

View solution

EXERCISE 3

Explain why knowledge-based checks like "mother's maiden name" are described as proving only "that the requester knows a fact," not that they are who they claim to be.

View solution

CHAPTER 4 QUICK REFERENCE

- Verification effort scales with the **risk of the request**, not with how suspicious it feels
- Callback verification: always an independently-looked-up number, never one the requester supplied
- Out-of-band confirmation: confirm through a different channel than the one the request arrived on
- Knowledge-based checks (employee ID, birthdate) prove only that the requester knows a fact — not who they are
- Judge risk from **what's being asked for**, never from how the request is phrased or how urgent it sounds
- Next: Chapter 5, applying this directly to the Chapter 1 password-reset request

CHAPTER 5 OF 10

Safe Password Resets & Account Recovery

Security Basics for Support Technicians

Chapter 5 · Safe Password Resets & Account Recovery

A password reset is the single most common high-risk request a support desk handles, and the single most valuable one to an attacker — it grants direct account access, and because technicians handle dozens of legitimate resets every week, a malicious one is designed to look exactly like all the rest. This chapter applies Chapter 4's risk-tiered verification specifically to resets, and finally resolves the request left open since Chapter 1.

Why Resets Sit at the Top of the Risk Tier

Chapter 4's table placed password resets firmly in the high-risk tier alongside access grants and disabling security controls. A reset deserves that placement for a specific reason beyond "it's sensitive": once completed, it hands over full account access immediately, with no further step required — unlike, say, a leaked employee ID, which still needs to be combined with something else to be useful. A successful reset is the entire attack, done in one step.

The Safe Reset Procedure

1. **Verify first, every time** — full high-tier verification per Chapter 4, with no exception for urgency. Stated urgency is not a reason to skip a step; per Chapter 1, it's very often the reason the request exists at all.
2. **Prefer self-service recovery over technician-mediated resets** — a self-service flow tied to a previously registered device or authenticator app verifies identity through something the account owner already possesses, which no attacker on a cold call has access to.
3. **Never transmit a new password in plaintext** — not by email, not by chat, not read aloud over the phone. Use a forced reset-on-next-login instead, so no actual password value is ever something a technician hands over at all.
4. **Never send the new credential through the same channel the request arrived on** — the same out-of-band principle from Chapter 4, applied specifically to the response, not just the verification.
5. **Document who requested it, when, and how it was verified** — this record is exactly the evidence Chapter 7 needs if a reset later turns out to have been part of a real incident.

Resolving Chapter 1's Request

"This is urgent — I'm locked out and about to walk into a board meeting in five minutes. I can't verify by phone right now. Please just reset it and send the new password to this email."

Run through the procedure above: this is unambiguously high-risk (a reset), which means full verification, with no exception carved out for urgency. The stated inability to verify isn't a reason to bypass verification — following the reasoning from Chapters 1 through 4, it's the exact shape a real attack takes. The correct response isn't a flat refusal, though — it's holding the verification requirement while offering a path that doesn't require abandoning it: point the requester to the self-service recovery flow, which takes seconds and needs no phone call at all, or offer to verify with their manager directly while they're in transit. A genuine employee in a genuine hurry can complete either path in under a minute. A request that suddenly goes quiet once a real verification path is offered has told you everything you need to know.

THE POINT OF THIS WHOLE EXAMPLE

"Can't verify right now" and "won't verify at all" are different problems with different answers. This procedure is built to solve the first without ever accepting the second.

MFA & Account Recovery Deserve the Same Treatment

Requests to reset, remove, or "temporarily disable" multi-factor authentication are at least as high-risk as a password reset — arguably higher, since MFA exists specifically to catch an account compromise even after a password has already been stolen. Removing it doesn't just grant access; it removes the safeguard that would have caught the very attack in progress.

THERE IS NO SUCH THING AS A "TEMPORARY" MFA BYPASS

A temporary bypass is a window with no built-in end — it stays open until someone remembers to close it, and an attacker who obtained the bypass has no reason to wait. Treat "just disable it for now, I'll turn it back on later" with the same full verification a permanent change would get, never less.

