GUERILLA SECURITY

THE MARTIAL ART OF INFORMATION SECURITY

2026 Edition, Revision 1

Originally published 1994. Revised 1998, 1999, 2000.

This revision: April 2026.

Abstract

The U.S. Marine Corps says "Killing tanks is fun and easy." That's what threat actors think of your information systems. Guerilla Security is a methodology for dealing with constant change in the security environment, stressing flexible policies, continuous risk assessment, and rapid response. This book covers the modern threat landscape — from ransomware and supply chain attacks to AI-augmented phishing and cloud misconfiguration; a practical framework for building and maintaining a security program using the RAPID methodology; and the governance, risk management, and compliance requirements that drive security decisions in regulated industries.

The core philosophy hasn't changed since 1994: effective security, like effective self-defense, requires practice, adaptability, and a realistic understanding that you cannot prevent all attacks — only prepare to survive them.

Andrew T. Robinson
RESCOR LLC
521 Lincoln Road, West Enfield, ME 04493
+1 863 SECURE1 (+1 863 732-8731)
www.rescor.net

Table of Contents

PART I: PHILOSOPHY AND FOUNDATION

Chapter 1: Introduction

Why Information Security Matters

The security of information systems is not a new concern. The fundamental principles have not changed since Julius Caesar used substitution ciphers to protect military communications. If the principles are well established, why write a book about them?

Three reasons — and they are the same three reasons I cited when I first wrote this book in 1994, except that each has intensified by orders of magnitude.

First, the risk of compromise is increasing. In 1994, most organizations were just connecting to the Internet for the first time. In 2026, every organization depends on information systems for its core operations, and those systems are interconnected through cloud services, SaaS platforms, APIs, AI tools, mobile devices, and supply chain dependencies that did not exist a decade ago, much less three decades ago. The attack surface has expanded from "the network perimeter" to "everything, everywhere, all the time."

Second, many more organizations are at risk. In 1994, Internet security was primarily a concern for government agencies, universities, and technology companies. In 2026, every organization — from a three-person dental practice to a multinational bank — depends on connected systems and handles information that someone, somewhere, wants to steal, encrypt, or exploit. The smallest organizations are often the most vulnerable, because they lack the resources and expertise that larger organizations can bring to bear.

Third, inertia remains the enemy. Established security policies and practices still have a high level of bureaucratic and administrative inertia. The people who design the policies are often removed from the operational realities. The people on the front lines either circumvent the policies to get work done or follow them robotically without understanding their purpose. Frameworks are adopted as checklists rather than tools for thinking. Controls are implemented once and never revisited.

Guerilla Security provides practical information on building and maintaining a strong security program. It introduces the RAPID methodology — a process of continuous, iterative improvement that can be used by executives, managers, technicians, and end users. I use the term "guerilla" because it implies flexibility and rapid response over monolithic, high-inertia security practices. My intent is that Guerilla Security should be to traditional information security what Desert Storm was to the Maginot Line.

Adopting a Realistic View

As part of my martial arts training, I practice "empty hand" techniques against knife-wielding attackers. At one point, I became very frustrated with my progress on these techniques. "These techniques don't work," I told my instructor. "I'll get cut in a real knife fight."

My instructor's response was: "Of course you will get cut, and you will die."

At first I thought this was absurd: why should I bother to study knife defense techniques if I had no chance to walk away? But my instructor's comment was more subtle than it seemed. First, I had to be prepared to take damage in a knife fight. More importantly, I had to approach the fight as if my life was on the line. I had to take the threat seriously, train realistically, or the damage might well be fatal.

What has this got to do with information security?

"I will be hacked, and my network will die." — Second Law of Guerilla Security

I will be hacked. The probability of compromise over time approaches certainty. It may be an external attacker, an insider, a compromised vendor, or an AI-generated phishing email that defeats your training and your filters. The threat environment changes faster than any defense can adapt while keeping the organization functional. You *cannot* have perfect security and still get useful work done.

My network will die — if I don't take the threat seriously. Avoidance is not an option. Buying a firewall and forgetting about it is not a strategy. Threat actors will find the door, the window, the side entrance, or the vendor with the weak password. Effective security, like effective self-defense, requires practice — continuous, realistic, and never-ending. If you fail to stay in practice, the attacker will have their way with your information resources. This can result in monetary, legal, productivity, reputational, and regulatory losses. For some organizations, a single successful attack can be an existential event.

The Three Laws of Guerilla Security

These three laws form the foundation of everything in this book:

1. **"My security is my responsibility."** — The First Law. No vendor, no framework, no regulation will do it for you. Acknowledging this responsibility is the first step.
2. **"I will be hacked, and my network will die."** — The Second Law. Assume breach. Build your defenses around detection, response, and recovery — not just prevention. Prevention will fail; the question is whether you are prepared when it does.
3. **"Whatever I know now that isn't wrong, will be tomorrow."** — The Third Law. Constant change demands constant validation. The software you patched last week has a new vulnerability this week. The threat that targeted your industry last quarter has new tools this quarter. The compliance framework you implemented has new requirements. Annual assessments are snapshots of a moving target.

These laws are explored in depth in Chapter 2. They are not pessimistic — they are realistic. The organization that accepts these realities and plans accordingly is far more resilient than the one that believes its defenses are adequate.

How This Book Is Organized

Part I: Philosophy and Foundation establishes the principles that inform every subsequent chapter — the Three Laws, the five security archetypes, the concepts of entropy and energy in risk management, and the RAPID methodology for continuous program improvement.

Part II: The Modern Threat Landscape describes who attacks you, how they do it, and what happens when they succeed. This section has changed more than any other since the original edition, because the threat landscape of 2026 bears almost no resemblance to that of 2000.

Part III: The Defense covers the practical elements of a security program — security policy, identity and access management, network and endpoint security, and data protection. Each section is grounded in modern technology and regulatory requirements, not the 1990s toolkit of the original edition.

Part IV: Detection, Response, and Recovery addresses what happens when prevention fails. Incident response, business continuity, and security testing are not afterthoughts — they are core capabilities that distinguish organizations that survive incidents from those that don't.

Part V: Governance, Risk, and Compliance covers the management framework that ties everything together. Risk management (using quantitative methods), regulatory compliance, third-party risk, and AI governance are the topics that keep CISOs, compliance officers, and board members awake at night.

The appendices provide a security program policy inventory, a modern glossary, a framework and regulatory reference guide, and a bibliography.

A Note on Terminology

The original edition of this book used the term "infosecurity" — a contraction of "information systems security" that was common in the 1990s but has since fallen out of use. This edition uses "information security" or simply "security" throughout.

The original also used "hacker" in the popular sense of a malicious actor. The classical definition of hacker — a person with deep technical knowledge and curiosity — has no negative connotation. The security industry has largely adopted "threat actor" or "adversary" for malicious actors, and "hacker" for the broader community. This edition uses "threat actor" or "attacker" when describing malicious activity, and reserves "hacker" for its classical meaning where appropriate.

Chapter 2: The Three Laws of Guerilla Security

The Three Laws are not technical recommendations. They are philosophical commitments — ways of thinking about security that determine whether your program succeeds or becomes an expensive exercise in wishful thinking. Every technical control, every policy, every budget decision should be evaluated against these laws.

The First Law: "My Security Is My Responsibility"

The First Law is about ownership. Your information security is your problem. Not your vendor's problem. Not your cloud provider's problem. Not your regulator's problem. Yours.

This seems obvious, but the history of security breaches is a history of organizations assuming someone else was handling it. The cloud provider will secure the infrastructure. The firewall vendor will stop the attacks. The compliance framework will tell us what to do. The managed security provider will watch the alerts.

Every one of those assumptions has been proven wrong, repeatedly, at scale.

Cloud providers operate on a shared responsibility model: they secure the infrastructure, you secure your configuration, your data, and your access controls. Most cloud breaches are not infrastructure failures — they are customer misconfigurations. An S3 bucket left open. An Azure storage account with public access. A GCP service account key committed to a public repository. The provider's infrastructure was fine. The customer's configuration was not.

Firewall vendors sell products that work as designed. But a firewall that permits everything it should and blocks everything it shouldn't is only as effective as the policy it implements. A poorly configured firewall provides a false sense of security that is worse than no firewall at all — because at least without a firewall, you know you're exposed.

Compliance frameworks — NIST CSF, ISO 27001, HIPAA, GLBA — provide structure for thinking about security. They do not provide security. An organization that implements every control in a framework and declares itself "compliant" may still be vulnerable to the specific threats that matter most in its environment. The framework tells you what to think about. It does not tell you what to do. And it certainly does not do it for you.

The First Law means: My security is my responsibility. I will build internal capability. I will understand my risk environment. I will make deliberate decisions about what to protect and how. I will not outsource my thinking.

The Second Law: "I Will Be Hacked, and My Network Will Die"

The Second Law is about realism.

The probability of a security compromise over time approaches 100%. This is not a failure of your security program — it is a statistical reality. Given enough time, enough attackers, enough attack surface, and enough human fallibility, something will get through. The question is not *whether* you will be compromised, but *when*, *how badly*, and *how quickly you recover*.

This law has two premises:

I will be hacked. Bill Cheswick — the father of the modern firewall — warned decades ago about the "crunchy shell around a soft, chewy center." Organizations that invest everything in perimeter defenses and nothing in detection, response, and recovery are betting their survival on a wall that only needs to fail once. In 2026, the perimeter doesn't even exist in a meaningful sense. Cloud services, SaaS platforms, remote workers, API integrations, AI tools processing your data, supply chain dependencies — the attack surface extends far beyond anything a firewall can control.

My network will die if I don't take the threat seriously. Attackers don't use a single vector. They probe every door, every window, every unlocked side entrance. They compromise vendors. They exploit trust relationships. Ransomware operators don't just encrypt your data — they exfiltrate it first, delete your backups if they can reach them, and publish what they stole if you don't pay.

If you accept that compromise is inevitable, your priorities shift:

- **Detection** becomes as important as prevention. Can you tell when something is wrong? How fast?
- **Response** becomes a core capability. Do you have a plan? Has anyone actually read it? Have you tested it?
- **Recovery** becomes a business requirement. How fast can you restore operations? From what state? With what data loss?

This is what the industry now calls "assume breach" and what zero-trust architecture implements at the technical level. We were saying it in 1994 — before it had a name, before it was a framework, and before it was fashionable.

The Second Law means: I will plan for failure. I will invest in detection and response, not just prevention. I will test my incident response plan. I will test my backups. The organization that assumes it will be breached and prepares accordingly survives. The one that assumes its defenses are adequate does not.

The Third Law: "Whatever I Know Now That Isn't Wrong, Will Be Tomorrow"

The Third Law is about humility. The one constant in information security is change, and the areas of change are broader than most organizations acknowledge:

Five areas of constant change:

Software. Every new feature is a new attack surface. Every update fixes old vulnerabilities and occasionally introduces new ones. The supply chain attacks of recent years — SolarWinds, MOVEit, 3CX, the XZ Utils backdoor — demonstrated that software you trust can become the vector, not through negligence but through deliberate compromise of the development pipeline itself.

Customer and user expectations. Users demand more access over time, not less. Remote work, BYOD, cloud collaboration, AI tools — each new capability expands the trust boundaries that threat actors can exploit. If your security program doesn't adapt to how people actually work, they will find workarounds that are worse than anything you were trying to prevent.

Encryption and authentication. 56-bit DES was once "strong enough." Then 128-bit AES was the standard. Now we're planning the transition to post-quantum algorithms because the cryptographic assumptions underlying RSA and elliptic curve may not survive the next decade. MFA methods that were considered strong — SMS one-time passwords — are now known to be vulnerable to SIM swapping and real-time phishing proxies. The only password policy that consistently matters is *length*. Complexity rules that mandate special characters and mixed case actually *reduce* entropy by constraining the search space and encouraging predictable substitution patterns (P@ssw0rd). Frequent rotation causes users to choose weaker passwords or increment a counter. Three-strike lockouts don't reflect the reality that legitimate users fat-finger their passwords more than three times with reasonable frequency.

"Best practices." There is no such thing as a "best practice" in information security. There are only good practices, conditioned by your specific risk environment, your resources, your regulatory obligations, and the current threat landscape. What's best for a 50-bed rural hospital is not what's best for a $2 billion commercial bank. What was best for either of them last year may not be best this year. Frameworks like NIST CSF and ISO 27001 provide structure for thinking, not answers. The organization that implements a framework as a checklist and declares itself "compliant" has confused the map for the territory.

Threats. In 2000, the primary threats were script kiddies exploiting known vulnerabilities and early worms propagating through unpatched systems. In 2010, it was organized crime and nation-state actors conducting targeted intrusions. In 2026, it is ransomware-as-a-service operators using AI to generate unique phishing content at scale, supply chain compromises embedded in trusted software, ciphertext harvesting for future quantum decryption, and AI-powered reconnaissance that can map an organization's attack surface faster than any human team.

The Third Law means: I will build programs designed to adapt, not endure. Continuous risk assessment — not annual. Iterative improvement — not multi-year transformation projects. Control validation — not "set and forget." The organizations that thrive are those that accept constant change and build it into their operating rhythm.

The Laws in Practice

The Three Laws work together:

- The First Law: I own this. It's my responsibility.
- The Second Law: Prevention alone will fail. I will plan for it.
- The Third Law: Whatever I build will need to change. I will adapt.

Together, they point toward a security program that is owned (not outsourced), resilient (not just preventive), and adaptive (not static). Ownership realism humility. This is the foundation on which the RAPID methodology, described in Chapter 5, is built.

Chapter 3: The Security Archetypes

Animal Styles

In martial arts, "animal styles" describe distinct approaches to combat. Crane style emphasizes balance and precision. Tiger style emphasizes power and aggression. Each style has genuine strengths, but a practitioner who can only fight one way is predictable — and predictability is fatal.

Information security programs exhibit the same pattern. Organizations develop a dominant style — a default approach to risk, controls, and adaptation — that reflects their culture, leadership, resources, and history. That style determines how they respond to threats, how they invest in controls, how they treat their workforce, and how they adapt (or don't) when the environment changes.

The five security archetypes described below are not arbitrary labels. They are positions along four dimensions that every security program exhibits: **awareness** (how well the organization understands its risk environment), **action** (how effectively it translates awareness into controls), **balance** (whether its response is proportionate to the actual risk), and **adaptation** (how quickly it adjusts when conditions change). The archetypes are referenced throughout this book wherever a concept maps to a characteristic behavioral pattern.

The Five Archetypes

The Ostrich is fast, nimble, and focused entirely on the business. Security is friction that slows things down, and the Ostrich would rather outrun threats than build defenses against them. The Ostrich's strength is agility and business focus — it doesn't over-engineer solutions or let fear drive decisions. Its weakness is that speed is not a strategy. Ransomware, supply chain compromise, and insider risk don't care how fast you move. The Ostrich scores low on awareness and action, because it has chosen not to look.

The Sloth is aware of the risks. It has read the headlines. It understands that something should probably be done. But the cost, the effort, and the organizational disruption of building a real security program keep it hanging comfortably on the branch, hoping nothing shakes the tree too hard. The Sloth's strength is pragmatism — it knows the difference between real risk and security theater. Its weakness is that awareness without action is negligence. Regulators and examiners are increasingly intolerant of organizations that understand their risks but choose not to address them. "We knew but didn't act" is worse than "we didn't know."

The Tortoise has built a hard shell — firewalls, intrusion detection, endpoint protection, compliance certifications. It believes in technology and process as the answer to security threats, and it has invested accordingly. The Tortoise's strength is measurable investment in controls. Its weakness is the "crunchy shell, soft chewy center" problem (Chapter 2): once an attacker gets past the perimeter — through phishing, a compromised vendor, or a misconfigured cloud service — there is little to stop lateral movement. Technology without culture is armor without a soldier inside it. The Tortoise scores high on action but low on adaptation and balance — it confuses process fidelity with program effectiveness.

The Wolverine takes security seriously — sometimes too seriously. It sees threats everywhere, enforces policies aggressively, and pushes back hard against anything that weakens its controls. The Wolverine's strength is genuine vigilance — it catches things others miss. Its weakness is that layers of controls increase complexity, and complexity itself becomes a source of failure (Chapter 4). Users route around Wolverine controls: shared passwords, personal devices, shadow IT. The Wolverine's security posture may be so restrictive that it creates the very vulnerabilities it was designed to prevent. A CISO should not be a Wolverine — but an incident responder probably should be. The Wolverine scores high on awareness and action but low on balance.

The Dragon has found the balance — not because it knows more than others, but because it has accepted how much it doesn't and can't know. The Dragon invests in technology but doesn't depend on it exclusively. It quantifies risk rather than relying on gut feel. It understands that security is a business function, not just an IT function. It builds programs that adapt to change because it knows the landscape will always outpace its understanding of it. The Dragon's greatest risk is complacency: the moment it stops questioning its assumptions is the moment it starts becoming a Tortoise. The Dragon scores high on all four dimensions — awareness, action, balance, and adaptation — and its defining characteristic is the discipline to act despite incomplete knowledge.

Archetypes and Security Functions

The archetypes are not personality tests — they are diagnostic tools. Different security functions benefit from different archetype characteristics:

Function	Ideal Archetype	Why
CISO / security leadership	Dragon	Must balance risk, cost, productivity, and compliance
Incident response	Wolverine	Vigilance and aggressive containment are assets during an active incident
Security architecture	Tortoise/Dragon	Systematic, technology-informed design — but must adapt to changing environments
Risk management	Dragon	Must evaluate trade-offs, not just threats
Security awareness training	Dragon/Wolverine	Must understand both the threats and the human factors that determine whether training works
Compliance	Tortoise	Process discipline and documentation are genuine strengths for audit readiness
Board oversight	Dragon	Must ask the right questions without micromanaging technical controls

An organization whose security leadership is entirely Wolverine will over-invest in controls and create friction that drives workarounds (Chapter 10). An organization whose leadership is entirely Tortoise will build a hard shell of compliance and miss the threats the framework doesn't cover. The most effective security programs have Dragons in leadership, Wolverines in incident response, and Tortoises in compliance — each archetype deployed where its strengths matter most.

The Security Animal Quiz[1] provides a self-assessment for individuals and organizations. The archetypes are referenced throughout this book: where you see a reference to a specific archetype, it is a shorthand for the behavioral pattern described here.

[1]The Security Animal Quiz (https://www.rescor.net/security-quiz/) identifies five security archetypes — Ostrich, Sloth, Tortoise, Wolverine, and Dragon — representing a progression from disengagement to balanced, adaptive security practice.

