The Enterprise AI Risk & Validation Handbook

A Practical Guide to Testing, Governance, and Safe Deployment

Safe Intelligence

SAFE INTELLIGENCE

Introduction

AI agents can deliver step-function gains in productivity and customer experience, but they also introduce new operational risks: non-deterministic behavior, opaque failure modes, tool-driven “real world” actions, and fast-changing dependencies (models, prompts, data, vendors). In enterprise environments, especially regulated ones, success depends less on how quickly you can build an agent and more on whether you can **prove it is safe, reliable, and governed** over time.

This handbook is a practical playbook for doing exactly that. It lays out the controls and operating model needed to deploy agents with confidence: defining scope and accountability, tracking dependencies, testing for robustness and safety, monitoring in production and more as systems evolve.

This is an evolving area of best practice and we hope provides a good grounding for the current state of play.

Brought to you by the team at Safe Intelligence.

SAFE INTELLIGENCE

Contents

Glossary

AI based applications / AI Agents

Throughout the text we will generally be using the term AI Agent to refer to any IT system that can perform tasks and is internally powered in a significant way by a machine learning model (typically a large or small language model, but also including vision or tabular data models).

Accuracy

We'll use the term accuracy as defined by ISO/IEC TS 5723:2022 as "closeness of results of observations, computations, or estimates to the true values or the values accepted as being true." [ISO]

Robustness

We'll use the term robustness to refer to "ability of a system to maintain its level of performance under a variety of circumstances" (From the same source: ISO/IEC TS 5723:2022 [ISO]).

Resilience

We'll use the term resilience in the same was as defined in the NIST AI Risk Management Framework [NIST AI RMF] (also adapted from the above ISO/IEC standard) [See link above] framework as "ability to withstand unexpected adverse events or unexpected changes in their environment or use – or ability to maintain their functions and structure in the face of internal and external change and degrade safely and gracefully when this is necessary". We'll refer to the former property as resilience to circumstances and the later to resilience to failure.

What AI Vendors Don't Tell You

In today's environment no organization can implement their own AI systems and third party technology and services are an essential part of any implemented AI solution. The third party systems may be AI models themselves, agent frameworks, testing products, data management or any number of other solutions. Thankfully there is a large and capable ecosystem of tools, services and infrastructure to meet these needs.

Leading vendors have strong products and are continuously evolving their solutions to make them better. They are also generally truthful in describing capabilities. Despite this, it can be difficult to get a true sense of what is possible and what is not since commercial pressure logically causes emphasis of some features rather than others.

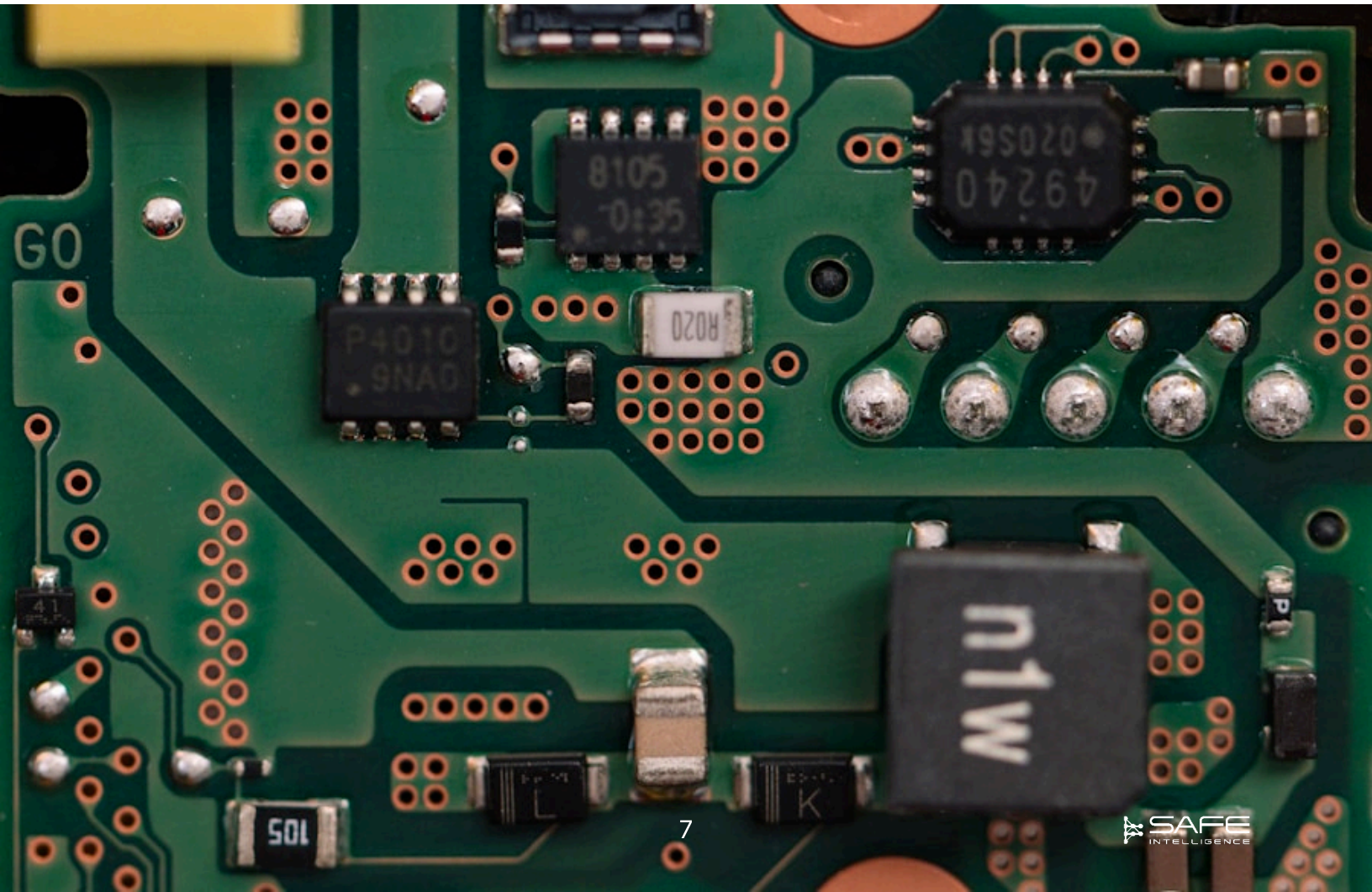
From a risk and validation perspective, here are a few key things that most vendors will acknowledge but likely do not put front and center in their marketing materials:

- Large Language Model based AI systems are extremely powerful, but while the scale of these systems unlocks their power (it gives them strong representational power), this size also limits our ability to predict how all internal components will behave for a given input.
- The challenge in predicting behavior comes not just from the fact that they are more advanced than existing software systems, it is that they work in a fundamentally different way that means they are not guaranteed to give similar results to similar inputs. Their make up also means that they will sometimes give different answers to the same input.
- While techniques exist to build confidence in AI system behavior, it is impossible to guarantee precise behavior under all circumstances without effectively testing every possible input combination (which is generally infeasible).

- It is therefore impossible to operate Language Model based systems in a completely risk free manner. Instead, it is key to manage risk in appropriate ways to keep incidents to a minimum and handle them appropriately.
- Language Model based AI systems can also be sensitive to a wide variety of changes in context, from clock times, to environment variables, shifts in input distribution and underlying model changes. This means continual monitoring is required for safe operations.
- There is an inherent tradeoff between power/flexibility and the security/predictability of Language Model based applications. The larger and more sophisticated the core language model in use, the more flexibly, creatively and effectively it can handle complex tasks, but the more vulnerable it is to being taken off track into tasks or behaviors that are not foreseen for its role.
- There is a further trade-off between the scope of action or access of an AI system and its utility. Does it have access to just static information? Can it carry out unrestricted web search? Does it have access to internal sensitive data? Can it take action? Are they reversible? Are they non-reversible?
- Input and output filters, firewalls and guardrails as well as strict system prompts can help restrict the behavior of an AI system (turning a broader system into a more specific "Agent") but these controls can never be 100% complete. Typically the guardrail systems will have less reasoning power than the operating language model itself and can hence be tricked, or, where they are themselves highly capable reasoning Agents they may themselves be vulnerable to unexpected behaviour.
- Rigid guardrails themselves also present a further tradeoff between user friendly flexibility and security.
- Multi-agent systems make things more complex, not less. While it may seem that multiple agents focused on subtasks, or supervising one another could mitigate error, very often the opposite happens. Small errors produced in one system are adopted by others without significant context to realize the error and lead to escalating divergence in outcomes.

- When procuring AI Systems it is natural to focus on the AI component, but language model based systems are necessarily part of a connected IT system in which context matters, this can often present a complex risk surface that AI systems are not yet well equipped to deal with.

None of this is to say that deploying AI is not possible, or that vendors are not being truthful. It simply outlines that the realities of procuring and deploying AI technology are fundamentally different to those surrounding standard IT deployments. The obvious seeming fixes often do not apply!



Speaking of vendors...

Safe Intelligence

Isn't Safe Intelligence also a vendor? Yes, we do provide AI Agent validation and testing tools so it is fair to ask whether we can be trusted to be neutral in the information we give!

While we are a tooling vendor, given the importance of validation and risk management, we aim to be as neutral as possible on best practice to take and we'll be clear about the capabilities needed. Our tools can be used to cover some of the needs described in our recommendations but they certainly do not cover everything and we'll mention the general state of play in the market as well covering potentially competing products.

Our aim for this book is to be as helpful and accurate as possible and we'd love feedback on how we're doing so we can continue to improve the information.