Unsafe vs. Safe, Side by Side

SITUATION	UNSAFE	SAFE
Delivering a new credential	Emailing the new password directly	Forcing a reset on next login
Urgent, can't-verify request	Skipping verification "just this once"	Offering a fast, legitimate self-service path instead
MFA reset request	Disabling MFA "temporarily" as a favor	Full high-tier verification, same as a permanent change

SITUATION	UNSAFE	SAFE
Confirming the requester's identity	Accepting a security-question answer alone	Independent callback or out-of-band confirmation (Chapter 4)

Hands-On Exercises

EXERCISE 1

Explain why a password reset is described as "the entire attack, done in one step," in contrast to a leaked employee ID.

View solution

EXERCISE 2

Explain the distinction this chapter draws between "can't verify right now" and "won't verify at all," and why offering a self-service path is the way to tell them apart.

View solution

EXERCISE 3

Explain why a "temporary" MFA bypass is treated as no safer than a permanent one, using this chapter's own reasoning about who controls when it actually ends.

View solution

CHAPTER 5 QUICK REFERENCE

- Password resets and MFA changes are always high-risk — full Chapter 4 verification, no urgency exception
- Never transmit a plaintext password by email/chat/phone — force a reset on next login instead
- Never deliver a new credential through the same channel the request arrived on
- "Can't verify right now" ≠ a reason to skip verification — offer a genuine fast path (self-service, manager confirmation) instead of a flat refusal
- There's no such thing as a safe "temporary" MFA bypass
- Document who requested a reset and how it was verified — this record matters directly in Chapter 7
- Next: Chapter 6, recognizing when a "routine" ticket is actually a security incident

CHAPTER 6 OF 10

Recognizing When a "Routine" Ticket Is Actually a Security Incident

Security Basics for Support Technicians

Chapter 6 · Recognizing When a "Routine" Ticket Is Actually a Security Incident

Chapters 2 through 5 were about stopping an attack before it succeeds. This chapter is about what happens after one already might have — recognizing the moment an ordinary-looking ticket is actually an early sign that something has already gone wrong, not a normal request to work through as usual.

Where This Sits Relative to `incident1`

This site's own Incident Response & Ticketing Workflows course teaches classification, severity, and escalation — but all of that starts from a ticket already recognized as a security-relevant incident. This chapter covers the step before that: noticing that a given ticket even belongs in that category in the first place. `incident1`'s own material picks up exactly where this chapter leaves off.

Symptoms Worth a Second Look

- **"I'm getting weird emails from myself"** — a classic sign of a spoofed or compromised mailbox, not a glitch worth dismissing
- **Unexpected password-reset emails the user didn't request** — someone else may already be attempting to take over the account
- **An unfamiliar sign-in notification** — "new device" or "new location" alerts the user doesn't recognize
- **Several users reporting similar odd behavior around the same time** — a pattern, not an isolated glitch, and worth treating differently from a single one-off report
- **An unfamiliar file, process, or browser extension the user didn't install** — a possible sign of malware, especially following Chapter 3's baiting material
- **Unusual slowness paired with unusual network activity** — often has a perfectly mundane explanation per `perfdiag1`/`netdiag1`, but worth asking the question rather than assuming the boring explanation automatically

The One Question That Actually Matters

Not every odd symptom is an attack — most aren't. But a genuine security lens means actually asking, for each one: **could this innocuous-looking symptom also be an early sign of compromise?** — rather than reaching for the most boring available explanation by reflex, every single time.

The Same Symptom, Read Two Ways

SYMPTOM AS REPORTED	ROUTINE READ	SECURITY-LENS READ
"Email isn't syncing right on my phone"	A sync bug, restart the mail app	Check for unfamiliar forwarding rules quietly added to the account
"My computer's been slow all week"	Needs a reboot or a disk check (`perfdiag1`)	Also worth a quick look for an unfamiliar running process
"I got a password reset email I didn't ask for"	Probably a fluke, dismiss it	A possible sign someone else is actively trying to take over the account

Both Failure Modes Have a Real Cost

Treating every odd symptom as a full security incident produces the same problem Chapter 4 warned against for over-verification: alert fatigue, wasted investigation time, and a support process people learn to route around. But never asking the question at all means genuine early warning signs quietly pass by unnoticed until the damage is much larger. The goal isn't maximum suspicion — it's asking the one question above, consistently, and only escalating when the answer to it is genuinely "yes, this could be."