Chapter 4: Entropy and Energy

Why Risk Always Increases Without Effort

In thermodynamics, entropy is the tendency of ordered systems toward disorder. Without the continuous input of energy, every system degrades. Machines wear out. Buildings crumble. Gardens become jungles.

Information security follows the same principle. The firewall rule you wrote last year was correct then. Since then, three applications were added, two were decommissioned, and the network was restructured. The rule is still there, but the environment it was designed for is not.

Entropy in information security is the natural increase in risk over time. New assets are deployed. New applications are adopted. New employees are hired. New vendors are onboarded. New threats emerge. New vulnerabilities are discovered. Regulations change. Technology evolves. Every one of these changes introduces new risk — whether or not you are paying attention.

Energy is the money and effort you spend to counter entropy. Patching systems. Training staff. Updating policies. Testing controls. Performing risk assessments. Reviewing access privileges. Monitoring for anomalies. Responding to incidents. Every one of these activities reduces risk — but only until the next change introduces new risk that the previous effort didn't address.

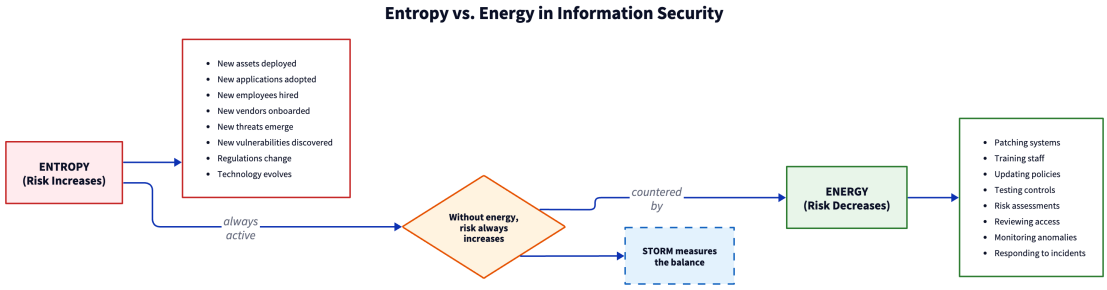

Figure 1: Entropy vs. Energy in Information Security

This dynamic has three important implications:

1. Risk management is continuous, not episodic. An annual risk assessment is a snapshot of a system in constant motion. The day after the assessment is complete, it is already stale. The threat landscape has changed, your technology has changed, your personnel have changed, your vendors have changed. Continuous risk assessment — frequent, lightweight cycles that track risk over time — is the only approach that keeps pace with entropy.

2. Standing still is moving backward. If your security program does not change, your risk increases — because everything around it is changing. The organization that maintains the same controls year after year is not maintaining the same risk level. It is accepting increasing risk, whether it realizes it or not.

3. The cost of energy must be proportionate. Not every risk requires the same level of investment. The goal is not zero risk — that is impossible and the pursuit of it creates its own risks (see Chapter 10 on optimal privilege vs least privilege). The goal is to invest energy where it reduces the most risk per dollar spent. This is the fundamental argument for quantitative risk measurement: if you can measure risk, you can direct your energy where it matters most.

STORM: Making Entropy Visible

RESCOR's Simplified Total Risk Management (STORM) methodology makes the entropy-energy dynamic visible and measurable. By quantifying risk as a percentage rather than a qualitative label ("low," "medium," "high"), STORM allows organizations to:

- **Track risk over time.** Is your energy expenditure outpacing entropy, or falling behind?
- **Compare across assessments.** Are you more or less secure than you were six months ago? Than you were a year ago?
- **Prioritize investments.** Where is risk concentrated? Where would a dollar of energy produce the most risk reduction?
- **Demonstrate to regulators and examiners** that your security program is producing measurable results — not just checking boxes.

Critically, STORM models both upside and downside risk — following the ISO 31000 definition of risk as "the effect of uncertainty on objectives" rather than the traditional CISSP definition of "the probability of loss events." This means STORM can evaluate whether a proposed security investment produces net benefit — accounting for both the risk it mitigates and the friction, cost, and productivity impact it introduces. This capability is explored in detail in Chapter 18.

The entropy-energy model also explains why security programs fail. Most failures are not caused by a single catastrophic event. They are caused by the slow accumulation of entropy — unpatched systems, unreviewed access, unupdated policies, untested backups — that eventually exceeds the organization's ability to respond. The breach that makes the news is the final straw, not the root cause. The root cause is the years of entropy that no one was measuring.

Chapter 5: The RAPID Methodology

Continuous Adaptation Over Grand Plans

The Rapid Adaptation Process for IT Governance and Deployment (RAPID) was conceived in 1992 and has been refined over three decades of practice. RAPID is characterized by successive iterations of relatively short, lightweight governance design cycles with a high level of interaction with, and feedback from, stakeholders across the organization.

RAPID was inspired by Rapid Application Development (RAD) principles in software engineering — the recognition that successive approximation and repetition produce better results than monolithic planning. In the rapidly changing environment of information security, the specifications of your security program may change during the process of building it. RAPID allows your program to adjust to these changes quickly, maintaining relevance, validity, and adaptability.

A note on history: RAPID predates the Agile Manifesto by nearly a decade. The Manifesto was published in 2001; RAPID was conceived in 1992 and first deployed in production engagements in 1993. Both arrived independently at the same fundamental insight — that iterative development with continuous feedback produces better outcomes than waterfall planning — but RAPID applied it to governance and security programs rather than software development. The principles are the same: short cycles, stakeholder engagement, continuous validation, and the acceptance that you will never produce a "final" deliverable.

Why Not Waterfall?

Traditional security program development follows a waterfall model: assess, plan, implement, maintain. Each phase is completed before the next begins. The entire process may take one to three years. By the time the program is fully implemented, the threat landscape has moved, the technology has changed, the organization has reorganized, and the program is already outdated.

RAPID rejects this model. Instead:

1. **RAPID can be performed concurrently at several governance tiers.** Strategic decisions (board and executive level) and tactical decisions (IT and operational level) can proceed in parallel, each informing the other.
2. **RAPID is self-correcting.** Decisions are evaluated by their outcomes, not by hindsight judgment of the decision-maker. An outcome that doesn't achieve its objective triggers a revision — not blame. A strategic policy that proves impractical at the operational level is revised based on observed results. An operational practice that conflicts with strategic objectives is identified and corrected. Effective outcomes propagate in both directions through governance tiers; ineffective outcomes are refined in the next cycle.
3. **RAPID never yields a "final" security program.** It yields only successive improvements to the current state. This is not a limitation — it is the point. A program designed to be "finished" is a program designed to become obsolete. A program designed for continuous improvement adapts as the environment changes.

Adaptation Over Process Fidelity

There is a disease common to organizations that adopt iterative methodologies: they turn the methodology itself into a rigid process. You see this in organizations that pursue "Scrum certification" and then enforce rigid roles, fixed sprint durations, mandatory ceremonies, and change control gates so strict that nothing can be committed without checking every process-imposed box. They have replaced one form of rigidity (waterfall) with another (process theater). The methodology that was supposed to enable adaptation has become the thing that prevents it.

RAPID is deliberately resistant to this failure mode. Its structure is minimal by design: governance tiers, stakeholders, an eight-step cycle, and outcome-based evaluation. There are no mandatory roles to certify. There are no prescribed sprint durations. There are no ceremony checklists. There is no RAPID certification — and there never will be.

This is intentional. The moment a process becomes more important than the outcome it serves, the process has become an obstacle. The purpose of RAPID is to produce a security program that works — not to produce evidence that you followed a process. If a critical vulnerability is discovered on a Tuesday and the next RAPID cycle isn't scheduled until next month, you don't wait. You act, document what you did and why, and incorporate the results into the next cycle. The process serves the outcome, not the other way around.

Organizations that confuse process fidelity with program effectiveness are Tortoises (Chapter 3) — they've built a hard shell of process and feel protected by it. But the shell is not the defense. The defense is the ability to adapt when the situation demands it. RAPID's value is not in its steps — it's in the culture of continuous adaptation that the steps are designed to build.

SGRC Teams and Governance Tiers

The RAPID process is executed by Security, Governance, Risk Management, and Compliance (SGRC) teams. Each team operates at a specific governance tier:

- **Strategic tier** — defines the security program in terms of business goals, risk appetite, and organizational objectives. Typically includes executive leadership, board representation, and compliance officers.
- **Tactical tier** — defines the technical interpretation and implementation of the strategic program. Typically includes IT leadership, security architects, and systems administrators.
- **Operational tier** — defines the day-to-day procedures and controls. Includes front-line staff, help desk, and end users who interact with the security program daily.

Most organizations will have at least two tiers (strategic and tactical). Larger organizations may have three or more. Small organizations can operate with a single team that addresses all tiers — only the granularity of decisions changes.

Each SGRC team should:

- Be small enough to convene quickly and make decisions rapidly
- Have the authority to make decisions within the scope of its tier
- Include all relevant stakeholders — not just IT and security, but also business operations, clinical staff (in healthcare), finance, legal, and compliance

The RAPID Cycle

The RAPID cycle follows eight steps, applied iteratively:

1. **Identify the assets you are trying to protect.** Assets include hardware, software, data, people, and processes. Most assets are a combination of these categories.
2. **Determine the risk associated with each asset.** How might the asset be attacked? What are the ramifications of a successful attack? Use quantitative methods (STORM) where possible — qualitative labels hide more than they reveal.
3. **Determine what level of risk is acceptable for each asset,** balancing ease of use, cost, and security. Risk appetite is a business decision, not a technical one, and it should be documented and approved by leadership.
4. **Determine what security controls are already in place.** For each asset, identify existing controls and evaluate their effectiveness. A control that exists on paper but is not enforced or tested is not a control.
5. **Identify weaknesses in the existing controls.** Where do the existing controls leave an asset at unacceptably high risk?
6. **Identify mitigating controls that will achieve the desired level of security.** Determine their cost and their impact on users and operations.
7. **Implement the mitigating controls.**
8. **Go back to step 1 and evaluate your assets in light of the new changes.** Each cycle may reveal additional cost constraints, user impact issues, or new risks introduced by the controls themselves. This is expected and normal.

This cycle should result in policy development iterations of at most a few weeks — not months or years. After an initial period of intensive activity (typically 4–12 cycles over the first year), the program settles into an ongoing maintenance rhythm where the cycle is repeated on a regular cadence — quarterly at minimum, monthly for organizations in high-risk environments.

Stakeholders

One of RAPID's core principles is that security decisions must incorporate feedback from all **stakeholders** — not just the security team. Stakeholders include:

- **Business leadership** — understands the organization's mission, risk appetite, and competitive pressures
- **Clinical or operational staff** — understands how the work actually gets done and where controls create friction
- **IT and engineering** — understands the technical constraints and implementation realities
- **Legal and compliance** — understands the regulatory requirements and liability exposure
- **Finance** — understands the budget constraints and cost-benefit trade-offs
- **End users** — understands the daily experience of interacting with security controls

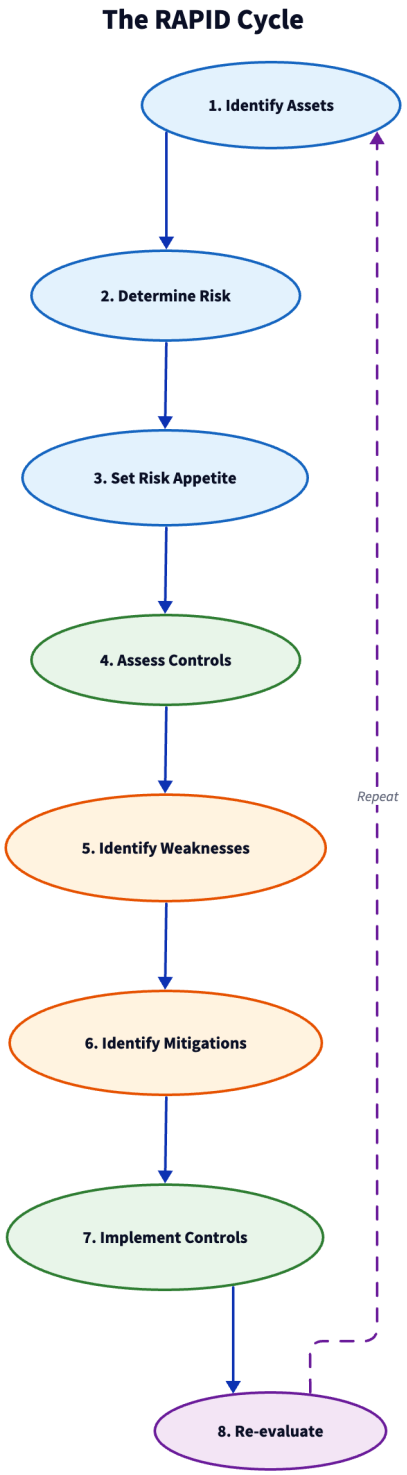

Figure 2: The RAPID Cycle

A security program designed without stakeholder input will fail. It will be too restrictive (if designed only by security), too permissive (if designed only by business), too expensive (if designed without finance), or non-compliant (if designed without legal). RAPID's iterative cycle ensures that all stakeholders are represented in every decision — and that no single perspective dominates.

Three Principles of Scalability

RAPID scales to organizations of any size through three principles:

1. **Reduction of complexity.** Break the program into manageable components that can be addressed independently. Don't try to solve everything at once. A 50-person credit union does not need the same program structure as a 50,000-employee bank — but it needs the same principles applied at an appropriate scale.
2. **Continuous improvement.** Every cycle improves the program. The first cycle will not produce a perfect program — it will produce a program that is better than what you had before. The second cycle will improve it further. Accept this and resist the temptation to wait until you have the "right" answer before acting.
3. **Stakeholder engagement.** Ensure that every decision incorporates feedback from the people who will be affected by it. Security decisions made in isolation are security decisions that will be circumvented.

RAPID and Modern Frameworks

RAPID is not a replacement for compliance frameworks — it is the process by which frameworks are implemented and maintained. RAPID is compatible with:

- **NIST Cybersecurity Framework (CSF) 2.0** — RAPID's iterative cycles align with CSF 2.0's six functions (Govern, Identify, Protect, Detect, Respond, Recover) and its emphasis on continuous improvement. The addition of Govern in CSF 2.0 reflects the same principle RAPID has embodied since 1992: governance is not separate from security operations — it drives them.
- **ISO 27001/27002** — RAPID provides the management process for implementing and maintaining the ISMS
- **ISO 31000** — RAPID implements ISO 31000's risk management principles through iterative risk assessment cycles
- **HIPAA Security Rule** — RAPID satisfies the requirement for an "accurate and thorough assessment of potential risks and vulnerabilities" (45 CFR 164.308(a)(1)(ii)(A)) through continuous rather than annual assessment
- **GLBA Safeguards Rule** — RAPID satisfies 16 CFR 314.4(b)'s requirement for ongoing risk assessment
- **NIST SP 800-37/800-53** — RAPID aligns with the Risk Management Framework's iterative approach to security control selection and assessment

The framework tells you *what* to address. RAPID tells you *how* to address it — continuously, iteratively, and with input from all the people who matter.

PART II: THE MODERN THREAT LANDSCAPE

Chapter 6: Threat Actors

Who Attacks You and Why

The original edition of this book divided threat actors into "insiders" and "Internet hackers," with insiders responsible for 70-80% of incidents. That statistic is still roughly accurate, but it is profoundly misleading if you interpret "insider" as "attacker." The majority of insider-caused losses are not attacks at all — they are mistakes made by well-meaning people operating in complex systems.

A system administrator who types `sudo rm -rf /lib` when he meant `sudo rm -rf ./lib` is not an attacker. He is a competent professional who made a one-character error in a system that offered no guardrails and no confirmation. The impact on availability is identical to a deliberate attack — the system is down, the data may be lost, the recovery clock is running — but the cause is a control failure, not a malicious act. Modeling all insider incidents as "threats" leads organizations to treat their own employees as adversaries, which creates exactly the kind of hostile, friction-heavy environment that drives the workarounds and shadow IT that cause the next incident.

Understanding who and what causes security incidents — and distinguishing between adversarial attacks and system failures — is essential for directing your defensive energy where it matters.

Insiders

An insider is anyone with legitimate access to your systems — employees, contractors, temporary staff, vendors with remote access, and business partners with API connections. Insiders remain the most common source of security incidents, but the nature of those incidents varies enormously:

- **Unintentional incidents** account for the majority of insider-caused losses. These are not attacks — they are the natural consequence of humans operating complex systems under time pressure. Misconfiguring a cloud resource. Sending sensitive data to the wrong recipient. Running a destructive command in the wrong directory. Clicking a link that looked legitimate. These incidents affect confidentiality, integrity, and availability just as effectively as deliberate attacks, but they require different controls: better defaults, confirmation prompts, reversible operations, and training — not surveillance and punishment.
- **Compromised insiders** have had their credentials or devices taken over by an external attacker. The insider may not know they are compromised. Business email compromise (BEC) attacks exploit this: the attacker uses a real employee's email account to issue fraudulent instructions that appear legitimate because they come from a trusted source.
- **Malicious insiders** deliberately abuse their access for personal gain, revenge, ideology, or coercion. A disgruntled employee who exfiltrates customer data before resigning. A system administrator who plants a logic bomb. A contractor who sells access credentials to a criminal organization. Malicious

insiders are real, but they are a small fraction of insider-caused incidents — and the controls that prevent unintentional errors (logging, access reviews, separation of duties) also detect malicious behavior.

The insider threat is amplified by the modern workplace. Remote work means employees access systems from networks you don't control. BYOD means corporate data lives on devices you don't manage. Cloud collaboration means documents are shared across organizational boundaries with a single click. AI tools mean employees paste sensitive data into third-party services that may retain it for model training.

Your Own Controls as a Threat

There is a category of availability loss that the security industry rarely acknowledges: the loss caused by your own security controls.

Availability encompasses productivity. A system that is technically "up" but takes an employee 45 minutes to authenticate into is not fully available. A customer who abandons a transaction because the fraud detection system flagged a legitimate purchase has experienced a denial of service — not from an attacker, but from you. An organization whose employees spend 1.6 hours per month fighting access controls (Ivanti, 2024) is losing the equivalent of 2.4 work-weeks per employee per year to self-inflicted availability reduction.