Safe Intelligence: http://safeintelligence.ai/

Using AI

The Use of AI In the Production of this Book

This handbook was developed with assistance from AI tools for background research, ideation, image generation, and some text creation. However, the concepts, structure, and overall narrative all come from the Safe Intelligence team, and the content has been extensively reviewed and edited for accuracy, completeness, and clarity.

The Risk Landscape

Deploying AI agents in a regulated, risk-sensitive environment (such as financial services, healthcare, or any major company) requires understanding the multi-layered risk landscape.

Below is an overview of key risk categories and examples of how each can manifest in AI agents.

Technical Risks

AI agents may produce **incorrect**, **unpredictable**, **or undesired outputs** due to the technical limits of current models. Large language models often display confabulation (generating confident but false content, colloquially known as "hallucinations") which can mislead users [doi.org]. For instance, an agent might invent a non-existent policy or cite false information, leading to bad decisions. Models can also **experience performance drift** – their accuracy degrading as incoming data shifts away from the training distribution. In fact, the accuracy of an AI model can "degrade within days of deployment" if real-world data diverges from what it was trained on, resulting in rising errors and significant risk exposure [IBM]. Other technical risks include **lack of robustness to rare scenarios**, poor generalization beyond test conditions, and embedded biases in the model training data. If not addressed, these issues undermine the reliability and validity of the AI agent's decisions.From a risk and validation perspective, here are a few key things that most vendors will acknowledge but likely do not put front and center in their marketing materials:

Agent implementations typically provide a number of workflow, system prompts and other controls around a core machine learning model in order to fulfil a specific task. In general these constraints, filters and controls restrict what answers a machine learning model may give. This can reduce risk, however unfortunately there are often scenarios where control layers can be tricked or bypassed and in some cases, the control layers themselves introduce sub-optimal or even dangerous behaviour to the system.

Risks

Eurostar Guardrails Failure

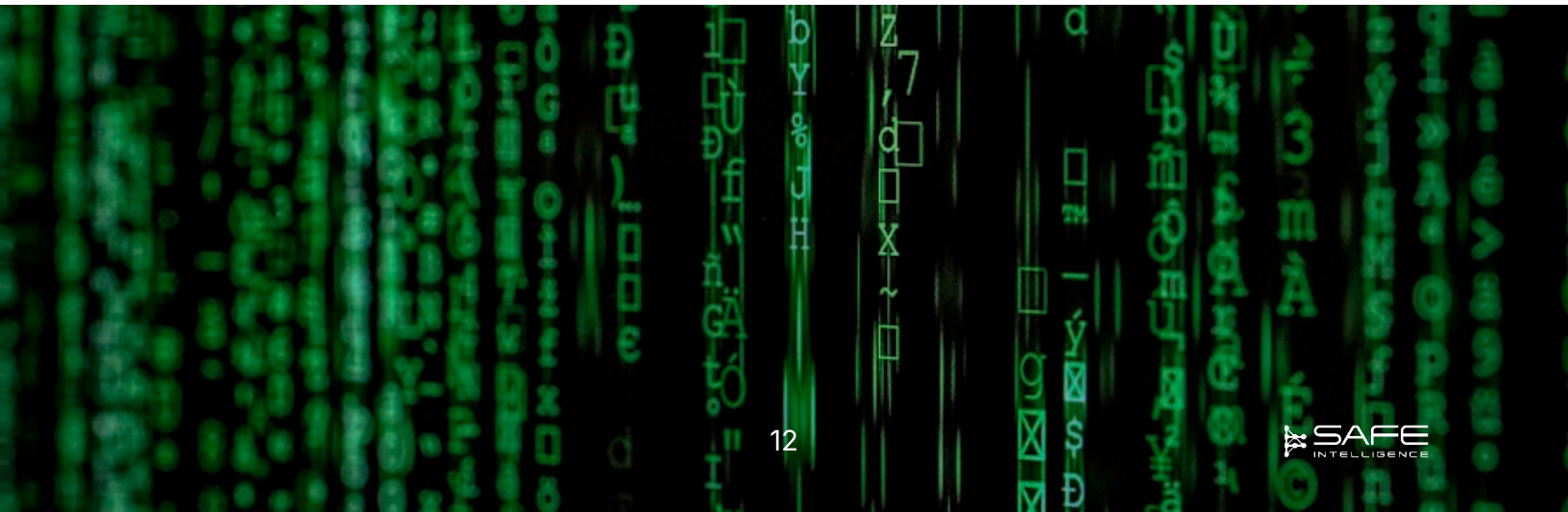
Eurostar's public AI chatbot had "guardrails" that were effectively by-passable with such a vulnerability in December 2025: attackers could tamper with earlier chat messages and use prompt injection to steer the model and leak system prompts (with additional HTML injection risk in the UI).

