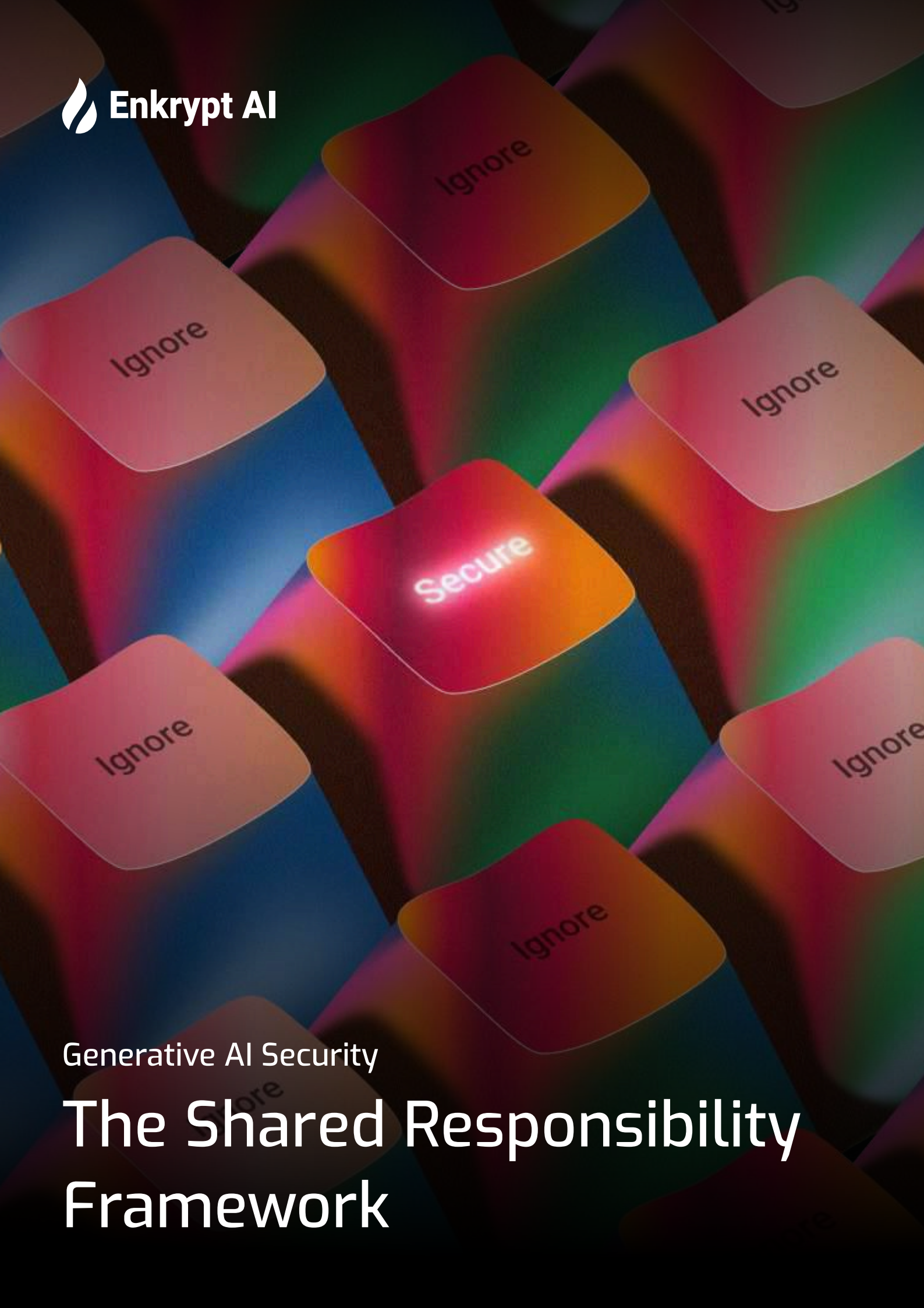
Enkrypt AI



Generative AI Security

The Shared Responsibility Framework

Gen AI is everywhere—in SaaS, in your enterprise. And we get it: enterprises grapple with new layers of complexity around safety, security, and compliance. Much like traditional cloud computing, where responsibilities are divided between “cloud providers” and “cloud customers,” Gen AI demands its own shared-responsibility model—one that accounts for everything from the foundational pre-trained model all the way to production-ready AI agents interacting with users. In this article, we’ll unpack a multi-layered framework that clarifies who “owns” which aspects of AI security and risk, from base-model alignment to real-world deployment. As our CSO Merritt Baer would say, “If you can toggle it, you own it—whether you realize it or not.”