Excessive security controls are, in a very real sense, a denial-of-service attack perpetrated by the organization against itself. This is why the First Law says "my security is my responsibility" — not "my security is my highest priority." Security is one consideration among several, including the ability of people to accomplish useful work. The organization that optimizes for security at the expense of everything else has not eliminated risk — it has converted external risk into internal productivity loss, customer attrition, and employee frustration. The risk hasn't decreased. It has changed form. (Chapter 9's section on control friction explores this dynamic in detail, with metrics and measurement approaches.)

Organized Crime

Cybercrime is a business. Ransomware-as-a-service (RaaS) operators run affiliate programs with revenue sharing, customer support, and negotiation teams. Initial access brokers sell compromised credentials and network footholds on criminal marketplaces. Money launderers convert cryptocurrency payments into clean funds through mixing services and shell companies.

The economics are straightforward: cybercrime is profitable, the risk of prosecution is low (especially across international borders), and the barriers to entry continue to fall. A ransomware affiliate needs no technical skill — just the ability to follow instructions provided by the RaaS operator.

Organized crime targets: - **Financial data** — direct theft, wire fraud, ACH fraud - **Personal data** — identity theft, synthetic identity fraud, sale on dark web markets - **Healthcare data** — insurance fraud, prescription fraud, blackmail - **Operational disruption** — ransomware payments, extortion through threatened data publication

Nation-State Actors

Nation-state threat actors have resources, patience, and objectives that differ fundamentally from criminal organizations. They may spend months or years inside a target network without taking any visible action — conducting intelligence gathering, establishing persistence, and positioning for future operations.

Nation-state motivations include: - **Espionage** — theft of intellectual property, trade secrets, government intelligence - **Pre-positioning** — establishing access to critical infrastructure for potential future conflict - **Disruption** — degrading an adversary's economic, military, or political capabilities - **Influence** — manipulating information to affect elections, public opinion, or policy

The "harvest now, decrypt later" strategy is a nation-state concern: adversaries intercept and store encrypted traffic today with the expectation of decrypting it when quantum computing advances make cur-

rent algorithms vulnerable. Financial data, healthcare records, and government communications have value that persists for years or decades.

Most organizations believe they are not nation-state targets. Most are wrong — not because the nation-state is interested in them specifically, but because they are part of a supply chain that includes a target. The small IT services firm that manages infrastructure for a defense contractor. The healthcare billing company that processes claims for a government agency. The cloud hosting provider that serves a financial institution. The attack path goes through you to reach the actual target.

Hacktivists

Hacktivists are motivated by ideology — political, social, environmental, or religious. Their techniques range from website defacement and DDoS attacks to data theft and publication intended to embarrass or expose their targets. Hacktivist campaigns can be unpredictable in timing, target selection, and intensity.

The line between hacktivism and state-sponsored activity has blurred significantly. State actors sometimes operate under hacktivist cover, and hacktivist groups sometimes receive state support or direction without being formally affiliated.

Automated Threats

Many attacks in 2026 are not conducted by humans at all. Automated scanning, credential stuffing, vulnerability exploitation, and botnet operations run continuously against every Internet-facing system. These attacks are opportunistic — they don't target your organization specifically, but they will find and exploit any weakness you expose.

AI has accelerated automated threats. AI-generated phishing content is grammatically perfect, contextually aware, and unique for each target — defeating pattern-based detection. AI-powered reconnaissance can map an organization's external attack surface, identify employees, and craft targeted social engineering campaigns faster than any human team. AI tools available to attackers are improving on a timeline measured in months, not years.

Chapter 7: Modern Attack Patterns

How They Get In

The specific tools and techniques change constantly (Third Law), but the patterns are more stable. Understanding these patterns helps you direct your defenses at the methods that matter rather than the specific exploits of the moment.

Reconnaissance

Every targeted attack begins with intelligence gathering. In 2000, reconnaissance meant port scanning and banner grabbing — connecting to services on your network to determine what software you were running. Those techniques still work, but modern reconnaissance is far broader:

- **Open-source intelligence (OSINT)** — LinkedIn profiles reveal your employees, their roles, and your technology stack. Job postings reveal what products you use. Conference presentations reveal your architecture. GitHub repositories reveal your code. DNS records reveal your infrastructure. Certificate transparency logs reveal your domains.
- **Credential databases** — Billions of username/password combinations from previous breaches are available on criminal marketplaces. If your employees reuse passwords (and they do), their corporate credentials may already be known.

- **Social media** — Employees share information about their workplace, their projects, their travel, and their frustrations. All of it is useful to an attacker planning a social engineering campaign.
- **Cloud enumeration** — Automated tools probe cloud environments for misconfigured storage buckets, exposed APIs, and default credentials. Your cloud resources may be discovered within minutes of deployment.

Reconnaissance is often invisible to the target. Unlike the port scans of the 1990s that triggered intrusion detection alerts, modern OSINT leaves no trace on your systems because it never touches them.

Social Engineering and Phishing

Social engineering — manipulating people into performing actions or divulging information — remains the most effective attack vector. Phishing is its most common form, but the sophistication has increased dramatically:

- **Spear phishing** targets specific individuals with messages crafted using OSINT. The email references real projects, real colleagues, real events — making it nearly indistinguishable from legitimate communication.
- **Business email compromise (BEC)** uses compromised or spoofed email accounts to issue fraudulent instructions. The CFO's email account sends a wire transfer request. The vendor's email account sends updated payment instructions. The IT department's account sends a password reset link. Each appears legitimate because it comes from a trusted source.
- **AI-augmented phishing** uses generative AI to create unique, contextually appropriate messages at scale. Each email is different — defeating signature-based detection. The language is perfect — defeating "look for spelling errors" training. The content is personalized — defeating "be suspicious of generic messages" advice.
- **Voice phishing (vishing)** and **SMS phishing (smishing)** exploit channels where users have fewer defenses and less time to evaluate legitimacy. AI-generated voice cloning adds another dimension: the caller sounds exactly like your CEO because it is a synthetic reproduction of their voice.
- **MFA fatigue** attacks bombard users with multi-factor authentication prompts until they approve one to make the notifications stop. This bypasses MFA without cracking it — the user defeats their own security.

Ransomware and Data Exfiltration

Ransomware has evolved from a nuisance into the dominant criminal threat to organizations of every size. The modern ransomware attack follows a predictable pattern:

1. **Initial access** — through phishing, stolen credentials, or exploitation of a vulnerable Internet-facing system
2. **Persistence** — the attacker establishes multiple footholds to survive remediation of the initial entry point
3. **Lateral movement** — the attacker moves through the network, escalating privileges and identifying valuable targets
4. **Exfiltration** — sensitive data is copied to attacker-controlled infrastructure *before* encryption begins
5. **Backup destruction** — the attacker identifies and destroys or encrypts backup systems to prevent recovery
6. **Encryption** — production systems are encrypted, rendering them unusable
7. **Extortion** — the attacker demands payment for the decryption key and threatens to publish the exfiltrated data

The double extortion model (encrypt *and* exfiltrate) changes the calculus fundamentally. Restoring from backup addresses the encryption, but the exfiltrated data is already in the attacker's hands. For a healthcare organization, published patient records are a HIPAA breach. For a financial institution, published customer data is a regulatory catastrophe and a trust-destroying event that no incident response plan can undo.

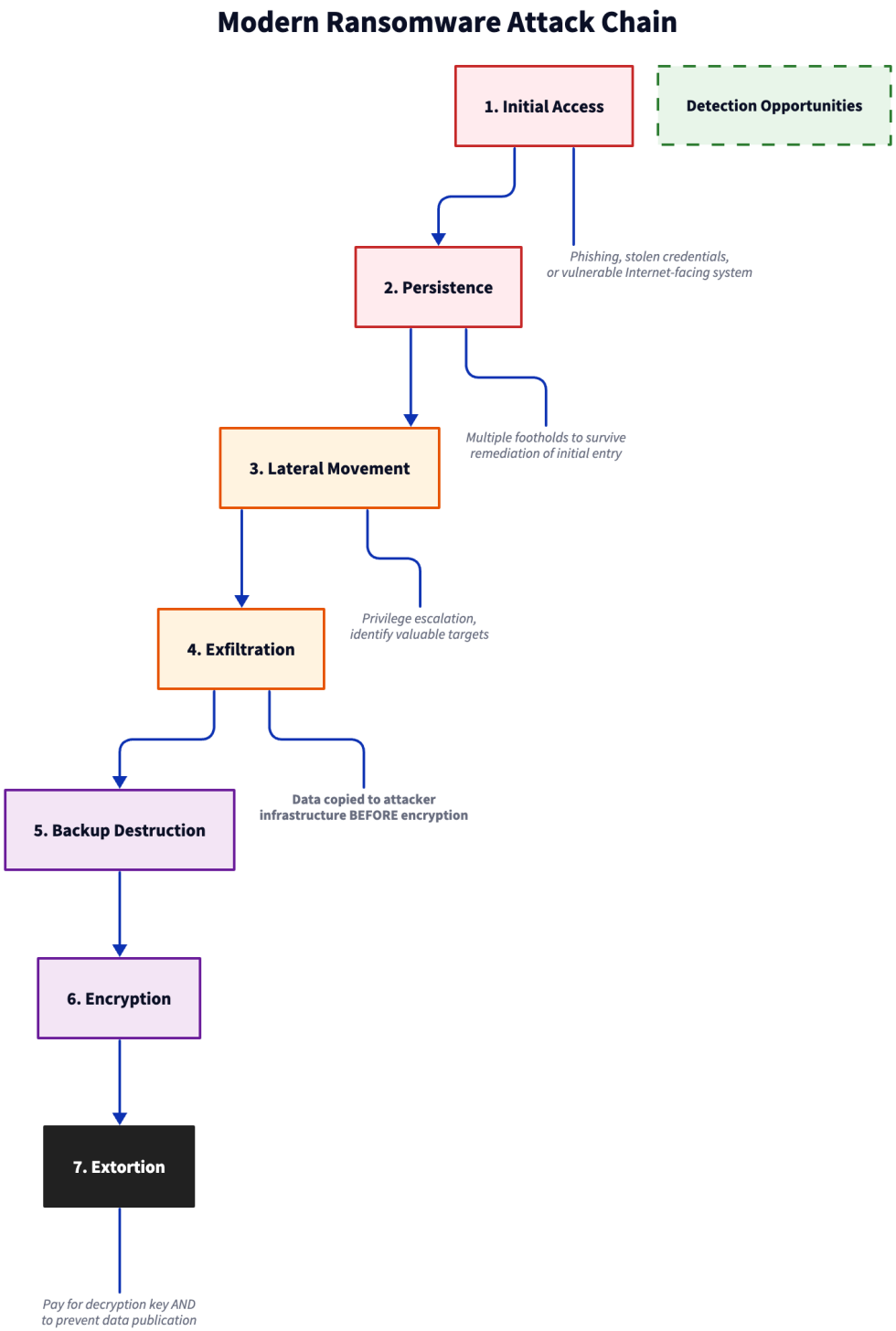

Figure 3: Modern Ransomware Attack Chain

Supply Chain Attacks

Supply chain attacks compromise a trusted vendor, software package, or service provider in order to reach the vendor's customers. The attacker doesn't need to breach your defenses — they breach your vendor's defenses, and the trust relationship you have with that vendor carries the attack into your environment.

Notable examples: - **SolarWinds (2020)** — a nation-state actor compromised the build pipeline of a widely used IT management tool, delivering malware to approximately 18,000 organizations through a routine software update - **Kaseya (2021)** — a ransomware operator exploited vulnerabilities in a managed services platform, encrypting systems at hundreds of organizations through their MSP - **MOVEit (2023)** — a zero-day vulnerability in a file transfer product was exploited to exfiltrate data from thousands of organizations - **XZ Utils (2024)** — a multi-year social engineering campaign placed a backdoor in a widely used open-source compression library, potentially affecting millions of Linux systems

Supply chain attacks are particularly dangerous because they exploit trust. The software update comes from your vendor. The API call comes from your partner. The access request comes through your MSP's credentials. Your defenses are designed to trust these sources — that's why the attacker targets them.

Identity-Based Attacks

As organizations strengthen their technical defenses, attackers increasingly target identity itself:

- **Credential stuffing** — automated testing of stolen username/password combinations against your login systems. If employees reuse passwords (and they do), this works.
- **Password spraying** — trying a small number of common passwords against a large number of accounts. This avoids account lockout thresholds while still finding weak passwords.
- **Token theft** — stealing authentication tokens or session cookies to bypass MFA entirely. The attacker doesn't need your password or your second factor — they need the token that proves you already authenticated.
- **Consent phishing** — tricking users into granting OAuth permissions to a malicious application. The user sees a familiar-looking consent prompt and clicks "Allow," granting the attacker persistent access to their email, files, or cloud resources.
- **Synthetic identity** — combining real and fabricated personal information to create entirely new identities that pass initial verification. These identities build credit history over months before "busting out" with maximum damage.

API Exploitation

APIs are the connective tissue of modern systems — and a primary target for data harvesting. Every cloud service, mobile app, and SaaS integration exposes APIs that return structured data. An attacker who compromises an API has direct, programmatic access to your data — no need to navigate a user interface or exfiltrate files manually.

Common API attack patterns include: - **Broken authentication** — APIs that don't properly validate tokens, allow token reuse, or fail to enforce authorization at the object level - **Excessive data exposure** — APIs that return more data than the client needs, relying on the client to filter. The attacker ignores the filter. - **Rate limiting failures** — APIs that allow unlimited requests, enabling mass data extraction - **Injection** — APIs that process user-supplied input without validation, enabling SQL injection, command injection, or other code execution

Many organizations rely on their cloud provider's or core banking platform's APIs without understanding the security controls in place, the data exposed, or the logging available. If your provider's API is compromised, it's your data that's exposed and your organization's name in the notification letter.

Chapter 8: What's at Stake

Consequences in the Modern Era

In 2000, the consequences of a security breach were primarily monetary and reputational. In 2026, they also include regulatory penalties, litigation, criminal liability, operational disruption, and — for some organizations — existential risk.

Regulatory Penalties

The regulatory landscape has transformed since 2000. Organizations in regulated industries face specific, enforceable requirements — and penalties for non-compliance that can dwarf the cost of the breach itself:

- **HIPAA** — The Office for Civil Rights (OCR) has imposed penalties ranging from $100,000 to $16 million for individual violations. The proposed mandatory encryption rule and 72-hour incident reporting requirement would increase both the compliance burden and the penalty exposure.
- **GLBA Safeguards Rule** — Amended in 2021 (effective 2023) to require specific technical controls including MFA, encryption, and continuous monitoring. Financial regulators can impose civil penalties, consent orders, and restrictions on business activities. For depository institutions, the consequences extend to FDIC deposit insurance assessments: a CAMELS composite rating of 1 carries an assessment rate as low as 2.5 basis points, while a CAMELS 5 rating — which a serious security failure can trigger — carries an assessment rate of up to 42 basis points. That is a 17x increase in deposit insurance costs, applied to every dollar of insured deposits. For a credit union or community bank, this alone can be a material hit to earnings.
- **State privacy laws** — Beginning with CCPA (California), dozens of states have enacted privacy legislation with varying requirements for data handling, consumer rights, breach notification, and enforcement. The patchwork creates compliance complexity that is disproportionately burdensome for small organizations.
- **SEC Incident Disclosure** — Publicly traded companies must now disclose material cybersecurity incidents within four business days of determining materiality. This applies to the incident's impact on operations, not just data loss.
- **GDPR** (for organizations handling EU residents' data) — Penalties up to 4% of global annual revenue or €20 million, whichever is greater.

Litigation

Data breach litigation has become an industry. Class action lawsuits are filed within days of breach announcements. The legal theory is straightforward: the organization had a duty to protect the data, it failed to meet that duty, and the individuals whose data was compromised suffered harm. Settlements regularly reach tens of millions of dollars for large breaches, and even small breaches can generate legal costs that exceed the direct damage.

Directors and officers face personal liability in some jurisdictions if they fail to exercise adequate oversight of cybersecurity risk. The question is no longer "was there a breach?" but "did the board take reasonable steps to prevent it?"

Operational Disruption

The operational impact of a security incident is often greater than the direct financial loss. A ransomware attack that takes systems offline for two weeks doesn't just cost the ransom — it costs two weeks of revenue, customer service disruption, employee productivity, and the long tail of recovery, remediation, and rebuilding trust.

For healthcare organizations, operational disruption can directly affect patient safety. Clinical systems offline means manual processes, delayed treatments, diverted ambulances, and postponed procedures.

For financial institutions, it means customers can't access their accounts, transactions can't be processed, and regulatory reporting can't be completed.

Reputational Damage

Trust, once lost, is expensive to rebuild and may never fully recover. Customers, patients, and members have choices. When an organization demonstrates that it can't protect their data, some of them will choose to go elsewhere — and they won't fill out an exit survey explaining why.

The reputational cost is amplified by the speed and reach of modern communication. A breach that might have been a local news story in 2000 is a global story in 2026, shared across social media within hours and referenced in search results indefinitely.

Existential Risk

For small and mid-size organizations, a major security incident can be an existential event. The combination of remediation costs, regulatory penalties, legal fees, customer loss, and operational disruption can exceed the organization's ability to absorb. Insurance helps — but cyber insurance policies have narrowed in coverage and increased in cost, and many exclude specific attack types or require specific controls as preconditions for coverage.

The question is not whether your organization can afford to invest in security. The question is whether it can afford not to.

PART III: THE DEFENSE

Chapter 9: Security Policy

The Foundation Nobody Reads

A security policy is a formal statement of the rules by which people who are given access to an organization's technology and information assets must abide. That definition hasn't changed since RFC 2196 was published in 1997. What has changed is everything else about how policies are written, maintained, and — most importantly — ignored.

The security policy is simultaneously the most important defense you have and the most neglected. It is more important than any technology you can buy, because technology implements policy. A firewall without a policy is a box with blinking lights. An identity management system without a policy is a directory. An encryption implementation without a policy is overhead. The technology does what the policy tells it to do — and if the policy is wrong, outdated, or nonexistent, the technology faithfully implements that failure.

The reason security policy is neglected is that it is hard. A complete and effective policy covers every aspect of the organization's information environment. It is a lot of work to design, a lot of work to maintain, and it requires input from stakeholders who have other priorities. Most organizations respond to this difficulty by either not writing policies at all, or writing policies that are so generic they provide no actionable guidance.