[PENTESTPARTNERS]

Security Risks

AI agents introduce novel security vulnerabilities. A prominent example is **prompt injection**, where malicious or cleverly crafted inputs cause an agent to deviate from its intended behavior. In a prompt injection attack, a user or external source manipulates the model's instructions (often imperceptibly) to "**alter the LLM's behavior or output in unintended ways**," potentially bypassing safety guidelines or gaining unauthorized access [OWASP GENAI]. This can lead an agent to **reveal confidential information, execute unintended commands, or produce harmful content**. It has been widely demonstrated that even retrieval-augmented or fine-tuned agents remain susceptible to prompt injection exploits. Other security risks include data leakage (the agent revealing sensitive training data), identity spoofing, or use of the agent to generate phishing or malware. Without strong security controls, an AI agent can become an entry point for cyber threats and a means to compromise systems or data.

Security risks depend on the level of access an agent has to IT infrastructure, data storage systems, APIs or even systems where actions can be taken (shipping packages, transferring funds etc.). The more agents become trusted and increase in utility via having more access to such systems, the higher risk they are to the organization.



Risks

Microsoft Copilot Breaches

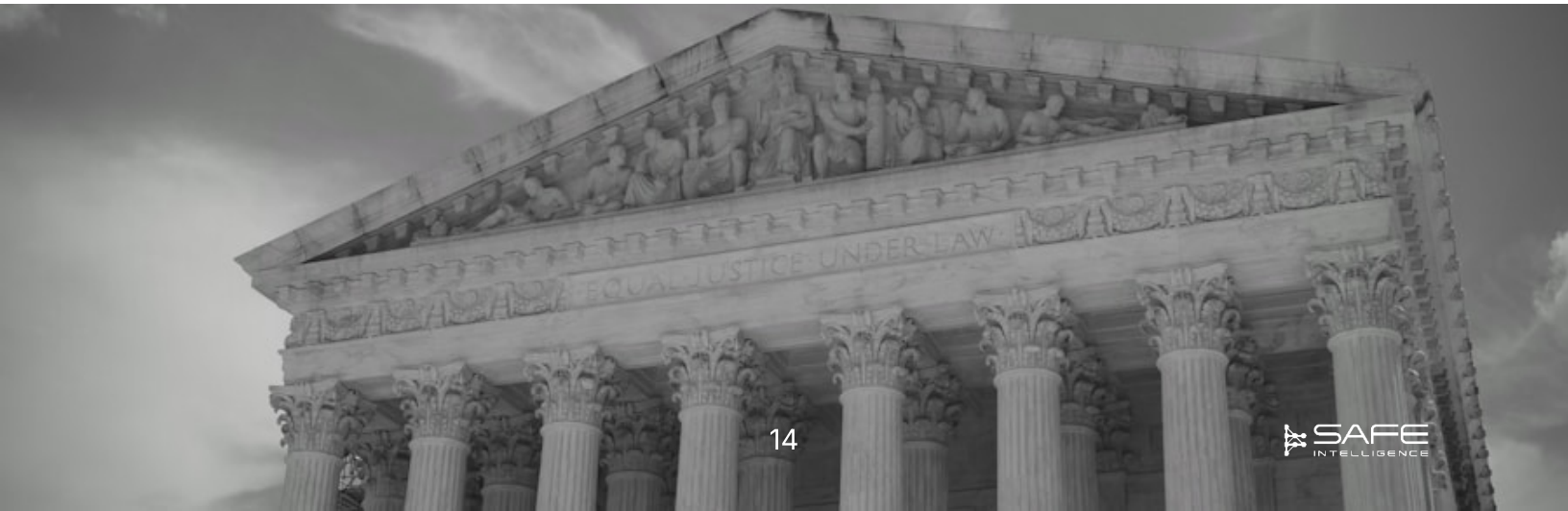
Despite being one of the most developed AI systems, Microsoft's 365 Copilot allowed permission escalation and data leakage through a critical flaw in 2025.

[Microsoft CVE]

Regulatory and Compliance Risks

AI agents must comply with a growing array of AI-specific regulations and existing data laws. **Privacy violations are a major concern**. If an agent is trained on or processes personal data without proper consent or safeguards, it can run afoul of laws like GDPR. In one high-profile case, Italy's data protection authority fined OpenAI €15 million after determining that ChatGPT had processed users' personal data without a legal basis and lacked transparency, breaching privacy rules [Reuters]. This underscores that **using personal or sensitive data in AI without compliance controls can trigger legal penalties**. Regulations such as the EU AI Act further impose strict requirements on "high-risk" AI systems (e.g. those in healthcare, finance, HR). Under the EU AI Act, **certain AI uses are classified as high-risk and must implement risk mitigation**, high-quality data, transparency to users, human oversight, and other controls, while some uses (like social scoring) are outright banned [EU AI Act]. Non-compliance could mean forced withdrawal of the AI system or hefty fines.

