

N9LYA TECHNICAL GUIDES

Cybersecurity A to Z

Threats, Defenses, and What Works

Jerome M. Kutche

N9LYA

Second Edition · September 2026

About This Book

Most cybersecurity books are written by one of two people.

The first has spent a career in enterprise IT security and writes about identity providers, cloud posture management, and a threat landscape that assumes everything you protect was manufactured after 2015 and has a software update mechanism.

The second is writing to sell a certification, and covers the exam objectives in the order the exam objectives are published.

Both produce useful books. Neither is quite this one.

I spent thirty-plus years in industrial electrical and controls work — motor control centers, Allen-Bradley and Siemens PLC systems, plant electrical maintenance and supervision. In that world, a “legacy system” is a controller installed in 1994 that runs a process worth several thousand dollars an hour, has no patch available, cannot be taken offline for testing, and was designed in an era when the idea that it might be attacked was not a design consideration at all. The security literature mostly does not talk about that world, and it is a very large part of the actual world.

So this book covers cybersecurity from the ground up — the concepts, the attacks, the defenses, the operations, the career — and then it also covers the parts most books skip. There is a full treatment of operational technology and industrial control system security, written by someone who has actually stood in front of the equipment. There is a homelab thread throughout, because the fastest way to understand security is to build something and then try to break it. And there is an unusual amount of attention paid to the physical and electrical layer, because a great many “cyber” incidents begin with someone walking through an unlocked door.

What This Book Covers, and What It Doesn't

It covers: how attacks actually work and why they succeed, how to build systems that resist them, how to detect what you couldn't prevent, how to respond when something happens, and how to build a career doing this.

It does not include working exploit code, malware, or step-by-step attack instructions. Not because the topics are forbidden — you cannot defend against what you don't understand, and this book explains attack mechanics in real detail — but because a defender needs to understand how a technique works and why it succeeds, which is different from possessing a tool that executes it. Where you need hands-on offensive practice, Chapter 25 shows you how to build a lab and points you at the intentionally vulnerable environments designed for exactly that purpose.

It is not a certification cram guide, though it covers most of what Security+ tests and a fair portion of what CySA+ and SSCP touch. Chapter 26 discusses certifications honestly, including which ones are worth the money.

Who Should Read It

Complete beginners should start at Chapter 1. Parts I and II assume nothing beyond general computer literacy and build the vocabulary and mental models everything else depends on.

IT people moving into security can skim Part II and start at Part III. You know the technology; what you need is the adversarial framing and the operational discipline.

People with OT or industrial backgrounds — electricians, controls technicians, plant engineers — will find Part I and Chapter 23 the most immediately useful, and should read Part II carefully. Your domain knowledge is genuinely scarce in this field. What you're missing is the IT security vocabulary, and it's learnable.

Homelabbers will find Chapter 25 and the exercises throughout are written for you specifically.

How to Use It

Each chapter ends with a **Lab Check** — verification tasks and exercises you can perform on systems you own. Do them. Security knowledge that has never been tested against a real system is not knowledge, it's vocabulary.

Everything in this book assumes you are working on systems you own or have written authorization to test. Chapter 4 covers why that sentence is not boilerplate.

About the Author

Jerome M. Kutche holds the amateur radio callsign N9LYA, first licensed in 1991. He spent over three decades in industrial electrical and controls work across manufacturing, materials, and heavy industry — Allen-Bradley and Siemens PLC systems, motor control, plant electrical maintenance and supervision — and holds NFPA 70E, lockout/tagout, Six Sigma Yellow Belt, and thirty-nine other professional certifications.

He serves as Indiana's 44Net/AMPRNet coordinator, administering the state's allocation within the amateur radio IPv4 address space, and is trustee for two amateur radio clubs. He operates a segmented homelab that has been audited, hardened, credential-rotated, and rebuilt more times than he cares to count, and a packet radio node that has taught him a great deal about what happens when you put a service on the public internet and wait.

Part I — Foundations

Chapter 1: What Security Actually Is



Figure 1: Cybersecurity risk functions

Security is not a product, a state, or a checkbox. It is a continuous process of managing risk to an acceptable level, where “acceptable” is a business decision and not a technical one.