The Three Laws of Security Policy

For the purposes of this book, there are three immutable laws of security policy:

1. Relevance. Your security policy must be relevant — it must meet the needs of your users and customers while adequately protecting against current threats. This requires balancing cost, risk, security, and ease of use. A policy that is too lenient in order to reduce cost or improve ease of use may leave your systems vulnerable. A policy that is too strict in order to maximize security may reduce productivity, drive customers away, and create the workarounds that become your next vulnerability.

2. Adaptability. Your security policy must be adaptable to constant changes in user needs, technology, and threats. If your policy cannot be quickly updated to reflect these changes, it will become irrelevant and your systems will become more vulnerable. This is the Third Law applied to policy: whatever your policy says now that isn't wrong, will be tomorrow.

3. Continuous validation. Your policy must be continuously validated against actual conditions. Even if your policy is relevant now, it won't stay that way. RAPID provides the methodology for continuous validation — but the commitment to validate must be embedded in the policy itself, not left to chance.

What Policies Your Organization Needs

The specific policies your organization requires depend on your industry, your regulatory environment, your size, and your risk profile. However, most organizations need at minimum:

- **Information security policy** — the overarching policy that defines roles, responsibilities, risk assessment requirements, and the governance structure
- **Acceptable use policy** — what employees, contractors, and third parties may and may not do with organizational systems and data
- **Access control policy** — how access is granted, reviewed, modified, and revoked. This is where optimal privilege (not least privilege) is defined
- **Data classification and handling policy** — how information is categorized and what protections apply to each category. Keep classifications simple: unrestricted, restricted, and confidential is sufficient for most organizations
- **Incident response policy** — what constitutes an incident, who is notified, what actions are taken, and what reporting is required
- **Business continuity and disaster recovery policy** — how critical operations continue when systems are unavailable
- **Password and authentication policy** — length requirements (the only requirement that consistently matters), MFA requirements, and account management procedures
- **Vendor and third-party risk policy** — how vendors are evaluated, monitored, and held accountable for security
- **AI governance policy** — whether AI tools are permitted by default (PAND) or prohibited by default (DANP), and the rules for data handling, authorization, and oversight

You do not need to write these from scratch. The RAPID cycle (Chapter 5) provides the process for developing them iteratively. The first cycle will not produce perfect policies — it will produce policies that are better than what you had before. The second cycle will improve them further.

Inheriting Vendor Defaults

If you fail to design your own security policy, you will inherit whatever defaults your hardware and software vendors provide. Those defaults are designed for ease of setup, not security. Cloud platforms ship with storage publicly accessible by default. Network equipment ships with default administrator credentials. SaaS applications ship with permissive sharing settings.

A complete and effective security policy must address the default configurations of every system in your environment. This means knowing what those defaults are — which requires an asset inventory and configuration baseline before you can write the policy that governs them.

Security Awareness Training

Policy without training is decoration. The best-written security policy in the world is useless if the people subject to it don't know what it says, don't understand why it matters, or don't know what to do when they encounter a situation the policy addresses.

Most organizations approach security awareness training as a compliance obligation: an annual online module, a quiz at the end, a checkbox on the audit evidence list. This is compliance theater. It satisfies the regulatory requirement (HIPAA, GLBA, PCI DSS, and state privacy laws all require security training) while producing no measurable change in behavior — which is the only thing training is supposed to accomplish.

Effective training changes behavior. Ineffective training checks boxes. The distinction matters because the human element remains the most exploited attack vector. Phishing, business email compromise, social engineering, credential reuse, and accidental data exposure are all human-factor attacks that no technical control can fully prevent. Training is the control — but only if it works.

What effective training looks like:

- **Frequent and brief** rather than annual and comprehensive. A five-minute exercise every month produces better retention and behavior change than a sixty-minute module once a year. Spaced repetition is how humans actually learn.

- **Relevant to the audience.** Executives need to recognize board-level social engineering and CEO fraud. Clinical staff need to recognize patient data handling risks. Finance staff need to recognize invoice fraud and BEC. A single generic module for all employees treats training as a formality, not a control.
- **Behavioral, not informational.** The goal is not for employees to *know* that phishing exists. The goal is for employees to *pause before clicking*, *verify unexpected requests through a second channel*, and *report suspicious activity without fear of embarrassment*. Test this with simulated phishing — not to punish people who click, but to measure whether the training is producing the behavior you need.
- **Integrated with incident response.** Employees should know exactly what to do when they suspect something is wrong: who to contact, what to preserve, and that reporting is expected and valued. The reporting channel must be frictionless — if reporting a suspicious email requires navigating three menus and filling out a form, people won't do it.
- **Measured by outcomes, not completion rates.** A 98% completion rate on the annual training module tells you nothing about security. Phishing simulation click rates, time-to-report for suspicious activity, and help desk ticket patterns for security questions tell you whether training is working.

The organization that treats security awareness as a compliance checkbox will have a perfectly documented training program and employees who click on phishing links. The organization that treats it as a behavioral control will have employees who are genuinely harder to deceive — which is, after all, the point.

Chapter 10: Identity and Access Management

Authentication in 2026

Authentication means proving someone is who they claim to be. It is the foundation of access control, and it is the control that attackers most frequently target — because if you can defeat authentication, you don't need to defeat anything else.

Passwords: What We Got Wrong for 30 Years

The original edition of this book recommended complex passwords with mixed case, digits, special characters, and regular rotation. That advice, which was standard practice for decades and is still enforced by many organizations, is wrong. Not slightly wrong — fundamentally wrong, in ways that actively reduce security.

Length is the only password policy that consistently improves security. A 20-character passphrase composed of common English words — "correct horse battery staple," from Randall Munroe's well-known XKCD illustration — has approximately 88 bits of entropy (the exact value depends on the word list and selection method). An 8-character password with mandatory complexity ("P@ssw0rd!") has approximately 50 bits — and is far harder to remember. NIST Special Publication 800-63B (Revision 3, 2017; Revision 4 draft, 2024) explicitly recommends against complexity requirements and mandatory rotation.

Complexity rules reduce entropy by constraining the search space. When you require at least one uppercase letter, one digit, and one special character, you are telling the attacker exactly what character classes to expect. Users respond to complexity rules with predictable substitution patterns: a becomes @, o becomes 0, s becomes $, and the password starts with an uppercase letter and ends with a digit and an exclamation point. Attackers know this. Their cracking dictionaries include these patterns.

Mandatory rotation causes weaker passwords. When users are forced to change passwords every 60 or 90 days, they respond rationally: they choose passwords that are easy to increment. Password1, Password2, Password3. Or Spring2026!, Summer2026!, Fall2026!. Each rotation produces a password that is trivially derivable from the previous one. The rotation policy that was supposed to limit the window of exposure instead ensures that every password in the sequence is compromised if any one of them is.

The exposure window argument for rotation also fails on its own terms. Even with a 30-day forced rotation, a compromised credential will on average have been compromised at the midpoint of the cycle — giving the attacker 15 days of access before the password changes. Fifteen days is close enough to forever that the distinction is meaningless. A competent attacker who has 15 days of access to a system will have established persistence, escalated privileges, and exfiltrated whatever they came for long before the password rotation fires. The control that was designed to limit the attacker's play time has become entirely superfluous — while continuing to impose real costs in productivity, help desk burden, and password quality.

The honest conclusion is that credential rotation for any reason other than actual or suspected compromise is a waste of time and energy. NIST 800-63B says as much. However, many auditors and regulatory examiners have not yet caught up with the evidence, and organizations in regulated industries may need to document their rationale for eliminating rotation rather than simply removing it. The RAPID approach applies: implement what the evidence supports, document why, and be prepared to explain your reasoning to an examiner who may still be operating from a 2005 playbook.

Three-strike lockout policies don't reflect reality. Legitimate users fat-finger their passwords more than three times with reasonable frequency — especially on mobile devices, with complex passwords they can barely remember because of the complexity rules you imposed. A three-strike lockout punishes legitimate users while barely inconveniencing attackers, who use credential stuffing and password spraying at rates well below lockout thresholds.

What works instead:

- **Minimum length of 16 characters** (or longer). Encourage passphrases. Make the maximum length as long as the system supports — never truncate silently.
- **Check against breached password databases.** The HaveIBeenPwned API provides this for free. Reject passwords that appear in known breach corpuses.
- **No mandatory rotation** unless there is evidence of compromise. NIST 800-63B is explicit on this point.
- **Progressive lockout** rather than hard cutoff. Increase the delay between attempts rather than locking the account entirely.
- **Multi-factor authentication** on everything that matters. Passwords alone — no matter how long — are insufficient for systems that contain sensitive data or provide privileged access.

Multi-Factor Authentication

MFA combines something you know (password) with something you have (device, token) or something you are (biometric). It is the single most effective control against credential-based attacks. Microsoft reported that MFA blocks over 99.9% of automated credential attacks against Azure AD accounts — a widely cited statistic that, while specific to one platform, reflects the general effectiveness of MFA against automated compromise.

However, MFA is not invulnerable:

- **SMS-based MFA** is vulnerable to SIM swapping (the attacker convinces the carrier to transfer your phone number) and real-time phishing proxies (the attacker relays the OTP from victim to target system in real time)
- **Push notification MFA** is vulnerable to MFA fatigue attacks — the attacker triggers repeated prompts until the user approves one to make them stop
- **TOTP (authenticator app) MFA** is stronger but still vulnerable to real-time phishing proxies
- **FIDO2/WebAuthn (hardware keys and passkeys)** is the strongest widely deployed MFA method. It is phishing-resistant by design because the authentication is bound to the specific domain — a phishing site cannot relay it

The right MFA deployment is risk-based: FIDO2 for privileged access and high-value systems, authenticator apps for general workforce, and SMS only where nothing better is available.

"MFA everywhere, for everything, no exceptions" sounds like strong security. It is not. It is the Wolverine instinct applied without risk analysis. Consider a hospital nurse who must authenticate to a medication dispensing system twenty times per shift. Each MFA prompt takes 15-30 seconds — retrieving a phone, unlocking it, opening the authenticator app, entering the code, waiting for verification. Over a twelve-hour shift, that is 5-10 minutes lost to authentication alone — for a system that is physically inside a locked medication room accessible only to credentialed staff. The MFA is protecting against a threat that the physical controls already address, while consuming clinical time that directly affects patient care.

Now consider the same hospital's VPN gateway, where a physician connects from a hotel room to access patient records. Here, MFA is essential — the physician is authenticating over an untrusted network to access sensitive data, and the credential alone is insufficient because it may have been phished, stuffed, or stolen. The risk justifies the friction.

The difference between these two scenarios is not the value of the data — it's the threat model. The medication system faces a physical access threat that MFA doesn't address. The VPN faces a remote credential threat that MFA directly addresses. Applying MFA to both without distinguishing between them doesn't improve security — it wastes clinical time on one system while correctly protecting the other.

Risk-based deployment is the Dragon approach. MFA everywhere is the Wolverine approach — fierce, well-intentioned, and counterproductive where the threat model doesn't support it.

MFA Recovery: The Weakest Link

MFA protects against stolen credentials. But what happens when a legitimate user loses access to their MFA device — a broken phone, a lost hardware key, a factory reset that wipes the authenticator app? The recovery mechanism must be at least as strong as the thing it recovers. If it isn't, the recovery path becomes the attack path.

The paradox. If a password reset bypasses MFA, then MFA is theater — the attacker resets the password and walks in. If an MFA reset bypasses the password, same problem in reverse. The authentication chain is only as strong as its weakest recovery mechanism, and for most organizations, the recovery mechanism *is* the weakest link.

The three-way dependency. Authentication rests on three factors: something you know (password), something you have (MFA device), and something you are (verifiable identity). Recovery should require at least two of the three:

- **Lost password, MFA device survives:** Verify identity through MFA, reset the password. Two factors survive — this works.
- **Lost MFA device, password survives:** The password alone is the thing MFA was supposed to supplement. Allowing the password alone to reset MFA makes MFA optional. This scenario requires the password *plus* an independent identity verification — a backup MFA method, a recovery code, or a human recovery process.
- **Both lost:** Identity proofing from scratch. This is expensive and, in virtual-only relationships where identity was never strongly verified at enrollment, may be nearly impossible. But it must exist — permanent lockout of a legitimate user is a denial-of-service attack against your own customer.

The practical solution: multiple enrolled MFA methods. Users should be required — not merely allowed — to enroll multiple MFA methods at setup:

- **Primary method** — authenticator app or FIDO2 key
- **Backup phone OTP** — to one or more phone numbers, including trusted family members. The objection that "the family member isn't the account holder" is a policy choice, not a security limitation. A designated recovery contact who can receive an OTP is stronger than a security question whose answer is on Facebook.
- **Printed recovery codes** — one-time codes generated at enrollment, stored offline (not on the same device as the authenticator app)
- **Backup email OTP** — to a pre-verified email address

It is possible for someone to lose their phone, their backup phone, their recovery codes, and their email access simultaneously. It is not common. Multiple enrolled methods make the catastrophic "everything lost" scenario rare enough that the expensive human recovery path is affordable rather than routine.

The customer service imperative. Organizations that provide no human recovery path have decided that permanent lockout is an acceptable outcome for legitimate users. For a cryptocurrency wallet, that may be the right trade-off. For a bank, a healthcare portal, or any service where the person's livelihood or health depends on access, it is not. A human agent who can verify identity through an independent channel (video call, in-person visit, knowledge verification with information not available on social media) must be available as a last-resort recovery mechanism.

This connects to the friction argument: an organization that makes account recovery impossible has not improved security. It has created a control that, for the locked-out legitimate user, is indistinguishable from an attack. The security program must account for the entire lifecycle of authentication — not just the login, but the enrollment, the recovery, and the experience of the person who depends on it.

Password and MFA resets must be coupled. A password reset without the option to reset MFA leaves a user who has lost both in an unrecoverable state. A system that allows password reset through email but requires the (now lost) MFA device for the MFA reset has created a one-way trap. If the recovery mechanism is strong enough to trust for one factor, it is strong enough to trust for both. Reset processes should offer the option to reset both factors simultaneously, with the same identity verification applied to both.

Control Friction

Control friction is the cumulative productivity cost of security controls on the people who must interact with them. Every authentication prompt, every access request, every approval workflow, every training module, every policy acknowledgment imposes a cost — measured in time, attention, and patience. That cost is real and measurable, but most organizations do not measure it.

The security industry treats friction as an acceptable side effect: "users must accept some inconvenience as the cost of a secure environment." That premise is wrong. Users must accept *proportionate* friction — friction that is justified by the risk it mitigates. Disproportionate friction is not security. It is waste.

Friction has four consequences:

1. **Productivity loss.** The direct cost of time spent on security interactions rather than productive work. A StrongDM survey found that 64% of organizations experience daily or weekly productivity impacts from access issues. Ivanti found that access friction costs an average of 1.6 hours per employee per month. For a 2,000-person organization at $100/hour loaded cost, that is approximately $3.8 million per year in lost productivity.
2. **Workarounds.** When friction exceeds a user's tolerance, they route around the control. Shared passwords. Personal email for work documents. Unauthorized cloud services. USB drives. AI tools without governance. Ivanti found that 61% of employees use unsafe workarounds when frustrated by security controls. The workaround is almost always less secure than the control it bypasses — meaning the friction didn't just fail to improve security, it actively degraded it.
3. **Customer attrition.** Friction applied to customer-facing processes drives customers to competitors with smoother experiences. The fraud detection system that blocks 2% of legitimate transactions. The onboarding process that takes three days and a notarized form. The support system that requires re-authentication at every handoff. These are security controls that directly reduce revenue.
4. **Alert fatigue and desensitization.** When users are subjected to constant security prompts, warnings, and approvals, they stop reading them. The MFA prompt becomes muscle memory — approve without looking. The security warning becomes background noise — click through without reading. The policy acknowledgment becomes an annual chore — scroll to the bottom and sign. The controls exist, but the human attention that makes them effective has been exhausted.

Measuring friction:

Friction should be measured and tracked as a security metric, alongside traditional metrics like vulnerability counts and patch compliance:

- **Access request volume and time-to-resolution** — high volume or long resolution times indicate excessive friction
- **Help desk tickets for authentication issues** — a leading indicator of password and MFA friction
- **Workaround frequency** — shared credentials, unauthorized tools, shadow IT usage
- **Customer abandonment rates** — transactions abandoned, onboarding dropped, support interactions escalated
- **MFA prompt frequency per user per day** — a direct measure of authentication friction

Session duration: the hidden friction multiplier.

If you implement MFA, extend session durations accordingly. This is the single most impactful friction reduction available to most organizations, and it costs nothing.

A 30-minute session timeout with MFA reauthentication means a user authenticates not once per day but 16 times per day. At 30-60 seconds per authentication (retrieve phone, unlock, open authenticator, enter code, wait for verification), that is 8-16 minutes per day per user lost to proving — repeatedly, from the same device, from the same browser, from the same network — that they are the same person who authenticated 29 minutes ago.

There is no meaningful security improvement in reauthenticating a user whose device fingerprint, browser fingerprint, network location, and behavioral pattern have not changed since the last authentication. The threat that MFA addresses — a stolen credential used from an unknown device — does not recur every 30 minutes. If the session is compromised, the attacker has the session token and the reauthentication doesn't help anyway.

Session durations of 12 hours are appropriate for most business applications. For low-sensitivity internal applications, 24 hours or longer is defensible. For high-sensitivity applications, shorten the session but combine it with continuous signals — device health, network location, behavioral anomalies — rather than forcing reauthentication on a timer. The reauthentication should be triggered by a change in risk context, not by the passage of time.

The threat that short session timeouts are actually trying to address is physical — someone sitting down at an unattended workstation and using an active session. But this is the same failed logic as password rotation: limiting the exposure window for a compromised session. And the math is the same: a 30-minute timeout provides an average 15-minute exposure window, which is more than enough for a malicious actor at an unattended machine. The correct control for the unattended workstation threat is a **screen lock** — 5 minutes of inactivity, unlock with biometric or PIN. The screen lock addresses the physical access threat directly, costs the returning user 2 seconds to unlock, and does not force reauthentication for the user who never left.

Short session timeouts solve the wrong problem with the wrong control at the wrong cost. The organizations that set them are almost always doing so because a compliance checklist said to, or because the default was short and nobody questioned it. Neither is a risk-based decision. Both are pure friction with no corresponding security benefit.

The friction test:

For every security control, ask: *Does the risk this control mitigates exceed the cost this control imposes?* If you can't answer that question quantitatively, you are operating on faith, not risk management. STORM quantitative risk measurement (Chapter 18) provides a way to answer it — measuring both the risk and the cost in the same terms so they can be meaningfully compared.