Worked Example: The Sync Bug That Wasn't

A user reports their email "isn't syncing right on their phone" — missing some messages, sent items appearing they don't remember sending. The routine read is a sync glitch, worth a quick app restart. Reading it through the lens above instead, a check of the account's own mail rules turns up a forwarding rule the user never created, quietly copying every incoming message to an external address. This is a well-known pattern in real account-compromise cases: once an attacker gains access, a forwarding rule lets them keep reading the mailbox even after the original login is noticed and the password changed.

> **DON'T FIX IT YOURSELF IN THE MOMENT**
>
> Once something like this turns up, the instinct is to just delete the rule and consider the ticket closed. Resist it — deleting the evidence before it's documented destroys exactly what an actual investigation needs. Chapter 7 covers what to do (and not do) in the minutes right after a discovery like this one.

Hands-On Exercises

EXERCISE 1

Explain how this chapter's own scope differs from `incident1`'s, and why the sync-bug example belongs to this chapter rather than that course.

View solution

EXERCISE 2

Explain why treating every odd symptom as a potential security incident is described as having a real cost, not just as being "extra careful."

View solution

EXERCISE 3

Explain why an unexpected mail-forwarding rule is a well-known pattern in real account-compromise cases, specifically in terms of what it lets an attacker keep doing even after the original login is discovered.

View solution

CHAPTER 6 QUICK REFERENCE

- This chapter is about **recognizing** a potential incident; `incident1` covers what happens once it's recognized
- Watch for: unfamiliar sign-in alerts, unrequested reset emails, unfamiliar forwarding rules, unexplained files/processes, similar reports across multiple users
- The core question: could this innocuous symptom also be an early sign of compromise? — ask it, don't reflexively assume the boring explanation
- Both over-flagging and under-flagging have real costs — alert fatigue on one side, missed compromise on the other
- Never quietly fix a suspicious finding yourself — document it first (Chapter 7)
- Next: Chapter 7, handling a suspected compromise without destroying the evidence

CHAPTER 7 OF 10

Handling a Suspected Compromise Without Destroying Evidence

Security Basics for Support Technicians

Chapter 7 · Handling a Suspected Compromise Without Destroying Evidence

Chapter 6 ended with a warning: don't quietly fix a suspicious finding the moment you spot it. This chapter covers exactly what to do instead — the same "evidence over guesswork" discipline `log1` built this entire subject around, and the same "capture before mitigate" sequencing `incident1`'s own Chapter 6 taught, both applied here to the specific moment a support technician finds a possible compromise.

Evidence First Isn't Optional — It's the Whole Point

Per `log1`'s own foundational argument, real evidence beats guesswork every time. Applied here: the very first thing that happens after a suspicious discovery determines whether anyone investigating it later has anything real to work from, or only someone's memory of what they think they saw.

The Capture-vs-Mitigate Tension, Resolved

`incident1`'s own Chapter 6 named this tension directly for general incidents: fixing something immediately can destroy the exact evidence needed to understand it, but leaving something broken indefinitely just to preserve evidence isn't right either, especially when it's actively causing harm. A live forwarding rule quietly leaking mail is exactly this case — every minute it stays in place is another minute of exposure.

STEP 1 — CAPTURE (MINUTES, NOT HOURS)

Document exactly what was found: a screenshot of the configuration, the precise rule/setting/file in question, when it appears to have been created if that's visible, and the time you found it. This step should take minutes, not become its own investigation.

STEP 2 — MITIGATE THE ACTIVE HARM

Once the finding is documented, stop the ongoing exposure — disable the forwarding rule, revoke the suspicious session, whatever specifically is causing continued harm right now. The order matters:

mitigating before capturing risks losing the exact evidence that explains what happened; delaying mitigation for hours "to be thorough" risks letting real harm continue for no good reason.

STEP 3 — ESCALATE WITH WHAT YOU'VE CAPTURED

Hand off the documented finding through your organization's own incident process — this is exactly the point where `incident1`'s own classification and escalation material takes over.