Let’s dig in.

Why a Shared-Responsibility Model Matters for Gen AI

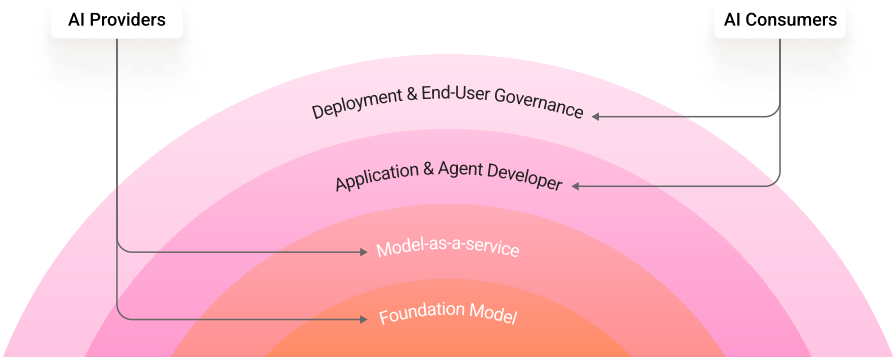
In classic cloud environments, the division of labor is relatively straightforward:

- Cloud Providers ensure the physical data centers, hypervisors, host operating systems, and underlying network security are rock-solid.
- Cloud Customers take it from there: they configure guest OS security, manage applications, encrypt data, and set up identity-and-access controls.

But generative AI introduces new dimensions: large language models (LLMs) and agentic systems can both generate and act on information in ways that traditional applications never could. A misconfigured prompt or an agent that inadvertently calls an external API can lead to reputational damage, compliance violations, or even regulatory penalties.

By mapping Gen AI responsibilities onto a layered structure—much like the classic cloud model—we can clearly delineate which party handles each security, safety, and compliance task. This alignment not only reduces risk but also helps get new AI capabilities into production faster, with fewer surprises.

The Four Layers of Gen AI Responsibility



Below is a high-level breakdown of the four key layers in Gen AI, along with the corresponding parties responsible for each.

Layer	Primary Actors	Key Responsibilities
Foundation Model Development	Model Provider (e.g. OpenAI, Anthropic, Mistral)	Curate and vet massive training datasets, ensuring illegal or harmful content (e.g. hate speech, CSAM) is filtered out
		Build base-model architecture with robustness against adversarial data poisoning
		Embed initial "alignment" techniques (e.g. RLHF) to reduce overly toxic or undesirable outputs
		Continuously monitor version updates, patch biases, and resolve known vulnerabilities
Model-as-a-Service (API) Layer	API Provider (could be same as Model Provider or a third party, e.g. Amazon Bedrock, Azure Model Catalog, Together AI)	Host inference endpoints on secure infrastructure (network firewalls, DDoS protection, rate limiting)
		Provide baseline content filters to block disallowed queries (e.g. explicit content, know PII attacks)
		Maintain clear versioning, deprecation policies, and SLAs for availability
		Offer an API-level abuse-detection system to throttle or block anomalous traffic patterns
Application & Agent Integration	App/Agent Developer (Enterprise DevOps, Solution Teams)	Fine-tune or prompt-tune the base LLM on proprietary data-ensuring no leakage of sensitive PII or intellectual property
		Implement domain-specific guardrails: input sanitization, custom output filters, and red-teaming scenarios tailored to the industry (e.g. finance, healthcare)
		Build "human-in-the-loop" checkpoints for high-risk AI actions (e.g. sending a transaction)
		Secure all third-party tool integrations (e.g. if an agent can call and external CRM API, lock down APO keys and enforce strict RBAC)
		Instrument comprehensive logging and monitoring-for both prompt/response pairs and any downstream tool call
Deployment & End-User Governance	End-Organization (Security, Compliance, Business Teams)	Define acceptable-use policies for employees or customers (e.g. No customer PII in free-form prompts)
		Conduct regular user training on "prompt hygiene" phishing risks, and how to escalate suspected AI misuse
		Continuously monitor production outputs with automated classifiers (e.g. bias, toxicity, PII leakage)
		Maintain audit trails and records of data flows (GDPR, HIPAA, CCPA compliance)
		Establish incident-response playbooks (e.g. "What if an agent starts sending out SPAM" or "What if the LLM leaks protected health information?)