A control that is bypassed 30% of the time is not providing 70% of its intended protection. It is providing some fraction of that — diminished by the false sense of security it creates and the workaround risk it generates. A slightly less restrictive control that is followed 100% of the time almost always produces a

better security outcome. This is the Dragon's insight: the optimal control is not the strongest control. It is the one that produces the best outcome when you account for how humans actually interact with it.

Optimal Privilege

The traditional "principle of least privilege" — reducing every user's access to the absolute minimum required for their current task — sounds like good security. In practice, as described in detail in our companion article *Least Privilege Can Be Poor Practice*, it produces role explosion, audit failure, productivity loss, and security workarounds that are worse than the risks it was supposed to mitigate.

The principle of **optimal privilege** replaces least privilege with a practical alternative: the smallest number of well-defined roles that let your organization function effectively while remaining auditable and monitorable. In practice, this means:

- **Consolidate roles aggressively.** Aim for tens of roles, not hundreds or thousands.
- **Accept that most roles will have more privileges than strictly necessary.** This is not a deficiency — it is a deliberate trade-off between access control granularity and operational reality.
- **Monitor roles rather than restrict them.** A smaller number of well-understood roles is far easier to monitor for anomalous behavior than thousands of fine-grained roles that no one can explain.
- **Measure the cost of friction.** Access request volume, time-to-resolution, and workaround frequency are security metrics. If they are high, your access control model is working against you.

Passkeys and the Future of Authentication

Passkeys (FIDO2 credentials stored on devices) represent the most significant shift in authentication since passwords were invented. A passkey is a cryptographic credential that is:

- **Phishing-resistant** — bound to the specific site, cannot be relayed
- **Nothing to remember** — no password to forget, rotate, or write on a sticky note
- **Nothing to type** — authenticated by biometric or device PIN
- **Unique per site** — a compromised passkey for one site cannot be used anywhere else

Passkeys are not yet universally supported, but adoption is accelerating across major platforms and services. Organizations should plan for a transition from passwords + MFA to passkeys as the primary authentication method, with passwords as a fallback for systems that don't yet support passkeys.

Your Phone Is the Keys to the Kingdom

Your mobile phone has become your most critical authentication asset. It receives SMS codes, push notifications, and voice calls. It runs your authenticator app. It stores your passkeys. It may be the only device that can unlock your most sensitive accounts.

This concentration of trust in a single device creates a fragility that most people don't appreciate until it fails. A broken screen, a lost phone, a factory reset that wipes the authenticator app, or even a routine upgrade to a new model can sever access to dozens or hundreds of accounts simultaneously. If an MFA system has fingerprinted the device, a change in fingerprint — even to a legitimate new phone — rightly represents a risk escalation that triggers additional verification. Biometric enrollments (fingerprint, face) do not survive a phone upgrade — you have to re-register them, which requires authenticating through the very mechanisms that depend on the biometrics you can no longer use.

The loss of a phone without a PIN or biometric lock is catastrophic. The attacker has your authenticator app, your push notification channel, your SMS codes, and potentially your passkeys. They can approve their own MFA prompts, receive their own recovery codes, and reset passwords at will.

Mitigations:

- **Enroll multiple MFA methods** before you need them (Chapter 10, MFA Recovery). A backup phone number, printed recovery codes, and a secondary authenticator on a different device ensure that losing one device does not lock you out of everything.
- **Use a PIN or biometric lock** on your phone. Always. A phone without a lock screen is an unlocked vault.
- **Export authenticator data before upgrading.** Most authenticator apps now support cloud backup or export. Google Authenticator, Microsoft Authenticator, and Authy all offer mechanisms to transfer to a new device — but only if you set them up before the old device is gone.
- **Treat phone loss as a security incident.** Report it immediately. Revoke active sessions. Change passwords on critical accounts. The window between phone loss and attacker exploitation may be minutes, not hours.

Organizations should recognize the phone as a critical asset in their security program — not just as an endpoint to manage, but as the single point of failure for authentication across the entire identity infrastructure.

Chapter 11: Network Security in a Post-Perimeter World

The Death of the DMZ

The original edition devoted significant space to the DMZ — the demilitarized zone between your protected network and the Internet. Firewalls sat at the boundary, controlling traffic between zones. The protected network was trusted; the Internet was untrusted; the DMZ held the systems that needed to be accessible from both.

That model is dead — but the perimeter hasn't entirely dissolved.

In 2026, your employees work from home, coffee shops, and airports. Your applications run in AWS, Azure, and GCP. Your data lives in SaaS platforms you don't control. Your vendors connect to your systems through APIs. Your customers access your services through mobile apps. The clear boundary between "inside" and "outside" no longer defines where your data lives or where your users work.

But almost every organization still has an internal network that is firewalled from the Internet and from other uncontrolled external access. The internal network is not the repository for most of the organization's information assets — those have migrated to cloud services — but it remains a real environment where transitive trust can be granted with lower risk than trust extended from the outside. If you accept that the purpose of information systems is not to be secure but to do useful work, then some form of transitive trust cannot be jettisoned with the DMZ.

The network behind a firewall has its attack surface decreased by the mitigating efficacy of the firewall. An access request originating from the internal network represents a lower risk — not zero risk, but lower — than one arriving from the Internet or even from a VPN. Risk-based authentication can and should consider network location as a factor: MFA may be essential for the VPN connection but unnecessary for a workstation on a physically secured, firewalled internal segment.

Zero trust does not mean "distrust everything equally." It means "verify proportionally to the risk." The internal network reduces some risk. It does not eliminate it — and it certainly does not protect cloud-hosted resources that the internal network never touches.

Zero Trust

Zero-trust architecture replaces the perimeter model with a principle: **never trust, always verify.** Every access request is evaluated based on the identity of the requester, the health of their device, the sensitivity of the resource, and the context of the request — regardless of whether the request originates from "inside" or "outside" the network.

Zero trust is not a product you buy. It is an architectural approach implemented through:

- **Identity-centric access control** — every access decision is based on authenticated identity, not network location
- **Microsegmentation** — network segments are isolated so that compromise of one segment does not automatically grant access to others
- **Continuous verification** — authentication and authorization are re-evaluated throughout the session, not just at login
- **Least-function access** — users and systems are granted access to specific resources for specific purposes, not to entire network segments
- **Assume breach** — the architecture assumes that any component may already be compromised and limits the blast radius accordingly

Zero trust is the Second Law ("I will be hacked") implemented as architecture.

Network Segmentation

Even in a zero-trust model, network segmentation remains valuable. Segmentation limits lateral movement — the attacker who compromises one system cannot automatically reach every other system on the network.

Effective segmentation separates: - **Production systems** from development and test environments - **Clinical or operational systems** from administrative systems (critical in healthcare) - **End-of-life systems** on isolated VLANs where they cannot be reached from production networks - **IoT and OT devices** from general-purpose computing environments - **Guest and visitor networks** from corporate systems

Segmentation is not a one-time exercise. It must be reviewed whenever the network changes — and in a cloud-native environment, the network changes constantly.

Cloud Security

Cloud security is not a separate discipline — it is information security applied to a different deployment model. The fundamental principles (policy, access control, monitoring, incident response) are identical. What differs is the implementation and the shared responsibility model.

In every cloud platform (AWS, Azure, GCP), the provider secures the infrastructure and the customer secures the configuration, the data, and the access controls. Most cloud security failures are customer failures — not because the cloud is insecure, but because the customer misconfigured it.

Common cloud security failures include: - **Storage exposed to the Internet** — S3 buckets, Azure blob containers, and GCP storage buckets with public access enabled - **Excessive IAM permissions** — service accounts and roles with far more access than necessary (a 2023 Palo Alto Unit 42 study found that 99% of cloud identities had excessive permissions) - **Missing encryption** — data at rest and in transit not encrypted when it should be - **Insufficient logging** — cloud audit trails disabled or not monitored - **Default configurations** — security groups, network ACLs, and service settings left at permissive defaults

Cloud security requires the same RAPID approach as any other aspect of your security program: iterative assessment, stakeholder engagement, and continuous validation.

Chapter 12: Endpoint and Application Security

The Endpoint Is the New Perimeter

If the network perimeter has dissolved, the endpoint — the device where users interact with systems and data — has become the front line of defense. Laptops, desktops, mobile devices, and increasingly IoT devices are where attacks land, where data is accessed, and where controls are either enforced or bypassed.

Endpoint Detection and Response (EDR/XDR)

Traditional antivirus — signature-based detection that compared files against a database of known malware — was adequate when threats changed slowly. It is not adequate in 2026, when attackers use polymorphic malware, fileless techniques, living-off-the-land binaries, and AI-generated payloads that have never been seen before.

EDR (Endpoint Detection and Response) and its evolution XDR (Extended Detection and Response) provide: - **Behavioral analysis** — detecting suspicious patterns of activity rather than known signatures - **Real-time monitoring** — continuous visibility into endpoint activity - **Automated response** — isolating compromised endpoints, killing malicious processes, rolling back changes - **Forensic data** — detailed telemetry for incident investigation

EDR/XDR is not optional for organizations that take security seriously. It is the detection capability that the Second Law demands.

Patch Management

Unpatched systems remain one of the most common attack vectors — not because patching is difficult, but because it is unglamorous, repetitive, and easy to defer. Every deferred patch is entropy: risk increasing because you didn't spend the energy to address it.

Effective patch management requires: - **Inventory** — you cannot patch what you don't know you have - **Prioritization** — not all patches are equal. Critical vulnerabilities in Internet-facing systems take precedence over cosmetic updates to internal tools - **Testing** — patches can break things. Test before deploying to production, but don't let testing become an excuse for indefinite delay - **Automation** — manual patching doesn't scale. Automate everything you can, and track what you can't

Application Security

Applications — web applications, APIs, mobile apps, and internal tools — are where your data lives and where your users interact with it. Application security is not an afterthought bolted on after development; it is a discipline integrated into the software development lifecycle.

The OWASP Top 10 provides a useful starting point for understanding common application vulnerabilities: injection, broken authentication, sensitive data exposure, broken access control, security misconfiguration, and others. These vulnerabilities have been known for decades and continue to appear because developers are not trained to avoid them and organizations do not test for them.

Secure development practices include: - **Security training for developers** — not annual compliance training, but practical instruction on secure coding patterns - **Static and dynamic analysis** — automated tools that identify vulnerabilities in code before and during execution - **Dependency management** — knowing what third-party libraries your applications use and monitoring them for known vulnerabilities - **API security** — authentication, authorization, rate limiting, input validation, and logging for every API endpoint

---

Chapter 13: Data Protection

Classify, Encrypt, Protect, Recover

Data is what attackers ultimately want. Systems, networks, and applications are means to an end — the end is your data. Data protection is therefore the last line of defense: even if every other control fails, strong data protection limits the damage.

Data Classification

Classification determines what protections apply to what data. The most common mistake is creating too many classification levels. A military-grade classification system (Unclassified, Confidential, Secret, Top Secret, plus compartments) is appropriate for national defense. It is not appropriate for a 200-person healthcare organization.

For most organizations, three levels are sufficient:

- **Unrestricted** — information that may be released to the general public. Marketing materials, public-facing website content, published policies.
- **Restricted** — information for internal use only. Employee records, internal communications, operational procedures, financial reports.
- **Confidential** — information that would cause material harm if disclosed. Patient health information (PHI), customer financial data, trade secrets, credentials, encryption keys.

The value of classification is not in the labels — it's in the handling rules attached to each label. Confidential data must be encrypted at rest and in transit, access must be logged, and transmission to external parties must be controlled. Restricted data requires access controls but may not require encryption at rest. Unrestricted data requires no special handling.

The enforcement problem. Classification without mandatory access control is theater at worst, kludgy at best. The military classification system works because it is enforced by hardware and software that physically prevents unauthorized access to classified material — mandatory access control (MAC) in the operating system sense, not the network sense. In the private sector, there are very few good mechanisms to label information and enforce those labels consistently. Data Loss Prevention (DLP) tools detect sensitive data attempting to leave the organization, but this is catching the horse after it has left the barn. Microsoft Information Protection (MIP) and Azure Rights Management (RMS) provide labeling and encryption, but in practice they reduce to "authenticated users have access" combined with groups or roles — which is access control, not classification enforcement.

New information is particularly hard to label. When an employee creates a document, drafts an email, or enters data into a system, there is no natural workflow that prompts classification. Try to enforce a policy that all public information must be labeled "PUBLIC" and anything not so labeled is considered restricted — it sounds sensible in policy, but in practice nobody labels anything, so everything defaults to restricted, which means the classification system conveys no information at all.

The pragmatic approach is to classify *repositories* rather than *documents*: the SharePoint site is restricted, the public website is unrestricted, the EHR is confidential. Access controls enforce the classification at the container level. Individual document labeling is useful where DLP can act on it, but it should not be the primary enforcement mechanism — because in practice, it won't be used consistently enough to rely on.

Encryption

Encryption protects data confidentiality — ensuring that even if data is intercepted or stolen, it cannot be read without the decryption key.

Encryption in transit protects data moving between systems. TLS (the successor to SSL described in the original edition) is now ubiquitous — virtually all web traffic, email between servers, and API communications are encrypted in transit. This is one area where the threat landscape has genuinely improved since 2000, when most Internet traffic was plaintext.

Encryption at rest protects data stored on disks, in databases, and in cloud storage. It is primarily a defense against physical theft of media and unauthorized access to storage. A critical nuance: encryption at rest protects against exactly one threat vector. Any application or user with legitimate access receives cleartext data — the encryption is transparent to authorized access. An encrypted database accessible from the Internet is not meaningfully more secure than an unencrypted database accessible from the Internet. The encryption protects against someone stealing the disk, not against someone querying the database through the application.

The cost of encryption at rest is not zero. Performance overhead from whole-device or whole-disk encryption ranges from negligible (1-3% for modern hardware-accelerated AES on SSDs with built-in encryption controllers) to substantial (up to 30-80% for software-based encryption like LUKS on high-throughput database servers). For a laptop or workstation, the overhead is invisible. For a database server processing thousands of transactions per second, it can be a material performance degradation — and performance degradation is itself an availability risk. The argument for encryption at rest is strong for portable devices that are easily lost or stolen. The argument is weaker for servers in physically secured data centers where the threat model (physical theft of media) is already addressed by physical controls.

This is why the proposed HIPAA mandatory encryption rule is problematic (see Chapter 19): it treats encryption as a universal control rather than a risk-based one, and in doing so eliminates the risk analysis that determines where encryption actually matters.

Post-quantum considerations. Current encryption algorithms (RSA, elliptic curve) rely on mathematical problems that are believed to be intractable for classical computers but may be solvable by sufficiently powerful quantum computers. The "harvest now, decrypt later" strategy (Chapter 6) makes this a present concern, not a future one. Organizations should be aware of the NIST post-quantum cryptography standards and begin planning for migration — not because quantum computers can break your encryption today, but because the data you encrypt today may still be sensitive when they can.

Backup and Recovery

Backups are not a security control — they are a survival mechanism. A backup you have never restored is not a backup; it is a hypothesis. A backup stored on the same network as your production systems is not a backup; it is an additional copy that will be encrypted alongside everything else when ransomware hits.

Effective backup strategy includes: - **Regular, automated backups** to a location that is not accessible from the production network - **Immutable backups** that cannot be modified or deleted for a defined retention period - **Regular restoration testing** — not annual, not when you remember, but on a defined schedule with documented results - **Defined recovery time objectives (RTO)** and **recovery point objectives (RPO)** — how fast you need to recover and how much data loss you can tolerate - **Offline or air-gapped copies** for the most critical data

The ransomware attack model specifically targets backups (Chapter 7). If the attacker can reach your backups, they will encrypt or destroy them before encrypting your production systems. Your backup architecture must assume that the production network is compromised.

Write once, read never (without special intervention). The principle behind immutable backups is simple: once written, a backup must never be accessible for modification or deletion without extraordinary human intervention. This is the only way to guarantee a non-compromised recovery point when the production environment — including the backup management system itself — has been compromised by ransomware or a malicious insider.

Mechanisms for achieving immutability:

- **Object lock / WORM storage.** Cloud providers (AWS S3 Object Lock, Azure Immutable Blob Storage, GCP Bucket Lock) support write-once-read-many (WORM) policies that prevent deletion or modification for a defined retention period. Even an administrator with full access to the cloud account cannot delete a WORM-locked object until the retention period expires. This is the strongest widely available mechanism.

- **Air-gapped copies.** Backups written to media that is physically disconnected from any network after the backup completes. Tape stored in a vault. External drives locked in a safe. The backup is inaccessible to any network-based attacker — including ransomware, compromised administrators, and compromised backup software.
- **Separate administrative domains.** Backup infrastructure managed by different credentials, in a different account or tenant, with no trust relationship to the production environment. The attacker who compromises the production domain controller has no path to the backup system because the backup system does not trust the production domain.
- **Vendor-managed immutability.** Some backup vendors (Veeam, Rubrik, Cohesity) implement their own immutability layers that prevent deletion through the backup management interface. These are useful but only as strong as the vendor's implementation — verify that the immutability cannot be bypassed by an administrator who has compromised the backup server itself.

The common failure mode is backups that are technically "immutable" but administratively accessible. If the same credentials that manage your production environment also manage your backup retention policies, an attacker who compromises those credentials can modify the retention policy before deleting the backups. True immutability requires separation of administrative control — the person or system that writes the backup must not be the same person or system that can delete it.

PART IV: DETECTION, RESPONSE, AND RECOVERY

Chapter 14: Detection and Monitoring

Seeing What Matters

The Second Law tells us that prevention will fail. Detection is what determines whether that failure is a contained incident or a catastrophe. The difference between an organization that discovers a breach in hours and one that discovers it in months is not technology — it is attention.

Three Types of Controls — and Why the Balance Is Wrong

Security controls fall into three categories:

- **Preventive controls** stop incidents from occurring. Firewalls, access controls, encryption, MFA, input validation — the controls that most of this book describes, and where most organizations concentrate their spending.
- **Detective controls** identify incidents that preventive controls failed to stop. Log monitoring, SIEM, EDR/XDR, anomaly detection, threat intelligence — and, critically, trained humans who notice when something is wrong.
- **Corrective controls** limit damage and restore operations after an incident is detected. Incident response plans, backup and recovery procedures, business continuity plans, forensic capabilities.