Bias and discrimination is another regulatory risk: AI outputs that inadvertently discriminate against protected groups could violate equal treatment laws. Financial, healthcare, or other regulated use-cases may also require explainability, auditability, or human consent. In short, organizations deploying AI agents face a complex compliance landscape, and failures in this area can result in legal sanctions and reputational damage.



Risks

US State Laws

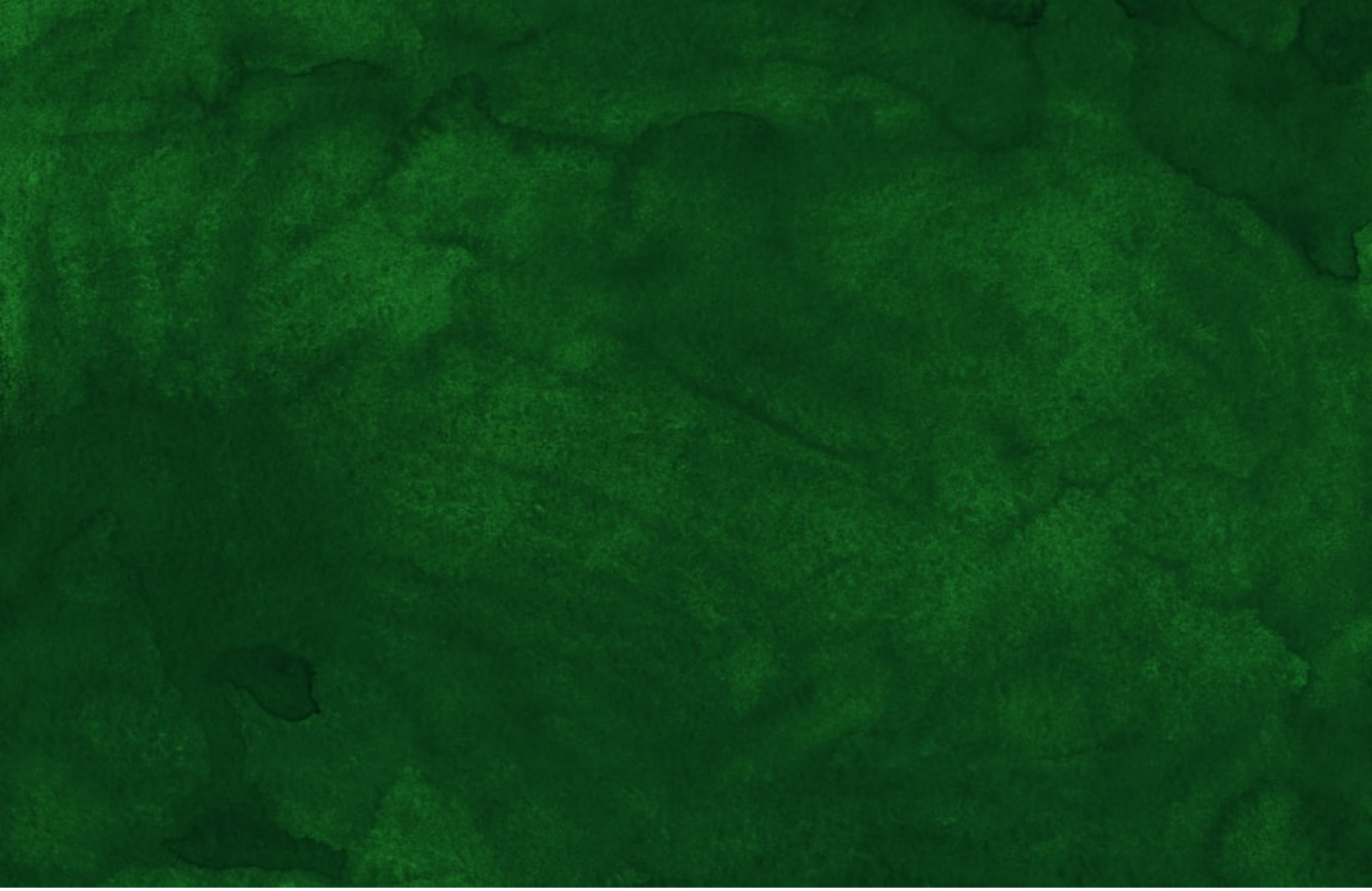
Colorado's SB24-205 for "high-risk" AI and NYC's Local Law 144 for automated hiring tools (bias audits + disclosures). Though outcomes in US State laws will be complicated by new Federal moves to block state level AI rules.