That sentence contains most of what separates people who understand this field from people who have memorized tools.

The Three Properties

The classic model — the CIA triad — describes what you’re protecting:

Confidentiality. Information is available only to those authorized to see it. Breached by data theft, eavesdropping, misconfigured access controls, an unencrypted backup drive left in a car.

Integrity. Information is accurate and has not been altered without authorization. Breached by tampering, corruption, unauthorized modification. In many contexts this matters more than confidentiality — a falsified financial record or an altered process setpoint can do more damage than a leaked one.

Availability. Systems and information are accessible when needed. Breached by denial of service, ransomware, hardware failure, or an outage caused by your own change.

Two additions are worth knowing:

Authenticity — you can verify who created or sent something. **Non-repudiation** — the originator cannot credibly deny having done so.

The ordering differs by domain, and this is the single most important thing to internalize. In corporate IT, confidentiality usually leads: the nightmare is a data breach. In industrial control systems, **availability and integrity lead and confidentiality trails badly** — nobody much cares if an attacker learns the temperature of a furnace, but they care enormously if the furnace stops or if the temperature reading is falsified. In healthcare, integrity and availability are life-safety concerns.

A security program designed with the wrong priority ordering for its domain will protect the wrong things well. Chapter 23 develops this at length.

Threat, Vulnerability, Risk

These three words get used interchangeably by people who should know better. They are different.

A threat is something that could cause harm. A ransomware group. A disgruntled employee. A flood. Threats exist independent of you.

A vulnerability is a weakness that a threat could exploit. Unpatched software. An employee who hasn’t been trained to recognize a phishing email. A server room below the water line.

Risk is the combination: the likelihood that a threat exploits a vulnerability, times the impact if it does.

$\text{Risk} = \text{Likelihood} \times \text{Impact}$

This matters practically because **you reduce risk by acting on any of the three terms**, and the cheapest lever is not always the technical one.

You usually cannot eliminate threats. You can reduce vulnerabilities (patching, hardening, training). You can reduce impact (segmentation, backups, insurance). You can reduce likelihood (making yourself a harder target than the next candidate).

A backup does nothing to prevent ransomware. It reduces the impact from catastrophic to inconvenient, which is frequently a better investment than another prevention layer.

The Four Responses to Risk

Every risk gets one of four treatments, and naming which one you chose is what separates a security program from a pile of tools:

Mitigate. Reduce it. Patch the system, add the control, train the people.

Transfer. Move it to someone else. Insurance, or a contract that puts liability on a vendor. Note that you cannot transfer reputational damage or regulatory obligation, only money.

Avoid. Stop doing the risky thing. Decommission the legacy application. Don't collect the data you don't need — data you never collected cannot be breached, which is the most underrated control in this entire book.

Accept. Decide the risk is tolerable and document that decision, with a name attached to it.

Accepting risk is legitimate and necessary. You cannot mitigate everything and attempting to will exhaust your budget on low-value controls. What is not legitimate is accepting risk implicitly, by never having assessed it. The difference between a mature program and an immature one is not the number of accepted risks — it's whether they were accepted deliberately, by someone with authority to do so, in writing.

Controls

Controls are categorized two ways, and knowing both taxonomies helps you find gaps.

By type: - **Technical** — firewalls, encryption, access control lists, multi-factor authentication - **Administrative** — policies, procedures, training, background checks - **Physical** — locks, fences, cameras, badge readers, that door someone props open with a fire extinguisher

By function: - **Preventive** — stops it happening. A lock. - **Detective** — notices it happened. A camera. - **Corrective** — fixes it after. A backup restore. - **Deterrent** — discourages the attempt. A visible sign. - **Compensating** — an alternative when the primary control isn't feasible. This is the one that matters enormously in OT, where you often cannot patch and must compensate with isolation and monitoring instead.

Most organizations over-invest in preventive technical controls and under-invest in detective controls and administrative ones. Prevention eventually fails. Detection is what limits how bad the failure gets.

Defense in Depth

No single control holds. Layer them so that a failure in one is caught by another.