All three are necessary. The problem is that most organizations invest disproportionately in prevention and treat detection and correction as afterthoughts. This feels rational — preventing an incident is better than responding to one. But the Second Law tells us that prevention *will* fail. When it does, the organization with strong detective and corrective capabilities contains the breach in days. The one without them discovers it months later — often when a regulator or a journalist calls to tell them.

IBM's 2024 Cost of a Data Breach Report quantifies the gap: organizations with mature detection and response capabilities contained breaches in an average of 168 days; those without, 292 days. That 124-day difference is not a rounding error — it is the difference between a manageable incident and a catastrophe. No amount of additional preventive spending can close that gap once the attacker is inside.

The balance should be deliberate, not accidental. Measure your mean time to detect (MTTD) and mean time to respond (MTTR) alongside your preventive control coverage. If your MTTD is measured in months, your preventive investment is not compensating — it is creating a false sense of security that delays the detection your organization desperately needs.

The Human Element

The original edition of this book observed that "a diligent system administrator reviewing logs catches things that automated systems miss." That remains true in 2026, despite the enormous advances in security

technology. SIEM platforms, EDR/XDR, threat intelligence feeds, and AI-powered anomaly detection are all valuable — but they generate alerts. Alerts require humans to evaluate context, make judgment calls, and decide on action.

The failure mode is not insufficient technology. It is alert fatigue — the same desensitization described in the friction section of Chapter 10. An organization that generates 10,000 alerts per day and has one analyst to review them has not implemented detection. It has implemented noise. The analyst will triage by severity level, ignore everything below "critical," and eventually miss the critical alert buried in the flood.

Effective detection requires:

- **Tuning** — reducing false positives until the alert volume is manageable and each alert represents a condition worth investigating
- **Context** — enriching alerts with information that helps the analyst understand what happened, not just that something happened
- **Prioritization** — distinguishing between alerts that require immediate action and those that can wait for the next business day
- **Staffing** — having enough trained people to actually review alerts. Technology without staff is decoration.

Your Workforce as a Detection Layer

Technical detection tools — SIEM, EDR, anomaly detection — have a ceiling. They detect patterns they are configured to detect. They do not detect the CFO's email account sending wire transfer instructions at 2 AM, or the "IT help desk" caller who doesn't sound like anyone in IT, or the vendor invoice with subtly changed bank routing numbers. Humans detect these things — if they are trained to look, empowered to act, and confident that reporting will be taken seriously.

Your workforce is your most scalable detection capability. Every employee who interacts with email, handles financial transactions, accesses sensitive data, or talks to vendors is a potential sensor. The organization that treats employees as liabilities to be controlled — subjecting them to surveillance, restricting their access, and punishing their mistakes — is discarding this capability. The organization that treats them as a force multiplier — training them on the specific threats relevant to their role, making reporting frictionless, and responding visibly when they report — has a detection network that no technology stack can replicate.

This is the connection between Chapter 9's security awareness training and this chapter's detection capability. Training that changes behavior (Chapter 9) produces employees who detect threats (this chapter). Training that checks a compliance box produces employees who click through phishing simulations the same way they click through every other security prompt — without thinking. The investment in awareness is an investment in detection. They are the same budget line, whether your organization recognizes it or not.

Log Management

Logging is the foundation of detection. If an event is not logged, it did not happen — as far as your security program is concerned.

Effective log management follows principles that have not changed since the original edition:

1. **Log everything.** Disk space is cheap. Turn on all logging facilities and log all levels of messages. An informational message announcing an interactive login may be the indicator of an attack.
2. **Synchronize clocks.** Use NTP to synchronize the clocks on all systems participating in the audit trail. Without synchronized clocks, you cannot correlate events across systems — and a few seconds of difference can make correlation impossible.
3. **Centralize logs.** Forward logs from all systems to a dedicated log management platform (SIEM). The logging server should have restricted access so that even if an attacker compromises a system, the log entries recorded before the compromise are preserved.

4. **Retain logs.** Keep at least 90 days of searchable logs online and at least one year in archive. Regulatory requirements may mandate longer retention.
5. **Review logs.** This is the step most organizations skip. Collecting logs you never review is expensive storage, not detection. Define what you are looking for, assign responsibility for review, and hold people accountable for doing it.

SIEM and Threat Intelligence

A Security Information and Event Management (SIEM) platform aggregates logs from across your environment, normalizes them into a common format, correlates events across sources, and applies rules and analytics to identify potential security incidents.

Modern SIEMs also incorporate threat intelligence — data feeds from external sources that identify known malicious IP addresses, domains, file hashes, and behavioral indicators. Threat intelligence enriches your detection by connecting your internal events to known external threats.

The SIEM is only as effective as its configuration and the people who operate it. An out-of-the-box SIEM with default rules and no customization will generate a flood of irrelevant alerts while missing the specific threats that matter in your environment. Tuning a SIEM for your environment is an ongoing process — not a one-time setup.

Chapter 15: Incident Response

When Prevention Fails

Incident response is what happens when detection identifies something wrong. The four-phase model described in the original edition — and in our companion article *A Primer on Incident Response* — remains the foundation:

1. **Detection** — identifying that an incident has occurred or is in progress
2. **Response** — assessing the scope, containing the damage, and preserving evidence
3. **Recovery** — restoring affected systems and services to normal operation
4. **Analysis** — reviewing what happened, what worked, what didn't, and what to change

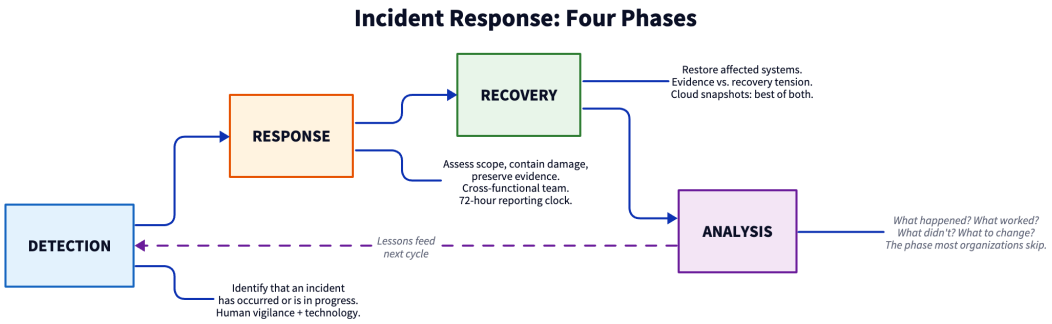

Figure 4: Incident Response: Four Phases

The IR Plan

An incident response plan is not a document that sits in a binder until needed. It is a living document that must be:

- **Read** by everyone who has a role in it — not just IT, but also legal, communications, compliance, clinical operations (in healthcare), and executive leadership
- **Tested** through tabletop exercises at least annually. A tabletop walks through a realistic scenario, with each participant describing what they would do at each decision point. This reveals gaps, conflicts, and assumptions that look fine on paper but fail in practice
- **Updated** after every test and every actual incident. The plan that was adequate last year may not account for new systems, new personnel, new regulations, or new attack techniques

72-Hour Reporting

Multiple regulatory frameworks now require or propose incident reporting within 72 hours of discovery:

- HIPAA proposed rule changes
- NCUA requirements for federally insured credit unions
- SEC disclosure rules for material cybersecurity incidents (four business days)
- EU GDPR (72 hours for personal data breaches)
- Various state breach notification laws

72 hours is aggressive. Your incident response process must be able to detect, classify, assess materiality, and initiate reporting within that window. If your process relies on a weekly security review to identify incidents, you will miss the deadline. If your escalation path requires approvals from people who are unreachable on weekends, you will miss the deadline.

Build the 72-hour clock into your IR plan. Know in advance who makes the reporting decision, what information the report requires, and how to reach every person in the escalation chain at any hour.

Cross-Functional Response

Incident response is not an IT function. It is a business function that requires coordination across:

- **IT and security** — technical investigation, containment, and remediation
- **Legal** — privilege considerations, regulatory notification requirements, litigation hold
- **Communications** — internal messaging to employees, external messaging to customers and media
- **Compliance** — regulatory reporting obligations and documentation requirements
- **Executive leadership** — business decisions about disclosure, remediation investment, and operational continuity
- **Clinical or operational leadership** (in healthcare and other critical services) — patient safety implications, manual fallback procedures

An IR plan that assigns everything to the IT team is not an IR plan. It is a hope that the IT team can figure out the legal, communications, compliance, and business continuity implications in real time, under pressure, without guidance. They cannot.

Evidence vs. Recovery

The tension between preserving evidence and restoring operations is real and must be resolved in advance — not during the incident.

Evidence priority means isolating affected systems without alteration. Preserve disk images, memory dumps, and log files. Engage forensic investigators. This takes time — potentially weeks — and the affected systems remain offline throughout.

Recovery priority means rebuilding from clean backups on fresh hardware as quickly as possible. Get the business running. This may destroy evidence that would have been useful for prosecution or root cause analysis.

In most cases, business recovery takes precedence. But make the decision deliberately, document it in your IR plan, and ensure that leadership understands the trade-off before an incident forces the decision under pressure.

The cloud changes this calculus. For workloads running on cloud VM instances, you can have both. A forensic snapshot of the affected instance — disk, memory state, and metadata — can be taken in seconds and transferred to a separate, isolated account. The snapshot is available for forensic analysis without anyone touching the original, preserving chain of custody. Meanwhile, the production instance can be terminated and rebuilt from clean images immediately. The evidence/recovery trade-off that dominated incident response planning for decades is largely eliminated for cloud-hosted systems. If your workloads are in the cloud and your IR plan still forces a choice between evidence and recovery, update the plan.

Chapter 16: Business Continuity and Disaster Recovery

When Systems Go Down

Business continuity planning (BCP) and disaster recovery (DR) address the question: how does the organization continue to function when critical systems are unavailable?

For many organizations — particularly small ones — BCP/DR planning is either absent or perfunctory. The plan, if it exists, assumes a short outage followed by a clean recovery. The ransomware reality is different: systems may be unavailable for days or weeks, data may be lost or compromised, and the recovery process itself may introduce new risks.

There is a tendency in larger organizations to move BCP/DR outside the security function — under operations, facilities, or a dedicated business continuity team. This is understandable from an organizational perspective, but it creates a risk: BCP/DR is the primary control for the availability leg of the CIA triad (confidentiality, integrity, availability). If BCP/DR lives outside the CISO's control, the security program may develop incident response plans that conflict with or duplicate the BCP/DR plans. At minimum, the CISO or a designee must be a stakeholder in the BCP/DR process, and the BCP/DR plan must be tested jointly with the incident response plan. Parallel efforts that don't coordinate produce plans that fail at the seams — exactly where the stress of a real incident concentrates.

Core Provider Dependency

Most small and mid-size organizations depend on a small number of core providers for critical functions. A community bank's core processing system handles deposits, withdrawals, lending, online banking, and regulatory reporting. A healthcare organization's EHR system is the backbone of clinical operations. When that provider has an outage, a breach, or a business continuity event, your organization is along for the ride.

This concentration risk is structural and largely unavoidable — the market has consolidated to the point where switching providers is a multi-year undertaking. But you can manage the risk:

- **Understand your dependencies.** Map which business functions depend on which systems, which systems depend on which providers, and which providers depend on which infrastructure.
- **Define recovery priorities.** Not every function can be recovered simultaneously. Which functions must be restored first? What manual fallback procedures exist for each?
- **Test the fallback.** Can your tellers process transactions manually? Can your clinical staff access medication records on paper? Can your finance team make payroll without the ERP system? If you haven't tested it, the answer is no.

Manual Fallback and Emergency Mode Operations

The most resilient organizations are those that can continue critical operations without technology — at reduced capacity and reduced speed, but without stopping. This requires documented manual procedures that are tested regularly and understood by the people who will execute them.

The readiness for manual fallback varies dramatically by industry, and the contrast is instructive.

Financial institutions have decades of practice operating without IT. Community banks and credit unions know how to operate from their trial balance and paper receipts. Tellers can process deposits, withdrawals, and loan payments manually. The procedures are documented, tested, and — in many cases — within living memory of staff who did it that way before the systems existed. Emergency mode operations (EMO) are de rigueur in the financial sector, and examiners expect to see them tested.

Healthcare is far more brittle. The industry spent two decades digitizing clinical operations under the Meaningful Use mandate, successfully eliminating paper charts, paper orders, and paper medication administration records. The unintended consequence is that many healthcare organizations have lost the ability to operate without their EHR. We have seen hospitals shut down their emergency department because a wristband printer went offline — a $200 peripheral that, when it failed, made it impossible to identify patients using the workflow the staff had been trained on. No one remembered how to write a wristband by hand, because no one had done it in years.

The irony is acute: Meaningful Use was intended to improve patient safety by eliminating the errors inherent in paper-based clinical processes. It succeeded. But it also created a single point of failure that the paper-based system never had. A paper chart cannot crash. A paper medication list does not require a network connection. A handwritten wristband does not depend on a printer driver.

Healthcare organizations need to maintain the capability to: - **Print or manually prepare patient identification** (wristbands, labels) without the EHR - **Print or access current medication lists and patient charts** at defined intervals and store them securely — so that if the EHR goes down, the most recent clinical information is available on paper - **Admit, treat, and discharge patients** using manual documentation that can be reconciled with the EHR when it recovers - **Communicate orders** between physicians, nurses, and pharmacy without the computerized order entry system

This is not a return to paper charts. It is a recognition that availability — the “A” in CIA — requires a fallback that does not depend on the system that failed. The procedures exist only if they are written down, and they work only if people have practiced them. An EMO plan that has never been tested is not a plan — it is a hope.

Chapter 17: Security Testing

Validating Your Defenses

Security testing answers a simple question: do your defenses actually work? Not in theory, not on paper, not according to the vendor — in practice, against the attacks that matter in your environment.

Types of Testing

Vulnerability assessment identifies known weaknesses in your systems — unpatched software, misconfigured services, default credentials, exposed management interfaces. Vulnerability assessments are automated, repeatable, and should be performed continuously or at least monthly. They tell you what’s wrong; they don’t tell you whether an attacker can exploit it.

Penetration testing goes further: a tester attempts to exploit identified vulnerabilities to determine whether they can be used to gain unauthorized access, escalate privileges, or exfiltrate data. A good penetration

test simulates the attack chain described in Chapter 7 — reconnaissance, initial access, lateral movement, privilege escalation, and objective achievement.

Social engineering testing evaluates the human element — phishing simulations, pretexting calls, physical access attempts. These tests reveal how your people actually respond to attack techniques, not how they respond to training quizzes.

Application security testing evaluates web applications, APIs, and mobile apps for vulnerabilities. The modern application security testing landscape includes several distinct disciplines:

- **SAST (Static Application Security Testing)** analyzes source code, bytecode, or binaries without executing the application. SAST finds vulnerabilities early in development — hardcoded credentials, injection flaws, insecure cryptographic usage — before the code reaches production. The limitation is false positives: SAST tools flag patterns that look dangerous but may not be exploitable in context. SAST is most valuable integrated into the CI/CD pipeline, where it runs automatically on every commit. - **DAST (Dynamic Application Security Testing)** tests the running application from the outside — simulating an attacker who has no access to source code. DAST finds vulnerabilities that only manifest at runtime: authentication bypass, cross-site scripting, server misconfigurations, and exposed APIs. DAST complements SAST because some vulnerabilities are visible only in code, others only in execution. - **IAST (Interactive Application Security Testing)** combines elements of both: an agent runs inside the application during testing, observing code execution paths while DAST-style tests are performed. IAST produces fewer false positives than SAST because it sees actual runtime behavior, and identifies more vulnerabilities than DAST alone because it has visibility into the code. - **SCA (Software Composition Analysis)** inventories third-party libraries and open-source components in your applications and checks them against known vulnerability databases (CVE, NVD, GitHub Advisory Database). The supply chain attacks described in Chapter 7 make SCA essential: you are responsible for the vulnerabilities in every library your application includes, whether you wrote it or not. SCA should run continuously, because a library that was clean yesterday may have a disclosed vulnerability today. - **API security testing** specifically evaluates API endpoints for broken authentication, broken object-level authorization (BOLA), excessive data exposure, lack of rate limiting, and injection vulnerabilities. APIs are tested differently from web applications because they lack a user interface — the tester interacts directly with the programmatic interface, which often exposes more data and functionality than the UI.

The right combination depends on your environment: organizations that develop custom applications need SAST, DAST, and SCA integrated into their development pipeline. Organizations that primarily consume vendor applications need DAST and API testing against their deployment configurations. Every organization needs SCA — because every application has dependencies.

Continuous Testing vs. Annual Assessment

An annual penetration test is a snapshot — valuable, but stale within weeks as systems change, patches are applied (or not), and new services are deployed. Continuous testing — automated vulnerability scanning, regular configuration audits, periodic penetration tests, and ongoing social engineering exercises — provides a more accurate and current picture of your security posture.

The RAPID methodology applies here as well: testing is not a one-time event but an iterative process integrated into the program's ongoing rhythm. Each test cycle identifies weaknesses, each remediation cycle addresses them, and the next test cycle validates the fix while identifying new weaknesses introduced by the changes.

The Value of External Testing

Internal testing — performed by your own IT or security team — is valuable for routine validation. External testing — performed by a qualified third party — is essential for objective assessment. Your own team has blind spots: assumptions about what "should" be secure, familiarity with the environment that prevents seeing it as an attacker would, and organizational pressure to produce favorable results.

An external tester sees your environment the way an attacker does — from the outside, without assumptions, without institutional knowledge, and without pressure to confirm that everything is fine. The findings may be uncomfortable. That is the point.

PART V: GOVERNANCE, RISK, AND COMPLIANCE

Chapter 18: Risk Management

Measuring What Matters

Risk management is the discipline that connects every other chapter in this book. Security policy (Chapter 9) is a statement of risk decisions. Access controls (Chapter 10) are risk mitigations. Detection (Chapter 14) monitors for risk events. Incident response (Chapter 15) manages risk that has materialized. Every dollar spent on security is — or should be — a dollar spent on reducing risk.

Before we can discuss how to measure risk, we need to define what risk *is* — because the definition determines everything that follows.

The traditional definition, enshrined in the CISSP common body of knowledge and in the mental models of most security practitioners, is: **risk is the probability of loss events.** Under this definition, risk is inherently negative. Every risk is a bad thing. Controls reduce risk. More controls reduce more risk. The only question is how much risk you are willing to tolerate.