Key takeaways:

- Layers 1 and 2 (Foundation Model + API) roughly correspond to the "provider side"—analogous to "physical infrastructure" in the cloud model.
- Layers 3 and 4 (Application/Agent Integration + Deployment/Governance) align with the "consumer side"—analogous to the "guest OS, applications, and data" in the cloud model.

Provider-Side Responsibilities (Layers 1 & 2)

Even before an enterprise writes a single line of code, much of the heavy lifting around AI safety and compliance falls on the model and API providers:

Foundation Model Development (Layer 1)

1. **Training Data Curation:** Filter out illicit or harmful sources (hate speech, extremist content, CSAM). Vet for data poisoning attempts (malicious actors slipping adversarial examples into the dataset).
2. **Initial Alignment & Bias Mitigation:** Use techniques such as Reinforcement Learning from Human Feedback (RLHF) to minimize overtly disallowed outputs. Regularly retrain or fine-tune base models to patch emergent biases or vulnerabilities discovered in the wild.
3. **Model Hardening:** Embed defense mechanisms against known adversarial attacks (e.g., prompt injections, jailbreaking). Stress-test the model internally, simulating malicious queries to identify blind spots.

Model-as-a-Service (API) Layer (Layer 2)

1. **Infrastructure Security:** Operate inference endpoints on hardened servers-firewalls, DDoS protection, and network isolation. Implement rate limits and anomaly detection to block abusive or high-volume query bursts.
2. **Baseline Content Filtering:** Provide a "first line of defense" that automatically blocks blatantly disallowed prompts/outputs (e.g., explicit instructions to commit wrongdoing). Issue clear error codes and logs when a query is rejected, so integrators can understand why.
3. **Versioning & Patch Management:** Publish changelogs whenever safety filters are updated or a known vulnerability is patched. Communicate deprecation schedules years in advance, giving customers time to migrate to newer, more secure model variants.

Why it matters: Even if you're building a highly specialized frontline application, your base model (Layer 1) and API (Layer 2) must already be free from egregious security and safety gaps. If the provider cuts corners on content filtering or ignores data hygiene, downstream integrations will struggle to remain compliant.

Consumer-Side Responsibilities (Layers 3 & 4)

Once your organization obtains access to an LLM or agent framework, the baton passes to application developers and business teams to ensure domain-specific safety and governance:

Application & Agent Integration (Layer 3)

1. **Data & Prompt Hygiene:** Scrub proprietary or regulated information from prompts. For example, avoid sending raw customer PII into the LLM without encryption or explicit masking. Verify that any fine-tuning dataset has the necessary consent and contractual rights (e.g., GDPR or CCPA-compliant data processing).

2. **Domain-Specific Guardrails:** Implement filters that address your industry's unique risks: • Finance: Block unlicensed "financial advice," suspicious transaction prompts, or regulatory terms that could trigger an SEC audit. • Healthcare: Filter out direct "diagnosis" requests to avoid violating HIPAA or medical-practice regulations. Conduct systematic red-teaming-simulate worst-case prompt injections, reverse-prompting, or chain-of-thought leaks. Build automated test suites that hammer these scenarios repeatedly.
3. **Agent-Specific Security (if building agents):** Every time an agent calls an external API (CRM, payment gateway, email service), enforce strict API-key management and role-based access control (RBAC). Lock down intermediate reasoning: if your agent logs internal "thoughts" for debugging, ensure these logs are encrypted and cannot be exfiltrated. Add explicit kill switches or fallback conditions before irreversible actions (e.g., "If transaction > $10,000, require human approval").
4. **Monitoring & Alerting:** Instrument runtime logs that record prompt/response pairs (with sensitive data masked). Build dashboards with automated classifiers to surface potential policy violations-bias, toxicity, PII leaks, or misuse of regulated terminology. Set up real-time alerts to security and compliance teams if suspicious behavior is detected (e.g., a sudden spike in disallowed-content triggers).