What Not to Do

INSTINCT	WHY IT'S A PROBLEM
Delete logs or suspicious files immediately	Destroys the exact evidence an investigation needs, per `log1`'s own founding warning
Reset the account without saving what was found	Closes the door without anyone ever learning what was behind it
Quietly clean it up and close the ticket	Hides the finding from anyone who could confirm whether other accounts are affected too
Confront who you suspect is responsible directly	Risks tipping them off to cover their tracks before anything is confirmed

Worked Example: Resolving Chapter 6's Forwarding Rule

Returning to the ticket from Chapter 6 — the mail-forwarding rule quietly copying every message to an external address. Applying the sequence above: the technician screenshots the rule's exact configuration and creation timestamp, notes exactly when and how it was found, then disables the rule within minutes to stop the ongoing leak, and immediately escalates the documented finding through the organization's incident process rather than continuing to investigate alone.

PAST THIS POINT, IT ISN'T A ONE-PERSON JOB ANYMORE

Once a finding is documented and the immediate harm is stopped, continuing to investigate solo — trying to figure out the full scope, who else might be affected, or who's responsible — isn't thoroughness, it's a decision that should belong to your organization's own incident process, not to one technician working alone. `incident1`'s own material on escalation applies directly here: asking for help at this point isn't a failure, it's the correct next step.

Hands-On Exercises

EXERCISE 1

Explain why this chapter puts capture before mitigate, rather than the other way around, and why the capture step should still be measured in minutes, not hours.

View solution

EXERCISE 2

Explain why quietly cleaning up a suspicious finding and closing the ticket is listed as a problem, even when the immediate technical issue is genuinely resolved.

View solution

EXERCISE 3

Explain why confronting a suspected internal wrongdoer directly is treated as a mistake, using this chapter's own reasoning.

View solution

CHAPTER 7 QUICK REFERENCE

- **Capture first** (minutes) — screenshot/document exactly what was found, then **mitigate** the active harm, then **escalate** with the evidence in hand
- Never delete logs, suspicious files, or configuration before documenting them
- Never reset an account or "clean up" a finding without saving what was there first
- Never confront a suspected internal wrongdoer directly — it risks tipping them off
- Past initial capture-and-mitigate, this stops being a one-person job — escalate through `incident1`'s own process
- Next: Chapter 8, confidentiality and data handling in support work

CHAPTER 8 OF 10

Confidentiality & Data Handling in Support Work

Security Basics for Support Technicians

Chapter 8 · Confidentiality & Data Handling in Support Work

Chapters 6 and 7 covered what to do once something is actually wrong. This chapter is about something quieter and far more constant — the sensitive data support work touches on *every single ticket*, incident or not. A support technician routinely sees more personal and account data than almost any other role in an organization, simply as a byproduct of doing the job well.

Least Exposure, Not Least Effort

`remote1`'s own Chapter 6 established least privilege for *access* — only reaching for the permissions a task actually needs. The same principle applies to *data*: only look at, record, or copy what's actually needed to resolve the ticket in front of you. During a remote session, that means resisting the pull to browse unrelated files or folders "just because you can see them" — the fact that access is technically available doesn't make looking at it appropriate.

Ticket Notes: Write What's Needed, Not What's Available

Ticket systems are often readable by a wider audience than the sensitive data pasted into them warrants — other technicians, reporting tools, sometimes exported logs. Never paste a password, a full card number, or an identification number into a ticket note, even "just temporarily" to reference later.

WHAT HAPPENED	UNSAFE NOTE	SAFE NOTE
Identity was verified for a reset	"Confirmed DOB 04/12 and mother's maiden name Smith"	"Identity verified via independent callback per Chapter 4 process"
A password was involved in troubleshooting	Pasting the actual password value into the ticket	Noting only that a reset was performed, never the value itself

The safe version records exactly what a later reader needs — that verification happened, and how — without also recording the sensitive material itself.

Screen Sharing: What You See Isn't Yours to Keep

`remote1`'s own Chapter 5 covered the trust model behind screen sharing; this chapter covers what happens when that shared screen shows something unrelated and sensitive — another open tab with a coworker's HR record, a personal message, anything outside the scope of the actual ticket. Don't linger on it, don't comment on it, and don't note it anywhere unless it's genuinely relevant to a real security concern. Seeing something incidentally isn't a license to act on it or discuss it.

Data Minimization When Pulling Records

When troubleshooting requires querying or exporting data, pull only what the specific ticket needs — a single account's records, not a full table export, even if a broader export would be more convenient to work from. This is the same underlying principle `dbsec1` teaches from the database-administration side, applied here to the everyday habits of support work rather than to system design.