ISO 31000 offers a fundamentally different definition: **risk is the effect of uncertainty on objectives.** That single word — *objectives* — transforms risk from a one-dimensional negative (probability of loss) into a multi-dimensional analysis of outcomes. Uncertainty can produce negative outcomes (threats) or positive outcomes (opportunities). A decision to implement a control has both upside risk (it improves security) and downside risk (it costs money, creates friction, may break something). A decision *not* to implement a control also has both upside risk (saves money, reduces friction) and downside risk (leaves a vulnerability unaddressed).

This is the intellectual foundation for everything in this book about friction, optimal privilege, and the balance between security and productivity. The CISSP definition creates a discipline that can only say "no" — more risk is always worse, more controls are always better. The ISO 31000 definition creates a discipline that can evaluate trade-offs: does this action produce a net benefit to the organization's objectives, considering both the upside and the downside?

STORM models both upside and downside risk. Consider a simple example: you spend $1 on a lottery ticket with a jackpot of $500 million. The probability of losing your $1 is approximately 299,999,999 in 300,000,000 — very close to certainty. But the expected value is $500M ÷ 300M = $1.67. The expected return on the ticket is positive: $1.67 on a $1 investment. If you can afford to lose the $1, the lottery is a rational risk.

This trivial example illustrates three concepts that apply directly to security investment decisions:

1. **Expected value.** The expected outcome of an action is the sum of all possible outcomes weighted by their probabilities. A control that costs $50,000 and reduces a $2 million risk by 40% has an expected value of $800,000 — a good investment. A control that costs $500,000 and reduces a $200,000 risk

by 40% has an expected value of $80,000 — a bad investment, even though the risk is real and the control works.

2. **Risk appetite.** You might be able to tolerate losing $1 on a lottery ticket but not $5. Even if a $5 bet had an expected value greater than $5, you might not take it because the potential loss exceeds what you can absorb. Risk appetite is the maximum loss you are willing to accept in pursuit of an objective — regardless of the expected value.
3. **Risk tolerance.** If your risk appetite is $5, you will take any bet where the potential loss is $5 or less and the expected value is positive. You will not take a bet where the potential loss exceeds $5, even if the expected value is strongly positive. This is why small organizations cannot take the same risks as large ones — their appetite for loss is smaller, even when the proportional expected value is identical.

Every security investment decision is a bet. The ISO 31000 framing lets you evaluate it as one — with both upside and downside, not just downside. STORM provides the math to make that evaluation quantitative rather than qualitative.

The problem is that most organizations cannot actually measure their risk. They use qualitative labels — low, medium, high, critical — that feel like measurement but are not. Qualitative risk assessment is a consensus exercise, not an analytical one. Ten people in a room assign "medium" to a risk because it feels medium, not because they have evaluated the probability and impact against a defined scale. The result is a risk register full of colors that cannot be compared, trended, or used to make investment decisions.

Why Qualitative Risk Assessment Fails

Qualitative risk assessment fails for one fundamental reason, from which all other failures derive: **the counting problem.**

Organizations measure risk using qualitative labels — low, medium, high, critical — and then attempt to manage those labels as if they were data. They count them. Company A has 10 high risks, 50 medium risks, and 30 low risks. Company B has 100 high risks, 1,000 medium risks, and 500 low risks. If Company A remediates its worst risk, reducing from 10 highs to 9 highs, has it meaningfully reduced its risk? All it has done is reduce the count of high findings by one — it knows nothing about whether the remaining 9 highs are better or worse than before, whether the remediated risk was the most important one, or whether the remediation introduced new risks elsewhere.

The temptation to compensate is to assign numeric values to labels — Low=1, Medium=2, High=3 — and add them up. This produces a number that looks quantitative but is not. The numbers are ordinal: they indicate rank order but carry no information about magnitude. A "3" at a 200-person credit union does not represent the same risk as a "3" at a 20,000-person commercial bank. You cannot add, subtract, average, or perform any arithmetic on ordinal values and produce a meaningful result. The organizations that do this anyway are performing mathematical operations on labels and treating the output as measurement. It is not.

The counting problem destroys three capabilities that effective risk management requires:

1. Comparability. Is your "high" the same as my "high"? Is a "high" vulnerability risk comparable to a "high" compliance risk? Without a common unit of measurement, you cannot prioritize across risk categories. Every risk is "high" to the person who identified it.

2. Trend analysis. If your risk was "medium" last year and "medium" this year, has it improved, deteriorated, or stayed the same? You cannot tell. The label is the same, but the underlying conditions may have changed dramatically in either direction. If you had 100 highs last year and 90 highs this year, you might convince yourself that risk has decreased — but all you really know is that you have fewer high findings, not lower risk.

3. Decision support. If Risk A is "high" and Risk B is "high," and you have budget to address one, which do you choose? Qualitative labels provide no basis for that decision. In practice, the organization either

addresses both (spending more than necessary on one), addresses neither (because it can't decide), or addresses whichever risk has the loudest advocate in the room. Labels destroy the information value that risk assessment is supposed to produce.

Quantitative Risk Measurement: STORM

RESCOR's Simplified Total Risk Management (STORM) methodology addresses these failures by measuring risk as a percentage — a number that can be compared, trended, and used to make decisions.

STORM rests on three mathematical properties, proven across thousands of assessments since 1999:

1. **Worst case.** Overall risk is never less than your single worst individual exposure. If your worst vulnerability represents 15% risk, your overall risk is at least 15% — no matter how well protected everything else is.
2. **Multiple exposure.** Multiple exposures are worse than one. An organization with ten 5% vulnerabilities is at greater risk than an organization with one 5% vulnerability. Risk accumulates.
3. **Diminishing impact.** Each successive exposure has less additional impact than the first. The tenth vulnerability does not add as much risk as the first. This property ensures that STORM measurements remain bounded and meaningful rather than growing without limit.

A concrete illustration: imagine a building with four doors. Three doors are locked, bolted, and alarmed. The fourth is simply closed. The worst-case property tells us that the building's security posture is determined by that fourth door — no matter how well the other three are secured. If you leave a second door open, the multiple-exposure property tells us that risk increases — the attacker now has to try only half as many doors to find one that opens. But the diminishing-impact property tells us that the risk does not double: one open door already determined the worst case, and the second open door adds less marginal risk than the first. These properties are not arbitrary — they reflect how vulnerability surfaces actually behave.

STORM measurements have objective meaning. This is the critical distinction between STORM and qualitative-with-numbers. Assigning Low=1, Medium=2, High=3 produces comparability of a sort — you can tell that a "3" is worse than a "2" — but the comparison carries no objective meaning. Is a "3" three times as bad as a "1"? Twice as bad as a "1.5"? Nobody knows, because the numbers are labels, not measurements. STORM values, even at their lowest data maturity, are measurements on a consistent scale. A risk measured at 42 RU is objectively, meaningfully greater than a risk measured at 18 RU — not because someone assigned those numbers, but because the underlying asset, threat, and vulnerability assessments produce them through a repeatable calculation. The comparison is not just ordinal ("this is worse than that") — it is meaningful ("this is how much worse, and here is why").

These properties enable STORM to:

- **Compare risk across assessments.** Is the organization more or less secure than it was six months ago? By how much?
- **Prioritize remediation.** Which vulnerability contributes the most to overall risk? Address that one first.
- **Evaluate control effectiveness.** What is the expected risk reduction from a proposed control? Is the investment justified?
- **Report to boards and examiners.** A single percentage that represents the organization's risk posture is comprehensible to non-technical audiences in a way that a heat map of red, yellow, and green boxes is not.

Information Cost and the L-to-N Transition

The reason most organizations don't attempt quantitative risk assessment is not that they prefer qualitative — it's that they believe quantitative is too expensive. This belief is grounded in real experience. Traditional quantitative methods like Monte Carlo simulation, annualized loss expectancy (ALE) calculations, or full

FAIR analyses require detailed data about threat frequency, loss magnitude distributions, and control effectiveness — data that most organizations do not have and cannot afford to collect. The *information cost* of a rigorous quantitative assessment can exceed the cost of the controls it recommends.

Information cost is the effort required to obtain the data necessary for a risk assessment. Qualitative assessment has low information cost: you ask people what they think, they assign labels, and you move on. The result is cheap and fast — and, as described above, largely meaningless. Traditional quantitative assessment has high information cost: you gather historical loss data, model probability distributions, estimate annualized frequencies, and perform statistical analysis. The result is precise and actionable — but the cost and time required put it out of reach for most organizations.

STORM resolves this trade-off through what we call the **L-to-N transition** — "L" and "N" being the first letters at which "qualitative" and "quantitative" differ — a successively approximate quantitative result derived from information that costs not much more to collect than a qualitative assessment. STORM does not require actuarial loss data, probability distributions, or Monte Carlo simulations. It requires the same inputs a qualitative assessment uses — identification of assets, threats, vulnerabilities, and controls — but processes them through transforms (HAM533 for threat assessment, SAVP/SCEP/CCP for vulnerability and asset assessment) that produce numeric values on a consistent scale.

The data maturity factor. The L-to-N transition is not binary — it is a continuum governed by a data maturity factor that reflects the quality of the input data. At low maturity, asset values, threat likelihoods, and vulnerability exposures are estimated using simple, bounded-choice approximations — the same inputs a qualitative assessment would collect. The resulting STORM values (expressed in Base Risk Units, or BRU) are not precise in absolute terms: a risk measured at 0.42 BRU does not mean there is exactly $420,000 at stake in a $1 million asset base. But it *can* be directly compared to a second risk measured at 0.18 BRU (18 Scaled Risk Units, or RU), and that comparison will have meaning — the first risk is objectively, measurably greater than the second.

As data maturity increases — as asset valuations become more precise, threat likelihoods are informed by historical frequency data, and vulnerability surfaces are measured rather than estimated — the maturity factor progressively reduces the approximation in STORM's aggregate calculation. At full maturity, where an asset is actually known to represent 10% of the total asset base, the vulnerability surface is measured at 20%, and the aggregate threat likelihood is 33%, STORM produces a result that converges on the traditional quantitative answer: approximately $67,000 in quantitative risk against a $1 million asset base. Few organizations will reach full maturity across all risk domains — but the point is that STORM works at every point along the continuum, producing increasingly precise results as data quality improves, without requiring a different methodology at any stage.

This is the practical argument for STORM: even at minimum maturity, it provides a continuum of risk values rather than three or four or five buckets. And because STORM values are on a consistent scale, you can always project qualitative labels onto them once you establish risk appetite and tolerance — "anything above 30 RU is what we call 'high' in this organization" — while retaining the underlying numeric precision for comparison, trending, and investment decisions. You get the comprehensibility of labels with the analytical power of numbers. There is no legitimate reason to settle for labels alone when numbers are available at the same information cost.

Organizations that claim they cannot afford quantitative risk assessment are actually saying they cannot afford the traditional *implementation* of quantitative risk assessment. STORM changes that equation.

Risk Appetite and Risk Tolerance

Risk appetite and risk tolerance are related but distinct concepts, and the distinction matters operationally.

Risk appetite is the net downside risk an organization is willing to accept in pursuit of its objectives. Under the ISO 31000 framing, this is a differential quantity: the net of upside and downside risk that leadership considers acceptable. An organization with a $1 million revenue stream and a 20% profit margin might be willing to accept 5% net downside risk — meaning it will tolerate security investments that leave up to

$50,000 of expected annual loss unaddressed, because the cost of further mitigation would exceed the benefit.

Risk tolerance is the acceptable variation around that appetite. If the appetite is 5% net downside risk, a tolerance of ±5% means leadership will not escalate unless residual risk exceeds 10% — but it also means that a reduction below 0% (where security investments are producing net positive returns through reduced insurance premiums, avoided incidents, or competitive advantage) is recognized and valued, not just treated as "good enough."

Both are business decisions, not technical decisions, and both should be documented and approved by the board or senior leadership.

Most organizations have neither. They operate with an implicit risk appetite of "as little risk as possible" — which sounds prudent but is not, because it provides no basis for deciding when a risk is acceptable. Every risk assessment produces a list of findings, every finding demands remediation, and the security program becomes an infinite list of things to fix with no way to determine when enough is enough.

A documented risk appetite provides the stopping point: "We accept residual risk at or below X%, and we will invest in controls to bring risk below that threshold. Risks below the threshold are accepted and monitored. Risks above the threshold are remediated or transferred." This is the only way to run a security program without either over-investing (Wolverine) or under-investing (Ostrich).

Risk Treatment: Four Options

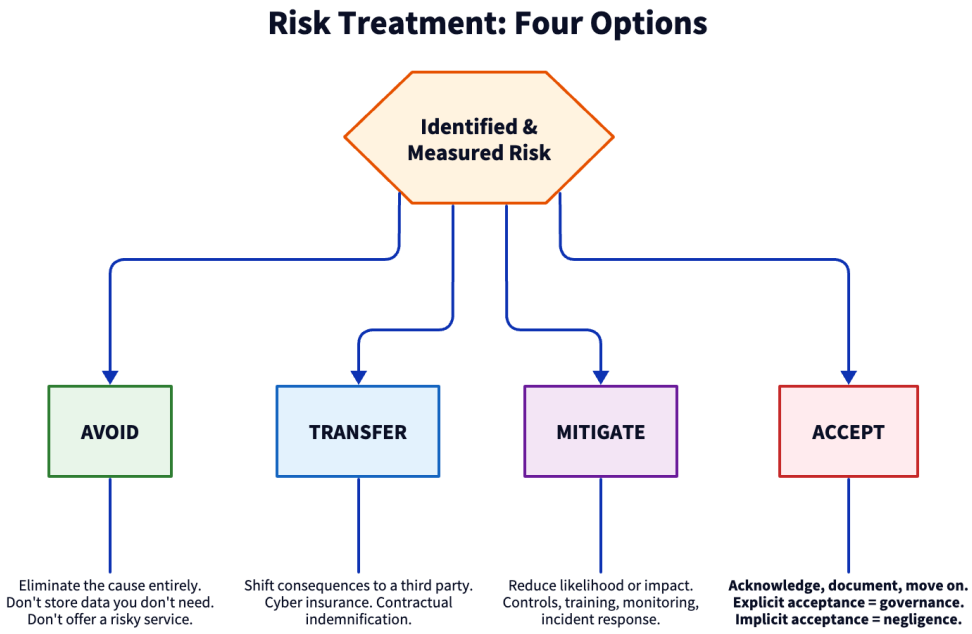

Figure 5: Risk Treatment: Four Options

Once you have identified and measured a risk, there are exactly four things you can do with it:

Avoid. Eliminate the cause of the risk entirely. If you don't store data you don't need, you can't lose it. If you don't offer a service, it can't be exploited. Avoidance is the most effective treatment — but it is often not practical, because the risk-generating activity is also a business-generating activity. The organization that avoids all risk avoids all opportunity.

Transfer. Shift the financial consequences of a risk to a third party. The most common form of risk transfer is insurance. Cyber insurance policies can cover breach response costs, business interruption, regulatory fines (in some jurisdictions), and legal defense. However, insurance is not a substitute for controls:

- Cyber insurance policies have narrowed significantly in recent years. Many exclude ransomware payments, nation-state attacks, or losses resulting from unpatched known vulnerabilities.
- Insurers increasingly require specific controls — MFA, EDR, offline backups, incident response plans — as preconditions for coverage. Failure to maintain these controls can void the policy.
- Insurance covers financial loss. It does not cover reputational damage, customer trust, or the operational chaos of an incident in progress.
- Premiums have increased substantially, particularly for organizations in healthcare and financial services. The cost of transfer must be weighed against the cost of mitigation.

Contractual liability provisions — requiring vendors to indemnify you for breaches originating from their systems — are another form of transfer. They are valuable but imperfect: a vendor that causes a breach and then goes bankrupt provides no indemnification. The contract is only as good as the vendor's ability to honor it.

Mitigate. Reduce the likelihood or impact of a risk through controls. This is what most of this book is about: security policies, access controls, encryption, monitoring, incident response, and all the other elements of a security program. Mitigation is the most common treatment — but not every risk requires mitigation, and not every mitigation is worth its cost (see Control Friction, Chapter 10).

Accept. Acknowledge the risk, document the decision, and move on. Acceptance is appropriate when the cost of avoidance, transfer, or mitigation exceeds the expected impact of the risk — or when the risk is within the organization's documented risk appetite.

The critical distinction is between **explicit acceptance** and **implicit acceptance**. Explicit acceptance is a governance act: the risk is identified, measured, evaluated, and a documented decision is made to accept it, with a rationale, an owner, and a review date. Implicit acceptance is what happens when no one addresses the risk — it persists by default, undocumented, unowned, and unreviewed. Implicit acceptance is not risk management. It is negligence.

Every risk in your register should be tagged with its treatment: avoided, transferred, mitigated, or accepted. If you can't tag it, you haven't made a decision. If you haven't made a decision, you've implicitly accepted it — which means you haven't managed it at all.

The Board's Role

Risk governance is a board-level responsibility. The board does not need to understand the technical details of every vulnerability — but it does need to:

- **Set risk appetite** — how much risk is the organization willing to accept?
- **Receive regular reporting** — is risk within appetite? What are the top exposures? What is the trend?
- **Fund the program** — is the security budget proportionate to the risk?
- **Hold management accountable** — is the program producing measurable results?

A board that delegates risk entirely to IT has abdicated its governance responsibility. A board that micromanages technical controls has confused its role. The right level of engagement is strategic: set the appetite, fund the program, monitor the results, ask hard questions.

Chapter 19: Compliance Is Not Security

The Checkbox Trap

Compliance means meeting the requirements of a specific regulatory framework, standard, or contractual obligation. Security means protecting your information assets against the threats that actually exist in your environment. These are related but not identical objectives.

An organization can be fully compliant and completely insecure. HIPAA requires a risk analysis — but it does not specify what methodology to use, what findings to act on, or how quickly to remediate. An organization that performs a superficial risk analysis, documents the findings, and files the report is compliant. An organization that performs a thorough risk analysis, identifies critical vulnerabilities, and remediates them is secure. The regulation cannot distinguish between the two.

Conversely, an organization can be highly secure and technically non-compliant — because it implemented controls that are more appropriate for its environment than the ones the framework specifies, but did not document them in the format the auditor expects.

The checkbox trap is the belief that compliance equals security. It does not. Compliance is a minimum standard. Security is an ongoing practice. Compliance is audited periodically. Security is tested continuously. Compliance asks "did you do what the regulation requires?" Security asks "are you protected against the threats that matter?"