A useful mental image: an attacker must traverse the perimeter, the network segment, the host, the application, and the data — and each layer should independently attempt to stop, slow, or reveal them.

The design question is not "is this control sufficient?" It is "**when this control fails, what catches it?**" If the answer is nothing, you have a single point of failure regardless of how good that control is.

Least Privilege

Every user, process, and system gets exactly the access it needs to do its job, and nothing more.

This is the highest-value principle in the entire field, and the most consistently violated. Reasons it gets violated: granting broad access is faster than working out what's actually needed; revoking access breaks things and nobody wants to be the person who broke something; and permissions accumulate as people change roles and nobody ever removes the old ones.

The consequence is that most breaches escalate further than they should. An attacker who compromises one account should be contained by what that account can reach. When every account can reach everything, one compromise is total compromise.

Related: **separation of duties** (no single person can complete a sensitive process alone) and **need to know** (access is limited to what's required for a specific task, not a general clearance level).

Assume Breach

The mature posture. Do not design as though your perimeter will hold — design as though an attacker is already inside, and ask what limits their movement, what detects them, and how quickly you'd know.

This reframing changes investment priorities. It makes segmentation more valuable than a better firewall. It makes logging and detection more valuable than another prevention product. It makes tested backups more valuable than nearly anything.

Zero trust is the formalization of this idea: no implicit trust based on network location. Every access request is authenticated, authorized, and evaluated regardless of whether it originates inside or outside your network. The marketing around the term is dreadful; the underlying principle is sound.

Security Costs Something

Every control has a cost beyond its price: friction. Multi-factor authentication adds seconds to every login. Least privilege adds requests to a helpdesk. Network segmentation adds troubleshooting complexity.

Security that is too painful gets bypassed, and the bypasses are invisible and worse than the original risk. The employee who can't get access through the proper channel emails the file to their personal account. The engineer who can't reach the PLC through the jump host plugs a laptop directly into the control network.

Designing controls that people can actually live with is not a compromise of security. It is security. A control that is worked around provides zero protection and creates a false sense of coverage, which is worse than no control at all.

Lab Check

1. Pick a system you own. Name one threat, one vulnerability, and state the risk in a sentence.
2. For that risk, which of the four responses have you actually chosen? Was it deliberate?
3. List your controls on that system by function. Which functions have nothing in them?
4. Identify one place where least privilege is violated in your own environment. Why?

Chapter 2: Thinking Like an Attacker

Defenders think in systems, controls, and coverage. Attackers think in paths.

That difference explains most defensive failures. A defender looks at a network diagram and sees ten protected systems. An attacker looks at the same diagram and sees the one path through the printer's management interface to the file server.

Learning to see paths is the core skill of this field.

The Asymmetry

A defender must be right everywhere, always. An attacker must be right once.

This is quoted constantly and it is only half true, which makes it dangerous.

The true half: your attack surface is large and any weakness is a candidate.

The false half: it implies defense is hopeless, and it isn't — because **the attacker must also chain multiple successes together, and they must do it without being detected**. Initial access is rarely enough. They need privilege, persistence, discovery, lateral movement, and eventually a way to accomplish their goal. Every step is an opportunity to fail and an opportunity to be caught.

This is why detection matters as much as prevention. You do not need to stop the first step. You need to stop or notice any step in the chain before they reach their objective.

What Attackers Actually Optimize For

Attackers are not adversaries in a movie. They are people or organizations with constraints, and understanding their economics predicts their behavior.

Most attacks are opportunistic, not targeted. An automated system scans the entire internet continuously for known vulnerabilities and default credentials. It does not know or care who you are. If you are vulnerable to something in its list, you get compromised. The overwhelming majority of small-organization and home compromises are this.

Defense against opportunistic attack is straightforward and boring: patch, don't expose things, use unique credentials, enable MFA. It works because you don't have to be secure, only harder than the next candidate.

Targeted attackers are different. Someone who specifically wants into your environment will invest time, research your people, and adapt when blocked. Defense here requires detection, because prevention will eventually be circumvented by someone patient.