Sending Sensitive Data Safely

Chapter 5 already established never sending a password in plaintext. The same rule extends to any sensitive data: never over an unencrypted channel, and never to a personal email address, no matter how much faster it would be for the person asking. If a legitimate business reason exists to transfer sensitive data, it should go through a channel your organization has actually approved for that purpose — not whatever's most convenient in the moment.

CURIOSITY IS A REAL CONFIDENTIALITY VIOLATION, NOT A HARMLESS PEEK

Looking up a colleague's ticket history, a public figure's account details, or an ex-partner's records out of pure curiosity — with no malicious intent and no plan to act on it — is still a genuine violation, and a surprisingly common one in real incidents. Many real confidentiality breaches involve no external attacker at all, just someone with legitimate access who looked at something they had no work reason to look at.

Hands-On Exercises

EXERCISE 1

Using the ticket-notes comparison table, explain what the safe version records and what it deliberately omits, and why that specific omission matters.

View solution

EXERCISE 2

Explain why this chapter treats "least exposure" for data as the same underlying principle as `remote1`'s own least privilege for access, rather than as a separate, unrelated rule.

View solution

EXERCISE 3

Explain why curiosity-driven access with no malicious intent is still described as a genuine confidentiality violation, rather than a harmless exception.

View solution

CHAPTER 8 QUICK REFERENCE

- **Least exposure:** only view/record/export what the ticket actually needs — technical access to more isn't a reason to look at more
- Never paste passwords, full card numbers, or ID numbers into ticket notes — record that verification happened, not the sensitive material itself
- Something incidentally visible during screen sharing isn't yours to comment on, linger over, or note down
- Pull the minimum data needed for troubleshooting, not the most convenient export
- Sensitive data travels only through approved, encrypted channels — never personal email, regardless of convenience
- Curiosity-driven access is a real violation — most confidentiality breaches involve legitimate access misused, not an outside attacker
- Next: Chapter 9, session and workstation security for support technicians

CHAPTER 9 OF 10

Session & Workstation Security for Support Technicians

Security Basics for Support Technicians

Chapter 9 · Session & Workstation Security for Support Technicians

Every chapter so far has focused on requests, tickets, and data flowing *through* a support technician. This chapter turns to something more basic and easy to overlook: the technician's own workstation. It's not just one more employee's device — it's typically a hub with a reach far wider than any single account, and that reach is exactly why it deserves its own chapter.

Why This Workstation Specifically Matters More

A compromised regular employee workstation typically exposes that one person's own accounts and data. A compromised support workstation is different in kind: it often has active sessions into ticketing systems, remote-access tools (`remote1`'s own territory), and sometimes elevated or admin-capable credentials — reaching many other accounts and systems at once. Compromising it doesn't just expose one identity; it hands an attacker the exact tools this course has spent eight chapters teaching how to recognize being misused against someone else.

	REGULAR EMPLOYEE WORKSTATION	SUPPORT TECHNICIAN WORKSTATION
Typical reach if compromised	That one person's own accounts and files	Many other users' accounts, via ticketing and remote-access tools
Sessions typically open	Personal email, a handful of work apps	Ticketing system, remote sessions, sometimes elevated credentials
Consequence of a walk-away compromise	Access to one inbox or account	A pivot point into every system that workstation can reach

The Habit That Matters Most: Locking the Screen, Every Time

Chapter 3 covered physical impersonation — someone walking through a badge-locked door by looking like they belong. An unlocked, already-authenticated support workstation removes the need for any of that effort at all; the badge-locked door and the fake delivery uniform become irrelevant if the workstation itself is simply sitting open. Lock the screen every time you step away, including "just a minute" — that's precisely the length of time an opportunistic attempt actually needs.

WORKED EXAMPLE: THE COFFEE BREAK

A technician steps away for coffee without locking their screen. A visitor who doesn't belong there — following Chapter 3's own impersonation playbook, dressed and behaving like they have a reason to be nearby — has a brief, genuine window at an already-authenticated ticketing system, remote-access tool, and whatever else is currently open. None of the social-engineering effort from Chapters 2–3 was even necessary; the unlocked screen did all the work for them. A locked screen defeats this scenario completely, regardless of how convincing the visitor might otherwise have been.