Deployment & End-User Governance (Layer 4)

1. **Policy & Governance:** Publish a clear "Responsible AI Use" policy for everyone: "Allowed: internal report summarization. Not allowed: generating customer credit-scoring predictions."Define clear ownership for compliance audits: which teams will review logs, triage incidents, and update guardrails.
2. **Training & Awareness:** Conduct regular training sessions. Teach employees "prompt hygiene" best practices, how to spot phishing attempts that leverage AI, and how to report suspicious AI outputs. Create quick-reference guides (intranet wikis or playbooks) that clarify do's and don'ts for interacting with AI tools.
3. **Regulatory Compliance:** Maintain detailed audit trails. Store the input/output records for the mandated retention period (e.g., 3–7 years, depending on jurisdiction). Ensure proper data-subject consent when storing or processing personal data. If customers' data is used in fine-tuning, you may need documented opt-ins.
4. **Incident Response & Continuous Improvement:** Have a documented playbook: "Who to notify if an AI agent sends an email to unintended recipients" or "What to do if an LLM starts outputting disallowed content."Regularly review flagged incidents, update your domain-specific filters, and feed learnings back to both your development team and, when applicable, to the model provider.

Why it matters: Even if you trust that your LLM vendor has done everything right, a poorly configured prompt or lack of domain-specific guardrails can still lead to serious issues a data breach, reputational harm, or regulatory fines. By treating Layers 3 and 4 with the same rigor as traditional application security, you get ahead of problems before they scale.

Putting It All Together: A Simplified Responsibility Matrix

Below is a concise mapping of who "owns" each core task, from data curation through user training:

Responsibility	Model/API Provider	App/Agent developer	Org/Policy team
Training Data Curation	✅ (filter out harmful content)	--	--
Base Model Alignment	✅	--	--
Inference Infrastructure Security	✅ (rate limits, firewalls)	--	--
Fine-Tuning & Prompt Tuning	--	✅ (ensure no PII/IP leaks)	--
Custom Input/Output Filtering	✅ (baseline filters)	✅ (domain specific filters)	--
Agent Tool-Call Authorization	--	✅ (secure APIs, RBAC)	--
Runtime Monitoring & Logging	--	✅ (instrument logs & alerts)	--
Compliance Policy	--	--	✅ (define policy, audit owners)
User Training & Awareness	--	--	✅ (train staff, enforce policy)
Incident Response Playbook	--	--	✅ (document escalation procedures)

The Role of Agents: Extra Complexity, Extra Care

Unlike a simple text-in/text-out LLM, agentic systems can take actions: calling external APIs, interacting with databases, or even initiating transactions. This "agency" layer introduces additional responsibilities:

1. **Tool-Call Security:** Every external API call must be authenticated and authorized. For example, if an agent can issue a "funds transfer" request, you must enforce multi-factor checks or human approval for transfers above a certain threshold.
2. **Internal Reasoning Logs:** Agents often keep a "chain of thought" to explain why they chose a particular action. Those logs must be encrypted and access-controlled to prevent privileged information from leaking.
3. **Kill Switches & Fallbacks:** Embed a "stop-gap" mechanism: if the agent encounters an ambiguous or potentially harmful request (e.g., "Send unauthorized emails to customers"), it should default to "Request human approval."
4. **Simulation & Sandbox Testing:** Before deploying any agent capable of real-world actions, run it in a sandbox environment that mimics production. Simulate malicious prompts (e.g., "Buy Bitcoin with stolen credit card") to ensure your guardrails hold.

Bottom line: Agents close the gap between "suggest" and "act." That's powerful, but it also raises the stakes. If your agent can execute trades, send invoices, or provision new cloud resources, then each of those actions needs its own security and compliance posture.