The economics matter. Criminal groups run a business. They want the highest return for the lowest effort and risk. Ransomware dominates because it monetizes access directly, at scale, without needing to find a buyer for stolen data. Understanding that an attacker is choosing between targets means that raising your cost — even modestly — moves you down their list.

Nation-state actors have different constraints. They optimize for access and persistence, not immediate profit, and they are willing to spend years. Most readers of this book are not their target, and pretending otherwise leads to spending on the wrong things.

Attack Surface

Your attack surface is every point where an attacker could interact with your systems.

External: internet-facing services, web applications, VPN endpoints, email (a channel directly into your organization that you cannot close), DNS, cloud storage, anything a supplier connects to you with.

Internal: every service on every host, every share, every account, every trust relationship between systems.

Human: every person who can be phoned, emailed, or approached. This surface is often the largest and least defended.

Physical: every door, every unattended workstation, every network jack in a lobby, every device someone could plug in.

Supply chain: every vendor, every piece of software, every library your software depends on, every managed service provider with access to your environment.

The single most effective security activity available to most organizations is reducing attack surface. Not adding controls — removing exposure. Decommission the server nobody uses. Close the port. Uninstall the software. Delete the account. Remove the vendor's standing access.

Every removal is a permanent reduction requiring no maintenance. Every added control is a thing you must now operate correctly forever.

The Path Mindset

Practice this. Pick any environment you know and trace paths.

- If someone compromised the receptionist's workstation, what could it reach?
- Which accounts can log into the domain controller? Where else do those accounts log in?
- If the wifi password leaked, what's on that network?
- Which vendor has remote access, and what can they touch?
- Where do credentials get stored, and who can read them?
- What system trusts what other system, and why?

The paths that matter are the ones nobody drew on the architecture diagram. The management VLAN reachable from the guest network because of a misconfigured trunk. The service account with domain admin used by a monitoring tool on an unpatched server. The backup system with credentials to everything, sitting on the same network as the workstations.

Attackers find these because they look for them systematically and defenders don't.

Living Off the Land

An important modern reality: sophisticated attackers increasingly avoid malware entirely, using the legitimate administrative tools already present on your systems.

Remote management frameworks, scripting engines, built-in credential and system utilities, scheduled task mechanisms — all are administrative tools with legitimate uses, and all are useful to an attacker who has obtained access.

This defeats detection strategies based on identifying malicious files, because there is no malicious file. What detects it is **behavioral**: this administrative tool has never run on this workstation before; this account is authenticating to systems it has never touched; this process is spawning children it never spawns.

Which is why Chapter 18 argues that logging and baseline behavior matter more than another scanning product.

Ethical Framing

You will spend this book learning how attacks work. That knowledge is neutral. What makes someone a security professional rather than a criminal is not knowledge — it is authorization and intent.

Every technique described in this book is legal to practice on systems you own, and a crime to use on systems you don't. There is no ambiguity in that sentence, no gray area, and no defense based on good intentions. Chapter 4 covers the specifics.

Build a lab. Break your own things. That is where this knowledge belongs.

Lab Check

1. Draw your own environment. Now find three paths that aren't on the drawing.
2. List everything of yours reachable from the internet. Justify each or remove it.
3. Identify the account in your environment with the broadest access. Does it need it?
4. Name one thing you could decommission entirely this month.

Chapter 3: Threat Modeling

Threat modeling is the practice of thinking systematically about what could go wrong before it does. It is the highest-leverage security activity that costs nothing but time, and it is skipped almost universally.

The Four Questions

Every threat modeling methodology reduces to four questions:

1. What are we building or protecting? 2. What can go wrong? 3. What are we going to do about it? 4. Did we do a good job?

If you do nothing else from this chapter, sit down with those four questions and a whiteboard before your next significant change. That alone will surface things nobody had considered.

Threat Modeling

Connect valued assets to realistic attack paths and controls

ASSETS

Data / services
Safety / trust

ADVERSARIES

Capability / intent
Opportunity

ATTACK SURFACE

Identity / network
Software / people

CONTROLS

Prevent / detect
Recover

N9LYA TECHNICAL GUIDES

Figure 2: Threat modeling

Start With Assets

You cannot protect what you haven't identified. Most organizations cannot produce an accurate inventory of what they have, which means their security program is protecting an imagined environment.