Regulatory Landscape

Organizations in regulated industries must navigate multiple overlapping frameworks:

HIPAA Security Rule (45 CFR 164.308-312) — applies to covered entities and business associates handling protected health information (PHI). Requires administrative, physical, and technical safeguards. Distinguishes between "required" and "addressable" specifications — addressable does not mean optional; it means the organization must implement the specification or document why an equivalent alternative is appropriate.

GLBA Safeguards Rule (16 CFR 314) — applies to financial institutions. Amended in 2021 (effective 2023) to require specific technical controls including MFA, encryption, and access controls. Requires a written information security program (WISP) and a designated qualified individual responsible for security.

FERPA (20 U.S.C. § 1232g) — protects student education records. Less prescriptive than HIPAA or GLBA but increasingly enforced as schools adopt cloud services and digital learning platforms.

FISMA / FedRAMP — applies to federal agencies and cloud service providers serving federal agencies. Requires implementation of NIST SP 800-53 security controls and continuous monitoring.

NERC CIP — applies to the bulk electric system. Prescribes specific controls for critical infrastructure with penalties for non-compliance that can reach $1 million per day per violation.

State privacy laws — CCPA/CPRA (California), and a growing patchwork of state laws with varying requirements for data handling, consumer rights, breach notification, and enforcement.

Framework Mapping

Compliance frameworks overlap significantly. An access control implemented for HIPAA may also satisfy GLBA requirements. A risk assessment performed for NIST CSF may also satisfy ISO 27001 requirements. Maintaining separate programs for each framework is wasteful and unsustainable.

Framework mapping — identifying where requirements overlap and implementing a single control that satisfies multiple frameworks — is essential for organizations subject to multiple regulatory regimes. RAPID (Chapter 5) provides the iterative process for building and maintaining a unified program that satisfies multiple frameworks without duplicating effort.

As established in Chapter 1, frameworks provide structure for thinking — not answers. RAPID tells you how to implement what the framework identifies. STORM (Chapter 18) tells you whether it's working. Together, they provide a complete governance model that satisfies compliance requirements while actually improving security — which is, after all, the point.

The Mandatory Encryption Debate

A specific example of the tension between compliance and security: the proposed update to the HIPAA Security Rule would make encryption a required specification rather than addressable.

Encryption at rest protects against exactly one threat: physical theft of media. Any application or user with legitimate access receives cleartext data — the encryption is transparent. An unencrypted database on a properly segmented network with no external ingress is arguably more secure than an encrypted database accessible from the Internet. Making encryption mandatory in all cases doesn't improve security — it eliminates the risk analysis that determines where encryption actually matters, replacing professional judgment with a checkbox.

The entire point of the addressable/required distinction was to let organizations make risk-based decisions. Mandatory controls obviate risk analysis. That's not security — it's compliance theater. And it's the Tortoise mentality[2] applied to regulation: a hard shell of mandatory controls that feels protective but doesn't address the actual threats.

Chapter 20: Third-Party Risk Management

Your Vendors Are Your Attack Surface

Third-party risk management (TPRM) addresses the risks introduced by your vendors, service providers, and business partners. In a world where every organization depends on external providers for critical functions, your security posture is only as strong as your weakest vendor.

The supply chain attacks described in Chapter 7 — SolarWinds, Kaseya, MOVEit — all exploited trust relationships between organizations and their vendors. The vendor's compromised software became the attack vector. The vendor's legitimate credentials became the attacker's credentials. The trust you extended to the vendor was the trust the attacker exploited.

Vendor Oversight

Effective TPRM includes:

- **Vendor inventory.** Know who your vendors are, what they have access to, and what data they process on your behalf. This sounds basic, but many organizations cannot produce a complete vendor list.
- **Risk assessment.** Evaluate each vendor's security posture based on the sensitivity of the data and systems they access. A vendor that processes your payroll requires different scrutiny than a vendor that supplies office furniture.
- **Contractual requirements.** Contracts should specify security obligations, incident notification requirements, audit rights, and liability provisions. A contract that does not address security is a contract that assumes the vendor's security is adequate — without evidence.
- **Ongoing monitoring.** TPRM is not a one-time assessment. Vendors change. Their security posture changes. Their subcontractors change. Continuous monitoring — through periodic reassessment, SOC 2 report review, and automated risk scoring — is essential.

[2]The Security Animal Quiz (https://www.rescor.net/security-quiz/) identifies five security archetypes — Ostrich, Sloth, Tortoise, Wolverine, and Dragon — representing a progression from disengagement to balanced, adaptive security practice.

The Small Organization Problem

Examiners expect TPRM programs that demonstrate ongoing monitoring of critical vendors, including their supply chain. For small organizations, this creates a practical problem: you have no leverage over large vendors. You cannot force your core banking provider to give you a penetration test report. You cannot audit Microsoft's security practices.

What you can do:

- **Review available documentation.** SOC 2 reports, SIG questionnaires, vendor security certifications.
- **Assess what you control.** Your configuration of the vendor's platform, your access management, your monitoring of the vendor's connections to your systems.
- **Document your due diligence.** The examiner's question is not "did you audit the vendor?" but "did you exercise reasonable diligence given your resources?" Document what you reviewed, what you assessed, and what you concluded.
- **Plan for vendor failure.** What happens if your core provider is unavailable for a week? A month? The business continuity plan (Chapter 16) must account for vendor dependencies.

Concentration Risk

When a small number of providers serve a large number of customers, a single provider's failure affects entire industries. A community bank's core processing system going down doesn't affect one bank — it affects every bank on that platform. A cloud provider's outage doesn't affect one customer — it affects millions.

Concentration risk is largely unavoidable for small organizations — the market has consolidated to the point where alternatives are limited and switching costs are prohibitive. But understanding the concentration, documenting it, and planning for it is not optional. The RAPID cycle should include periodic reassessment of concentration risk and the adequacy of contingency plans.

Chapter 21: AI Governance

The Newest Frontier

Artificial intelligence has introduced a category of risk that did not exist when the original edition of this book was written — or when any previous edition was written. AI tools are being adopted faster than governance frameworks can adapt, creating a shadow IT problem that makes the early days of cloud adoption look orderly by comparison.

Your employees are using AI tools whether you have approved them or not. They are drafting correspondence, analyzing data, summarizing documents, generating code, answering questions, and making decisions with AI assistance. Every time someone pastes confidential information, customer data, or proprietary content into an AI tool, that data may be retained by the provider, used for model training, or surfaced in responses to other users. This is not hypothetical — it is happening now, in your organization, probably today.

PAND vs. DANP

AI governance requires a policy decision: is AI permitted by default, or prohibited by default?

Permit All Not Denied (PAND) — AI tools are permitted unless explicitly prohibited. This is the more practical approach for most organizations, given the volume of AI embedded in existing tools (email filters, search engines, IDEs, CRMs, operating systems). Focus enforcement energy on identifying and blocking tools that pose specific data-handling risks.

Deny All Not Permitted (DANP) — AI tools are prohibited unless explicitly authorized. This is the more conservative approach, appropriate for organizations handling highly sensitive data (healthcare, financial services, government). However, in practice, DANP organizations must authorize entire platforms (e.g., "Microsoft 365 including Copilot features") rather than individual AI capabilities — because enumerating every AI feature in every tool is an impossible task.

The reality check from Chapter 10 applies here: AI is embedded in everything. Your email spam filter uses AI. Your search engine uses AI. Your browser uses AI. Your phone keyboard uses AI. A policy that attempts to authorize every AI-powered feature in every tool will be unworkable. The practical distinction is between **standalone AI tools** (where users actively submit data) and **embedded AI features** (built into already-authorized platforms). Governance should focus on the former.

AI-Specific Risks

Beyond the data exposure risk, AI introduces several categories of risk that traditional security frameworks do not address:

Bias and fairness. AI models used for lending decisions, hiring, insurance underwriting, or clinical decision support may produce biased results that violate fair lending laws, employment law, or clinical standards. Unlike a human decision-maker, an AI model cannot explain its reasoning to an examiner or a court. "The model said so" is not an adequate answer when asked why a loan was denied.

Hallucination and accuracy. AI models generate plausible-sounding content that may be factually wrong. An AI-generated summary of a patient's medical history that omits a critical allergy is not a security breach — it is a patient safety event. An AI-generated legal brief that cites nonexistent cases is not a data breach — it is a professional liability event. The risk is not that AI is malicious but that it is confidently wrong.

Model poisoning. AI models can be deliberately manipulated through poisoned training data. If an attacker can influence the data used to train or fine-tune a model, they can influence the model's outputs — potentially in ways that are difficult to detect.

AI-generated code. Developers using AI coding tools are producing code faster — and introducing vulnerabilities faster. AI models trained on public code repositories reproduce insecure patterns (SQL injection, missing input validation, hardcoded credentials) because those patterns are common in training data. More concerning: AI models hallucinate package names that don't exist, and researchers have demonstrated that attackers can register those hallucinated names, turning AI suggestions into supply chain attack vectors. When an entire development team uses the same AI model, the same vulnerability class gets injected across the codebase — creating a monoculture that a single exploit can compromise. Organizations that allow AI-assisted development need secure coding standards that specifically address AI-generated code review, dependency verification, and the prohibition of accepting AI suggestions without understanding them.

Regulatory uncertainty. AI regulation is evolving rapidly. The EU AI Act, proposed US federal regulations, and state-level AI legislation are all in various stages of development. Organizations that adopt AI without a governance framework risk non-compliance with regulations that don't yet exist but are clearly coming.

What AI Governance Requires

A minimum AI governance program includes:

- **A policy.** PAND or DANP, with clear rules for data handling, tool authorization, and incident reporting. (The RESCOR AI Policy Builder tool can generate a customized policy.)
- **An inventory.** What AI tools are in use? What data do they process? What vendor agreements govern data retention and use?
- **Training.** Employees must understand what data they can and cannot submit to AI tools, and what happens when they make a mistake.
- **Monitoring.** How will you detect unauthorized AI tool usage or data exposure?

- **Explainability requirements.** For AI used in regulated decisions (lending, clinical, underwriting), can the decision be explained to an examiner? If not, the AI should not be making that decision.
- **Human oversight.** AI-generated outputs used in clinical, financial, or legal contexts must be reviewed by a qualified human before action is taken. AI is a tool, not an authority.

APPENDICES

Appendix A: Security Program Policy Inventory

The following table lists the policies that a comprehensive security program should include. Not every organization needs every policy — the appropriate set depends on your size, industry, regulatory environment, and risk profile. Use this as a checklist, not a mandate.

Policy	Purpose	Typical Audience
Information Security Policy	Overarching policy defining roles, responsibilities, governance structure, and risk management approach	All personnel
Acceptable Use Policy	Defines permitted and prohibited uses of organizational systems, data, and resources	All personnel
Access Control Policy	Governs how access is granted, reviewed, modified, and revoked; defines optimal privilege approach	IT, HR, management
Data Classification and Handling Policy	Defines data categories and protection requirements for each	All personnel
Password and Authentication Policy	Length requirements, MFA requirements, session management, account lifecycle	All personnel
Incident Response Policy	Defines what constitutes an incident, notification procedures, roles, and reporting requirements	IR team, management, legal
Business Continuity / Disaster Recovery Policy	How critical operations continue during and after disruptions	Operations, IT, management
Vendor / Third-Party Risk Policy	Vendor evaluation, monitoring, contractual requirements, concentration risk	Procurement, IT, legal
AI Governance Policy	PAND/DANP decision, data handling rules, tool authorization, oversight requirements	All personnel
Change Management Policy	How changes to systems and configurations are proposed, approved, tested, and implemented	IT, development
Physical Security Policy	Facility access, visitor management, equipment disposal, clean desk	All personnel, facilities

Policy	Purpose	Typical Audience
Privacy Policy	How personal information is collected, used, stored, shared, and disposed of	All personnel, legal
Remote Work / BYOD Policy	Security requirements for remote access, personal devices, and home office environments	All personnel
Security Awareness Training Policy	Training requirements, frequency, topics, and compliance tracking	All personnel, HR
Encryption Policy	Where encryption is required, what algorithms and key lengths, key management procedures	IT, security
Audit and Logging Policy	What is logged, how logs are managed, retention requirements, review procedures	IT, security

Appendix B: Glossary of Modern Security Terms

A comprehensive glossary of terms used in this book, updated for 2026. Terms that appeared in the original 2000 glossary and remain relevant are included with updated definitions.

Advanced Persistent Threat (APT) — A prolonged, targeted attack in which an adversary establishes and maintains unauthorized access to a network for an extended period, typically for espionage or pre-positioning purposes.

Attack Surface — The total set of entry points through which an attacker might gain access to a system or network. Includes network services, APIs, user interfaces, physical access points, and supply chain connections.

Business Email Compromise (BEC) — An attack in which a compromised or spoofed email account is used to issue fraudulent instructions, typically involving wire transfers, vendor payment changes, or credential harvesting.

BYOD (Bring Your Own Device) — A policy allowing employees to use personal devices for work purposes. Introduces risks related to device management, data leakage, and network access control.

CASB (Cloud Access Security Broker) — A security tool that sits between an organization and its cloud service providers, enforcing security policies for cloud access.

Ciphertext Harvesting — The practice of intercepting and storing encrypted data for future decryption, typically in anticipation of quantum computing advances.

Credential Stuffing — An automated attack that tests stolen username/password combinations against login systems, exploiting password reuse across services.

EDR (Endpoint Detection and Response) — A security technology that monitors endpoint devices for suspicious activity, provides real-time analysis, and enables automated or manual response actions.

FIDO2 / WebAuthn — A phishing-resistant authentication standard that uses public-key cryptography bound to specific domains, eliminating the risk of credential phishing.

IAM (Identity and Access Management) — The framework of policies and technologies that manages digital identities and controls access to resources.

IoT (Internet of Things) — Network-connected devices that are not traditional computers — sensors, cameras, medical devices, industrial controllers, smart appliances. Often have limited security capabilities.

Lateral Movement — The techniques an attacker uses to move through a network after initial compromise, typically escalating privileges and accessing additional systems.

MFA (Multi-Factor Authentication) — Authentication requiring two or more independent verification factors: something you know (password), something you have (device/token), or something you are (biometric).

MFA Fatigue — An attack that bombards a user with MFA prompts until they approve one to make the notifications stop, bypassing MFA without cracking it.

Microsegmentation — The practice of dividing a network into small, isolated segments to limit lateral movement and contain breaches.

OSINT (Open-Source Intelligence) — Information gathered from publicly available sources — social media, job postings, public records, DNS records, certificate transparency logs — used for reconnaissance.

Passkey — A FIDO2 credential stored on a user's device, providing phishing-resistant, passwordless authentication.

Ransomware-as-a-Service (RaaS) — A criminal business model in which ransomware operators provide tools, infrastructure, and support to affiliates in exchange for a share of ransom payments.

SIEM (Security Information and Event Management) — A platform that aggregates, normalizes, correlates, and analyzes security logs from across an organization's environment.

SOAR (Security Orchestration, Automation, and Response) — A technology platform that automates incident response workflows, reducing the time from detection to containment.

Supply Chain Attack — An attack that compromises a trusted vendor, software package, or service provider to gain access to the vendor's customers.

Synthetic Identity — A fabricated identity combining real and fictitious personal information, used to open accounts, build credit, and commit fraud.

XDR (Extended Detection and Response) — An evolution of EDR that integrates detection and response across endpoints, networks, email, cloud, and identity.

Zero Trust — A security architecture that requires verification of every access request regardless of network location, based on the principle "never trust, always verify."

Appendix C: Framework and Regulatory Reference Guide

Framework / Regulation	Scope	Key Requirements
NIST CSF 2.0	Voluntary, all industries	Govern, Identify, Protect, Detect, Respond, Recover
NIST SP 800-53 Rev. 5	Federal systems, FedRAMP	Comprehensive security and privacy control catalog
NIST SP 800-63B	Authentication	Password length over complexity, no mandatory rotation
NIST SP 800-171	CUI in non-federal systems	110 security requirements for controlled unclassified information
ISO 27001:2022	International, all industries	Information security management system (ISMS) requirements
ISO 27002:2022	International, all industries	Security control implementation guidance
ISO 31000:2018	International, all industries	Risk management principles and guidelines
HIPAA Security Rule	US healthcare	Administrative, physical, and technical safeguards for PHI
GLBA Safeguards Rule	US financial institutions	Written information security program, specific technical controls
FERPA	US education	Protection of student education records
FISMA	US federal agencies	Risk-based security programs for federal information systems
FedRAMP	US cloud providers to government	Standardized security assessment for cloud services
NERC CIP	US bulk electric system	Critical infrastructure protection standards
PCI DSS 4.0	Payment card processing	Security standards for cardholder data environments
SOC 2	Service organizations	Trust services criteria: security, availability, processing integrity, confidentiality, privacy
CCPA/CPRA	California consumers	Consumer privacy rights, data handling requirements
GDPR	EU residents' data	Comprehensive data protection, 72-hour breach notification, up to 4% revenue penalties
SEC Cyber Rules	US publicly traded companies	Material incident disclosure within 4 business days

Appendix D: Bibliography

Cheswick, W.R. and Bellovin, S.M. (1994). *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison-Wesley.

Elliott, A. and Knight, S. (2010). "Role Explosion: Acknowledging the Problem." *Software Engineering Research and Practice*, CSREA Press, pp. 349-355.

Ivanti (2024). *DEX Security Research Report*. https://www.ivanti.com/resources/research-reports/dex-security

NIST (2017; Rev. 4 draft 2024). *Digital Identity Guidelines: Authentication and Lifecycle Management*. Special Publication 800-63B. https://pages.nist.gov/800-63-4/

NIST (2024). *Cybersecurity Framework (CSF) 2.0*. https://www.nist.gov/cyberframework

Palo Alto Networks Unit 42 (2023). *Cloud Threat Report, Vol. 7*. https://www.paloaltonetworks.com/prisma/unit42-cloud-threat-research

Perrow, C. (1999). *Normal Accidents: Living with High-Risk Technologies* (updated edition). Princeton University Press.

RFC 2196 (1997). *Site Security Handbook*. https://www.rfc-editor.org/rfc/rfc2196

Saltzer, J.H. and Schroeder, M.D. (1975). "The Protection of Information in Computer Systems." *Proceedings of the IEEE*, 63(9), 1278-1308.

StrongDM (2022). *The Access Productivity Report*. https://www.strongdm.com/report/access-productivity

---

Index