Session Hygiene

- **Close sessions when finished** — per `remote1`'s own "clean disconnection" standard, don't leave remote-access sessions or admin panels open indefinitely once a ticket is resolved
- **Short auto-lock timeouts** — a workstation that locks itself after a brief idle period limits the damage of a forgotten manual lock
- **Separate everyday and elevated accounts** — per `remote1`'s own least-privilege material, don't stay logged into an admin-capable account as your default; switch into it only for the specific task that needs it

Physical Security Basics

- Use a cable lock or equivalent for a laptop in any semi-public space
- Never leave a workstation unattended in a public or semi-public area, even briefly
- Use a privacy screen if your desk has visual exposure to sensitive ticket or account data from a walkway or shared space
- Store any physical security tokens or hardware keys securely, separate from the device they unlock — the same reasoning as never leaving a house key in the lock

Credential Hygiene for Your Own Accounts

Everything Chapter 5 assumed on the end user's side applies just as directly to a support technician's own accounts: a unique, strong password per account, MFA enabled everywhere it's offered, and a password manager rather than reused or written-down credentials. A technician who skips this for their own accounts while enforcing it strictly for everyone else is protecting the wrong side of the relationship.

Hands-On Exercises

EXERCISE 1

Using the comparison table, explain specifically why a compromised support workstation is described as a "pivot point" rather than just "one more compromised device."

View solution

EXERCISE 2

Explain why the coffee-break worked example says the visitor didn't need any of Chapters 2–3's social-engineering techniques to succeed.

View solution

EXERCISE 3

Explain why this chapter describes a technician who enforces Chapter 5's credential rules on end users but skips them for their own accounts as "protecting the wrong side of the relationship."

View solution

CHAPTER 9 QUICK REFERENCE

- A support workstation is a **pivot point** into many other systems, not just one more employee device
- Lock the screen every time you step away — "just a minute" is exactly the window a walk-up attempt needs
- Close sessions when done, use short auto-lock timeouts, and keep elevated accounts separate from everyday use
- Physical basics: cable locks, never unattended in public, privacy screens where visible, hardware keys stored separately
- Apply Chapter 5's own credential hygiene (unique passwords, MFA, a password manager) to your own accounts, not just the ones you support
- Next: Chapter 10, the capstone — three security-flavored support tickets, start to finish

CHAPTER 10 OF 10

Capstone: Three Security-Flavored Support Tickets, Start to Finish

Security Basics for Support Technicians

Chapter 10 · Capstone — Three Security-Flavored Support Tickets, Start to Finish

Nine chapters built the toolkit — recognizing phishing and social engineering, verifying identity, resetting credentials safely, spotting a routine ticket that isn't, handling a finding without destroying evidence, protecting data, and securing your own workstation. This capstone runs three fresh tickets through that toolkit end to end, matching the three-scenario shape most of this subject's own courses use.

Ticket 1: A Phishing Report That Wasn't Caught in Time

A user forwards a suspicious email to the support desk: "Is this legit? It's asking me to verify my account."

READING THE REPORT (CHAPTER 2)

The forwarded email shows all the classic tells: a sender domain that doesn't quite match the real company, generic greeting, and a "Verify Now" button whose visible text doesn't match its actual destination — a textbook phishing attempt.

THE DETAIL THAT CHANGES EVERYTHING

Near the end of the message, the user adds: "Actually, I did click it before I got suspicious, and I think I entered my password." What looked like a simple report-and-close ticket is now a suspected active compromise.

CAPTURE, THEN MITIGATE (CHAPTER 7)

The technician documents the phishing email itself (headers, the mismatched link, when it was received and clicked) before touching anything else, then forces a password reset per Chapter 5's own safe procedure — no plaintext credential, delivered out-of-band — and checks for any signs of unauthorized activity since the click.

ESCALATE

With the phishing email and the reset both documented, the finding is escalated through the organization's incident process, rather than treated as closed once the reset completes.

Ticket 2: An Unfamiliar App With Account Access

A user mentions, almost in passing, that they've been seeing occasional strange sign-in notifications lately — nothing that seemed urgent enough to report on its own.

RECOGNIZING IT'S WORTH A LOOK (CHAPTER 6)

Taken alone, each notification could have an innocent explanation. Asking the chapter's own question — could this also be an early sign of compromise? — the technician checks the account's connected third-party applications rather than dismissing the notifications as noise.