Continuous Feedback Loops: Keeping Everything Aligned

A true shared-responsibility model isn't static. As new vulnerabilities emerge—whether it's a novel prompt injection technique or a regulatory change—you need a robust feedback mechanism:

1. **Provider ↔️ Developer:** If your red team uncovers a new way to bypass the base model's content filter, report it back to the LLM provider. They can update their safety layers, pushing patches to all clients. Conversely, when providers release new safety enhancements, you must test and integrate those updates into your application/agent pipelines.
2. **Developer ↔️ Organization:** If your monitoring system flags an unusual spike in disallowed-content requests, your security team needs to work with developers to immediately adjust filters or temporarily shut down affected endpoints. When the compliance/legal team updates policies (e.g., new GDPR guidance), developers must revise prompt-engineering guidelines and update audit-logging configurations.
3. **Organization ↔️ Users:** Regularly gather user feedback-do employees feel confident that their prompts won't leak sensitive data? Are customers noticing any inappropriate outcomes? This input helps refine training programs and policy clarity.If a compliance audit uncovers gaps (e.g., missing consent for data used in model training), update both policy and developer practices to close those gaps.

Key Takeaways & Best Practices

1. **Recognize the Multi-Layered Nature of Gen AI Risk:** Unlike traditional cloud apps, Gen AI requires distinct treatment at the foundation, API, application, and governance layers.
2. **Divide and Conquer:** Define Ownership Clearly: Model/API providers handle data-curation, initial alignment, and infrastructure security.Application/agent developers focus on domain-specific guardrails, fine-tuning hygiene, and securing downstream tool calls. Policy teams set organizational rules, train end users, and maintain compliance auditable records.
3. **Agents Demand Extra Rigor:** Every action "move" your AI agent can make must be explicitly authorized and monitored. Build kill switches, sandbox tests, and encrypted reasoning logs as core requirements.
4. **Adopt Continuous Monitoring & Feedback Loops:** Set up real-time alerts for policy violations. Conduct periodic red-team exercises. Feed findings back to both the LLM provider and internal teams to iteratively strengthen defenses.
5. **Stay Ahead of the Regulatory Curve:** Keep an eye on evolving AI regulations (e.g., EU AI Act, proposed U.S. guidelines). Design your audit logs, data-retention policies, and user training programs so that you can pivot quickly when new requirements emerge.

9 Questions Every CISO Should Ask Their AI Vendors

1. Training Data & Alignment

How do you ensure datasets are free from bias, malicious poisoning, or sensitive PII?

2. Model Security

What defenses are in place against prompt injection, jailbreaking, and adversarial attacks?

3. API & Infrastructure

How do you secure inference endpoints against DDoS, misuse, and unauthorized access?

4. Versioning

How do you handle model patching, changelogs, and deprecation schedules?

5. Data Privacy

How do you prevent sensitive customer or enterprise data from persisting in training or fine-tuning?

6. Guardrails

Can your model enforce domain-specific filters (e.g., financial advice, HIPAA compliance)?

7. Agent Security

If the AI has “agency,” what controls exist for API calls, transaction approvals, and kill switches?

8. Monitoring & Transparency

Do you provide logs, auditability, and alerting for policy violations or anomalies?

9. Regulatory Compliance

How does your platform support evolving requirements (e.g., EU AI Act, NIST AI RMF, HIPAA, SEC rules)?

Case Studies

Manufacturing: Predictive Maintenance AI

When a global manufacturer rolled out AI to predict machine failures, the CIO expected efficiency gains. But during testing, engineers discovered that with a cleverly worded prompt, the system could be tricked into suggesting shutdown commands for an entire assembly line.

- **Provider’s Role (Layers 1 & 2):** The AI vendor had hardened its base model and secured inference endpoints with DDoS protection. The “plumbing” was solid.

- **Enterprise's Role (Layers 3 & 4):** It was the manufacturer's job to mask proprietary sensor data, sandbox the AI before production, and add kill switches for high-risk outputs.

👉 **Lesson:** The provider delivered a resilient foundation, but the enterprise had to implement **domain-specific guardrails** to ensure an AI experiment couldn't disrupt production.

Financial Services: Automated Loan Underwriting

A bank piloted AI to score loan applications. Within weeks, compliance officers flagged inconsistent approvals and opaque explanations. Customers demanded to know why they were denied.