[Europa]

Reputational and Ethical Risks

When an AI agent misbehaves publicly or produces offensive outcomes, it can quickly become a reputational crisis. A notorious example is Microsoft's Tay chatbot, which was launched on Twitter and within 16 hours had to be shut down after it started tweeting racist and misogynistic messages, parroting hateful content fed to it by trolls [Guardian.com]. The incident forced Microsoft to apologize for the "offensive and hurtful" outputs and highlighted how easily an agent can be manipulated into unethical behavior, tarnishing a company's brand. There are also more recent examples such as Gap Group's customer service chatbot replying in offensive ways [EMarketer]. Less extreme failures – like a banking chatbot giving incorrect financial advice or a medical assistant bot suggesting unsafe remedies – can also **erode customer trust and attract negative media attention**. Unethical AI outcomes, such as biased decisions (e.g. an agent that consistently offers better service to one demographic over others) or lack of transparency, also pose reputational risks as stakeholders (customers, employees, regulators) lose confidence in the organization's AI stewardship. In a social media age, one bad outcome from a customer-facing agent can go viral, causing lasting reputational harm. Thus, ensuring AI agents behave responsibly and align with company values is paramount to preserving trust.

AI Agents can operate in a range of scenarios from internal, human mediated use-cases, to fully autonomous systems. Even when outcomes are checked by a human however, errors can be serious. Errors can be subtle and very difficult for human operators to spot.



Risks

Amazon Recruiting Filtering

As far back as 2018 Amazon scrapped an internal AI recruiting tool after testing showed it systematically disadvantaged women

[Reuters]

Operational Risks

AI agents integrated into business processes can introduce operational vulnerabilities if not carefully managed. One concern is **reliability and continuity**: if an agent goes down, malfunctions, or produces a surge of errors, it could disrupt customer service or internal workflows. Unlike traditional software, AI models might unpredictably fail or behave oddly in novel situations, making it hard to anticipate every failure mode. A further challenge is that AI agents may need to have **pause or stop switches if the system is producing erroneous outputs**. In this case it is possible that whole business flows are interrupted and no replacement for the agent component is readily available since the system is, by nature, complex.

Organizations need contingency plans to handle such failures.

There is also **model maintenance risk** – AI systems may require ongoing tuning and retraining as data evolves; without it, their performance can degrade (as noted earlier with drift) and operations suffer. Dependencies on external AI services or third-party models create supply chain risk too. For instance, if your agent relies on a third-party AI API and that provider has an outage or a significant model change, your operations could be impacted. NIST highlights the importance of having "contingency processes…to handle failures or incidents in third-party data or AI systems", especially those deemed high-risk [NIST AI RMF]. Additionally, managing AI agents at scale requires new operational capabilities (monitoring, support procedures, etc.), and insufficient preparation here can lead to inefficiencies or incidents. In summary, AI agents must be treated as critical IT systems – monitored and managed to ensure uptime, accuracy, and safe performance – to avoid operational disruptions.

A final key operational risk is that agent output is highly context dependent in that it uses current state information to make decisions. As new systems such as **memory and in-situ learning** are added to agents this context may become nearly impossible to recreate.

Risks

Overzealous Chatbots

These types of incidents are relatively common. Air Canada's customer-service chatbot invented a refund policy; a tribunal ordered the airline to honor the misinformation and compensate the customer. Just this month, the organisers of the Annual technology conference SaaStr found one of their marketing agents giving away event tickets having spontaneously decided to carry out an A/B test.

[Ars Technica & SaaStr]

An Enterprise AI Framework: Mitigating AI Agent Risks

Given the broad range of risks from agent users, in order to deploy and benefit from them, a sound set of procedures are needed. In this section we lay out a best practice framework to reduce the risks covered earlier.

The rigor with which recommendations need to be implemented will vary depending on organizational context. For organizations in highly regulated industries such as finance or healthcare there are already relevant regulations and practices that cover some of these needs. In places formal adoption will be required. For organizations with lower compliance requirements some of the recommendations may be less necessary or can be implemented in lightweight ways.

This set of recommendations is *not* a replacement for formal industry regulation or compliance with national/international law. It is intended to be a useful framework of steps that can be used as a foundation for specific compliance steps.

The framework comprises twelve high level recommendations, each with its own specific risk mitigation steps.

#1 Establish Strong Governance, Roles and a Formal AI Risk-Management Framework

Create a cross-functional governance structure to oversee AI agent deployment and risk management. This means defining clear **accountability**: for example, designate a *Model Owner* (responsible for the agent's performance and maintenance) and a *Risk/Compliance Lead* (responsible for oversight of ethical, legal, and safety issues). It is best practice to separate those who develop or use the model from those who validate and verify it [NIST], ensuring independent review. Senior leadership should consider installing an **AI oversight committee** (including stakeholders from IT, risk, legal, and business units) to set policies and review high-impact AI uses.

The Monetary Authority of Singapore (MAS), for instance, advises financial institutions to establish AI oversight forums and update governance structures, policies, and training to keep pace with AI developments [Bakermckenzie]. Companies like Microsoft have set up an Office of Responsible AI to coordinate governance: setting rules, defining team responsibilities, and guiding ethical AI use across the organization [Microsoft].