Inventory: - Data — what you hold, where it lives, how sensitive it is, who can reach it - Systems — every device, service, and application

- People — who has access to what - Dependencies — what you rely on that you don't control

Then classify by impact. Not everything deserves equal protection. A useful three-tier scheme:

Crown jewels — loss or compromise is existential. Customer data, financial records, the process control system, the intellectual property the business is built on. Important — significant disruption, recoverable. Most business systems. Routine — annoying to lose, not consequential.

The value of this exercise is that it forces prioritization. Security budgets are finite and spreading them evenly across everything protects the crown jewels no better than the routine systems.

STRIDE

A structured way to answer “what can go wrong,” developed at Microsoft. For each component, ask whether it's susceptible to:

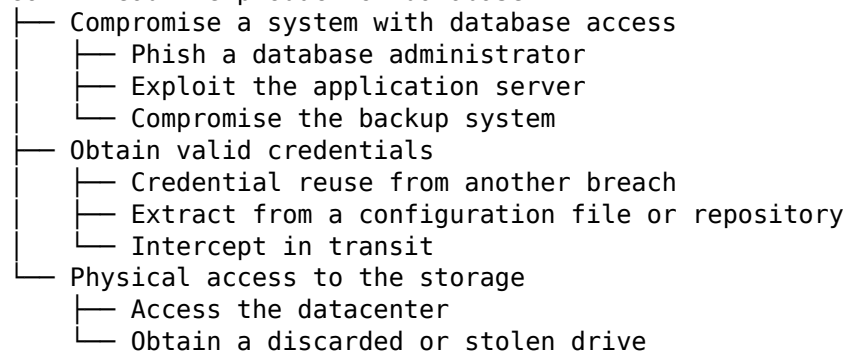
Category	The Threat	The Property Violated
Spoofing	Pretending to be someone else	Authenticity
Tampering	Modifying data or code	Integrity
Repudiation	Denying having done something	Non-repudiation
Information disclosure	Exposing data to the unauthorized	Confidentiality
Denial of service	Making something unavailable	Availability
Elevation of privilege	Gaining rights you shouldn't have	Authorization

Walk a system component by component and apply all six. It is mechanical, it is a little tedious, and it reliably surfaces things pure intuition misses — particularly repudiation and elevation, which people rarely think about unprompted.

Attack Trees

A different structure: put the attacker's goal at the root and decompose it into the ways it could be achieved.

GOAL: Read the production database



Then evaluate each leaf: how difficult, how likely, what would detect it, what would stop it.

This is excellent for finding the cheap path. Defenders tend to fortify the obvious route while a much easier one sits unexamined. In the tree above, the branch that usually works is a credential in a configuration file committed to a repository — not the exploit anyone was worried about.

MITRE ATT&CK

ATT&CK is a structured knowledge base of real-world adversary behavior, organized by tactic (the attacker's goal at each stage) and technique (how they achieve it).

Tactics run roughly: reconnaissance, resource development, initial access, execution, persistence, privilege escalation, defense evasion, credential access, discovery, lateral movement, collection, command and control, exfiltration, and impact.

Its practical value for a defender is **coverage mapping**. Take the techniques relevant to your environment and ask, for each: could we prevent this? Could we detect it? The result is an honest map of your blind spots, and it is usually uncomfortable.

There is a dedicated ATT&CK matrix for industrial control systems, which is the right reference for anyone working in OT and which Chapter 23 uses.

Do not attempt to cover everything. Start with the techniques that appear most frequently in incidents affecting organizations like yours.

Practical Threat Modeling

A session that actually works:

1. **Diagram it.** Boxes for components, arrows for data flows, and — critically — **draw trust boundaries** where data crosses from one level of trust to another. Boundaries are where most vulnerabilities live.
2. **Walk the flows.** For each arrow: what is it carrying, is it authenticated, is it encrypted, is it validated on the receiving end?
3. **Apply STRIDE** to each component.
4. **Write down what you find**, with an owner and a decision. Mitigate, transfer, avoid, or accept.
5. **Revisit** when the system changes materially.