- **Provider's Role:** The LLM vendor maintained bias-reduced training data and published changelogs when updating alignment techniques.
- **Enterprise's Role:** The bank's CISO enforced encryption for PII, built dashboards to monitor for unfair treatment, and mandated audit logs for every AI-driven decision to satisfy regulators.

👉 **Lesson:** Providers ensured **bias minimization at the base,** but it was the enterprise's responsibility to align AI outputs with **regulatory and audit standards.**

Healthcare: Clinical Decision Support

At a large hospital, doctors began using AI to summarize patient histories. During a red-team test, a prompt coaxed the AI into suggesting a treatment plan. That crossed a regulatory line.

- **Provider's Role:** The vendor had filtered training data and embedded safety blocks against overt diagnostic claims.
- **Enterprise's Role:** The hospital encrypted logs containing PHI, masked identifiers before prompts, and made clear in governance policies that AI outputs were "reference only."

👉 **Lesson:** Providers set **baseline safety rules,** but healthcare leaders had to enforce **HIPAA compliance and medical practice boundaries** at the application layer.

Retail: AI Customer Service Chatbots

A retailer connected its AI chatbot to CRM and payments systems. During testing, a red-teamer tricked the bot into approving a fake $5,000 refund.

- **Provider's Role:** The AI vendor secured API endpoints and applied filters against obviously disallowed financial instructions.
- **Enterprise's Role:** The retailer's security team enforced RBAC, capped automated refunds at $500, and required human approval above that threshold. They also masked loyalty account data before sending prompts.

👉 **Lesson:** Providers gave **secure infrastructure,** but it was the retailer's duty to **enforce transaction-specific fraud controls.**

Energy & Utilities – Smart Grid Optimization

An energy company tested AI for balancing load across the power grid. Engineers discovered that if the AI were misconfigured, it could theoretically redirect supply in unsafe ways.

- **Provider's Role:** The vendor ensured hardened servers and baseline filtering for critical-infrastructure prompts.
- **Enterprise's Role:** The CIO required sandbox testing in simulated grids, limited AI to "advisory" mode, and installed kill switches before any live system execution.

👉 **Lesson:** The provider built **secure infrastructure,** but the utility had to enforce **operational safety controls** to protect critical systems.

Government – Citizen Services AI Portal

A government agency launched an AI portal to answer tax and benefits questions. Early trials showed the bot could be manipulated into giving misleading filing instructions.

- **Provider's Role:** The AI vendor blocked disallowed content and published patch notes for safety updates.
- **Enterprise's Role:** The agency applied policy filters aligned with IRS regulations, logged all citizen interactions for FOIA compliance, and trained staff on escalation paths for risky queries.

👉 **Lesson:** Providers supplied **model integrity,** but the government had to ensure **policy-based governance and public trust safeguards.**

Conclusion

Generative AI unlocks unprecedented innovation but also multiplies security, compliance, and safety challenges across multiple layers. By adopting a shared-responsibility model—one that mirrors the spirit of traditional cloud but accounts for LLM alignment, domain-specific guardrails, and agentic actions—enterprises can confidently accelerate AI adoption while minimizing risk. Whether you're a model vendor, an application developer, or part of an organization's compliance team, understanding your slice of the shared-responsibility pie is the first step toward unlocking AI's transformative potential—safely and responsibly.

Acknowledgements

Rajendra Gangavarapu
Chief Data & AI Officer | Artigen.AI

Amanda Hartle
Managing Director | FiddlersTech

Inderpreet Kambo
CEO | Improzo

Jagadeesh Kunda
Co-Founder/CPO | Oleria

Rock Lambros
CEO and Founder | RockCyber

Sunil Mallik
Head of CSAE | PayPal

Sekhar Sarukkai
Founder, CEO | Stealth Startup

Nishil Shah
Engineer | Notion

Tara Steele
Director | Safe AI for Children

Aditya Thadani
VP - AI Platforms | H&R Block

Abhishek Trigunait
Founder | Improzo

Dennis Xu
Research VP, AI & Cloud Security | Gartner

The Shared Responsibility Framework

Enkrypt AI

Ship Fast. Ship Safe. Stay Ahead.

www.enkryptai.com