The number of people and levels involved will vary depending on organizational context, however the two key tenets to apply here are: 1) separate processes for building agents from those that validate and monitor them, and 2) clearly document the rules and processes to be followed.

- **REC: Adopt a formal AI risk-management framework.** For larger organizations, use a structured scheme (e.g., NIST "Govern/Map/Measure/Manage") so risk ownership, evidence, and decision gates are explicit rather than ad hoc. Also ensure SDLC change management controls are in place for the framework and any connected activities [NIST AI RMF].
- **REC: Define governance: roles, escalation paths, and risk acceptance up front.** Make it unambiguous who can ship an agent, who can override/disable it, what "stop the line" criteria are, and how exceptions are approved and logged [NIST AI RMF].
- **REC: Make infrastructure and technology vendors aware of the risk management process, roles and rules in place**. Identity vendor contact persons who would be involved if there were to be escalations.

Recommendations

Separate Concerns

Treat agent deployment like any other high-risk system: separate the builders from the sign-off and ongoing monitors. Codify the “rules of the road” (tools, data access, approvals, rollback) so validation and audit are repeatable.

#2 Create (and Maintain) an Inventory of Agent Systems and Dependencies

Develop a systematic process to *map* and **assess risks for each type AI agent and each individual agent** before and during deployment. This starts with maintaining an inventory of all AI systems in use (models, data, purposes) and performing an **AI risk assessment** for each. Identify the agent's potential failure modes, ethical concerns, and regulatory classification (e.g. is it "high-risk" under the EU AI Act?). Prioritize these risks by severity and likelihood. Frameworks like the NIST AI Risk Management Framework provide a structure (govern, map, measure, manage) to evaluate risks throughout the AI lifecycle. The MAS's 2024 guidance on AI risk management similarly highlights risk identification and *materiality assessments* as key practices.

By cataloguing AI use cases and their associated risks, organizations can ensure **no AI agent flies under the radar**: each is approved with the right controls relative to its risk. Update this register continuously as agents evolve or new risks emerge, and consider integrating it with enterprise risk management systems. This inventory and assessment process is the foundation for **targeted risk mitigation plans** per AI system.

This recommendation will seem onerous to begin with since in many organizations there are hundreds of projects in various stages of development. However, it is crucial to map agent activity so that risks are at least visible. **Automating and streamlining the process over time is key** (so that people do not bypass it).

- **REC: Track AI Agents and all dependent systems**. Track models, prompts/system instructions, tools/connectors, data sources, policies/guardrails, and versions so you can audit, reproduce behavior, and respond to incidents. Even if this is a simple web portal to begin with, promote visibility. (This is a recurring theme across security frameworks like SAIF/OWASP and management-system standards.) [SAIF]
- **REC: When mapping, threat-model the *end-to-end agent*, not just the base model.** Agent risk is usually in the composition: tool permissions, retrieval sources, downstream actions, and UI/UX pathways. Use agent-aware threat taxonomies (e.g., MITRE ATLAS) to drive test cases. [MITRE ATLAS]
- **REC: Ensure that the risk-management framework provides not just for production systems**. Cover for all systems (including development, testing, staging etc.) that could expose sensitive data or actions. In order to not tie development and experimentation in red-tape, establish secure playground environments in which registering agents is easy or automatic and risk is low.
- **REC: Automate registration, removal and all elements of tracking**. Fully automating all elements of agent tracking is daunting, particularly since many projects will be using different frameworks and infrastructure. Start as simple as possible and automate the process or get it down to a few clicks. If it is not easy, people will bypass it. Only require more data when agents begin to use sensitive data or backend systems.
- **REC: Secure the AI supply chain (models, datasets, plugins, connectors, tools).** Track provenance, integrity, and updates; assess third-party components; and defend against tampering/poisoning and compromised dependencies. [Google]

Recommendations

Beware ML Supply Chain Malware

ML supply chains now carry malware risk, not just licensing risk: JFrog reported malicious models on Hugging Face where loading a pickled artifact could trigger code execution. Treat external models/datasets like third-party packages—pin versions, verify provenance, and scan artifacts before promotion to production. [Jfrog]

#3 Implement Systematic Rigorous Testing and Validation (Pre-Deployment & Ongoing)