Ninety minutes with the right four people and a whiteboard beats any tool. The output is not a document — the output is the shared understanding in the room, and the document is just its residue.

Common Mistakes

Modeling the diagram instead of reality. Diagrams are aspirational. Verify what actually connects to what.

Ignoring the human and physical layers. Most models cover only technical components. The receptionist and the propped-open door are part of the system.

Only modeling external attackers. Insiders — malicious, careless, or compromised — bypass your perimeter entirely by definition.

Skipping the supply chain. Your vendors' security is your security. A managed service provider with standing administrative access to your environment is one of the largest risks most small organizations carry, and it is almost never modeled.

Producing a document nobody reads. The value is in the thinking and the decisions. A threat model in a folder that nobody has opened in two years is an artifact, not a control.

Lab Check

1. Diagram one system you own, including trust boundaries.
2. Apply all six STRIDE categories to it. Write down what you find.
3. Build an attack tree for the most valuable thing you protect.
4. Pick five ATT&CK techniques and honestly assess: could you detect each?
5. For every finding, record a decision and who made it.

Chapter 4: Authorization, Law, and Ethics

This chapter is short, and it is the one that keeps you out of prison. Read it before you touch anything.

The Central Rule

Accessing a computer system without authorization is a crime. In the United States this is primarily the Computer Fraud and Abuse Act; nearly every country has an equivalent, and many are broader.

Three things people get wrong:

Good intentions are not a defense. Scanning someone's network to helpfully report a vulnerability is unauthorized access. People have been prosecuted for exactly this. If you find something by accident, report it through a proper channel — do not investigate further to “confirm” it.

Exceeding authorized access is also covered. Having legitimate access to a system does not authorize you to access parts of it you weren't granted. An employee who reads records they have no business need for may be committing a crime, not merely a policy violation.

“It was easy” is not authorization. An unlocked door is not an invitation. A default password is not consent. A misconfigured share that anyone can read does not make reading it lawful.

What Authorization Looks Like

Real authorization is **written, specific, and from someone with the authority to grant it.**

A proper scope agreement specifies:

- **What** systems, by IP address, hostname, or domain — explicitly, not “our network”
- **What** techniques are permitted and which are excluded
- **When** testing may occur
- **Who** is authorized to perform it, by name
- **Who** to contact immediately if something breaks or if you find evidence of an actual compromise
- **What** you may do with the data you encounter
- **A statement** that the signer has authority to grant this

Verbal permission from your manager is not sufficient. Permission from a customer to test a system that a third party actually hosts is not sufficient — the hosting provider must also authorize it, and many explicitly prohibit it in their terms.

If the scope is ambiguous, stop and clarify before proceeding. The moment you exceed scope you have lost the protection the agreement provided.

Practicing Legally

You want hands-on experience. Here is how to get it without risk:

Your own lab. Systems you own, on a network you own, isolated from everything else. This is unlimited and Chapter 25 shows you how to build one.

Intentionally vulnerable environments. Purpose-built applications and virtual machines designed to be attacked, published specifically for training. Run them locally in your lab.

Legal practice platforms. Commercial and free services that provide hosted vulnerable environments with explicit authorization built into their terms of service. These are legitimate and widely used for skill development.

Capture the flag competitions. Sanctioned, scoped, and often excellent.

Bug bounty programs, carefully. These grant authorization within a published scope. Read the scope precisely and stay inside it — the authorization ends exactly where the scope does, and researchers have gotten into serious trouble by wandering outside it.

End of the free sample

Cybersecurity A to Z — Second Edition · by Jerome M. Kutche

That was the first 12 of 72 pages. The full book continues with the practical detail, worked examples and recovery steps that don't fit in a sample.

Get the full book (\$9.99, PDF + EPUB):

<https://booksrus.biz/product/cybersecurity-a-to-z/>

Also on Gumroad:

<https://n9lya4.gumroad.com/l/jbwwd>



All 17 Second Edition guides in one bundle for \$89.99:

booksrus.biz/product/n9lya-technical-guides-complete-bundle

This sample may be shared freely in unmodified form. © Jerome M. Kutche. Second Edition.