THE FINDING

An unfamiliar third-party app has standing access to the account's mail and files, authorized weeks ago through a login prompt the user doesn't specifically remember approving — a real, lesser-known attack pattern where a malicious app requests broad permissions through an otherwise-legitimate-looking consent screen.

CAPTURE, MITIGATE, ESCALATE (CHAPTER 7) — MINIMAL DATA TOUCHED (CHAPTER 8)

The technician documents the app's name, permissions, and grant date, then revokes its access — stopping the ongoing exposure — while pulling only the specific account's own authorization records, not a broader export. The documented finding is escalated for further investigation into what the app may have accessed while connected.

Ticket 3: An Urgent Reset Request, In Person

Someone walks up to the support desk directly: "Hi, I'm starting today, my manager said to come find you — I need my account unlocked right now, I've got a meeting in ten minutes and no one told me my temporary password."

RECOGNIZING THE PATTERN (CHAPTERS 1 AND 3)

Manufactured urgency, paired with an unverifiable backstory (a manager who isn't present, an account with no prior record the technician can check) — the same shape Chapters 1 and 3 both described, just delivered in person rather than by phone or email.

VERIFYING, APPROPRIATELY FOR THIS REQUEST (CHAPTER 4)

A password reset is high-risk regardless of channel. Rather than a badge or ID check alone — which can be faked the same way a phone number can be spoofed — the technician calls the claimed manager at an independently-looked-up number, not one the visitor provides.

RESOLUTION (CHAPTER 5) — WITH A NOD TO CHAPTER 9

The manager confirms no new hire is expected this week. The technician declines the request, documents the interaction, and locks their own workstation before stepping away to report it — the same basic habit Chapter 9 covered, applied without a second thought.

Chapter Attribution

TECHNIQUE USED ABOVE	SOURCE CHAPTER
Manufactured urgency as a recognizable pattern (Ticket 3)	Chapter 1
Reading a forwarded email's phishing red flags (Ticket 1)	Chapter 2
Recognizing an unverifiable, in-person backstory (Ticket 3)	Chapter 3
Independent callback verification for a high-risk request (Ticket 3)	Chapter 4
Safe, out-of-band forced password reset (Ticket 1)	Chapter 5
Recognizing a low-key report as worth investigating (Ticket 2)	Chapter 6
Capture, then mitigate, then escalate (Tickets 1 and 2)	Chapter 7
Pulling only the minimum data needed during investigation (Ticket 2)	Chapter 8
Locking the workstation before stepping away (Ticket 3)	Chapter 9

Honest Scope Note

WHAT THIS COURSE DELIBERATELY DOESN'T COVER

- No legal, HR, or regulatory breach-notification requirements (GDPR, HIPAA, and similar) — jurisdiction- and organization-specific, out of scope here
- No deep malware analysis or digital forensics tooling — this course teaches recognition and safe first response, not investigation techniques
- No substitute for your own organization's formal security policy — per Chapter 1, this course builds transferable judgment, not a memorized script
- No physical badge-system or building-security administration — Chapter 9 covers a technician's own habits, not facility security design
- No coding or application-level security — that's this site's own developer-focused Security subject, not this one

Hands-On Exercises

EXERCISE 1

Explain what specifically changed Ticket 1 from a routine phishing report into a suspected compromise, and which chapter's procedure took over at that point.

View solution

EXERCISE 2

Explain why Ticket 3 required an independent callback rather than accepting a badge or ID check alone, connecting this to the same reasoning used elsewhere in the course for phone-based requests.

View solution

EXERCISE 3

Explain why Ticket 2 was investigated at all, given that the user described the sign-in notifications as not urgent enough to report on their own.

View solution

CHAPTER 10 QUICK REFERENCE — COURSE COMPLETE

- Ticket 1: a phishing report that turned into a real compromise once a clicked link came to light — capture, safe reset, escalate
- Ticket 2: a low-key report investigated anyway, revealing an unauthorized third-party app with standing account access

- Ticket 3: an in-person urgent request resolved with the same verification discipline used for phone and chat requests
- The recurring theme across all ten chapters: **the request that asks you to skip a step is usually the one where the step matters most**
- This closes *Security Basics for Support Technicians*, 10/10 chapters — the seventh complete course under the Technical Support subject