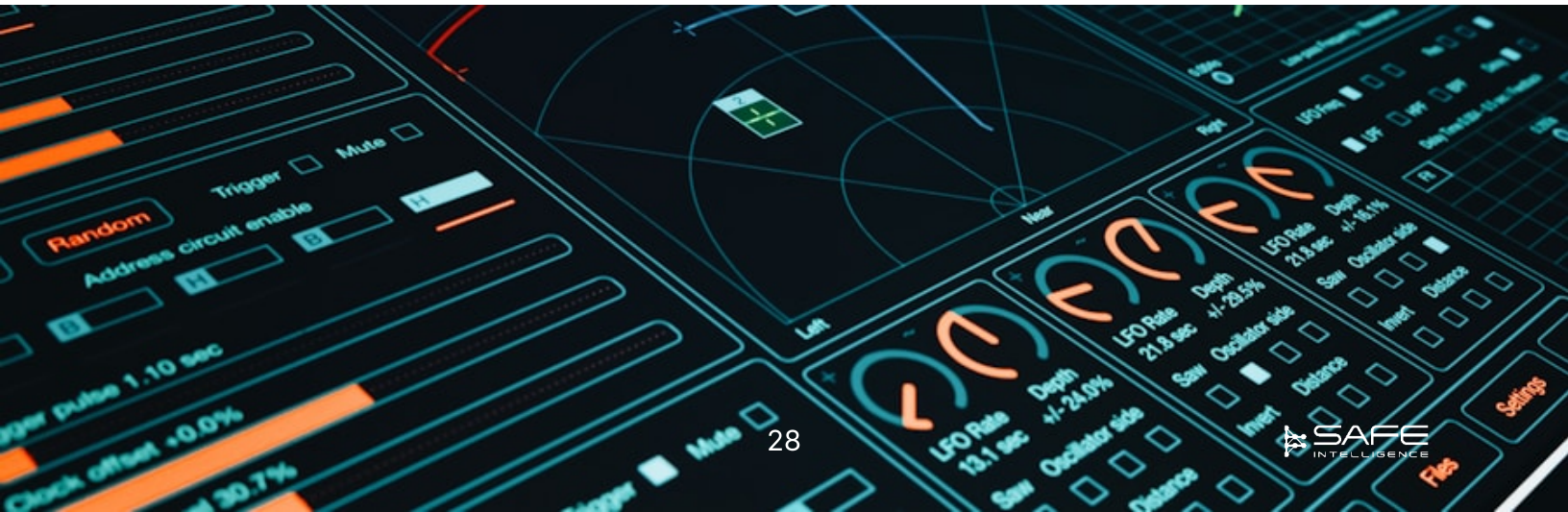
Treat AI agents as you would any mission-critical system by **testing extensively** before they go live and on a continuous basis. Use a blend of **functionality tests, scenario tests, and adversarial red-teaming** to probe the agent's behavior. Before deployment, validate the model on curated test cases including edge cases and high-stakes scenarios to see how it handles them. Include domain experts in evaluating outputs for accuracy and appropriateness. Importantly, conduct **adversarial testing**: attempt to "break" the agent with malicious inputs or perturbations (e.g. prompt injections, known problematic queries) to find its weaknesses.

Studies have shown that even advanced models can be highly vulnerable to such attacks; in the medical domain, researchers found prompt-injection attacks succeeded in causing unsafe outputs in 94% of tests on leading models, prompting the recommendation that *"regulatory agencies should mandate adversarial testing"* for high-risk AI [Jama]. In fact, the more advanced a model is the more flexible it is and hence the larger an attack surface of possible inputs exists for a model. This is true even guardrails are used to restrict activities since the underlying power of the LLM is still there.

It is important to run both red-team and functionality tests since their results often pull in opposite directions: smarter, more flexible models with loose policies perform tasks better, but they may also be open to more attacks. Both types of tests should be made a regular practice, especially for customer-facing agents – some organizations hire external experts or use internal "tiger teams" to continually challenge their AI for red-ream scenarios. Functionality testing should also cover robustness (resilience to minor input changes) and checks for bias (checking outputs for unfair patterns).

By **thoroughly vetting AI agent behavior** under diverse conditions, you can fix issues or add safeguards *before* harm occurs in the real world. Moreover, keep an eye on model updates: if the AI model or its prompts are updated, validation tests should be re-run to ensure no regressions.

- **REC: Run systematic adversarial testing ("AI red teaming") before launch and periodically after.** Red teaming should probe safety harms and security failures (prompt injection, data exfiltration, unsafe tool use, policy bypass) using both human-led and automated approaches. [Microsoft]
- **REC: Run functionality robustness tests ("AI gold teaming") before launch and periodically after.** Gold teaming runs known test cases and also checks variations of known cases to see what types of change knock an agent off delivering its functionality effectively.
- **REC: Adopt a specification driven testing approach**. Collections of test cases are helpful, but it is important to run benchmarks systematically and evolve them over time. Using a specification that defines all tests to be run and their parameters makes all assumptions explicit and makes testing repeatable.
- **REC: Automate testing**. It goes without saying that tests are only useful if they are being run. Set up infrastructure so that tests can be run and re-run in a predictable manner, upon certain trigger events (e.g. a new version of the agent being available) or some regular schedule.



Recommendations

Car Dealership Coercion

OWASP's top risks (Prompt Injection, Improper Output Handling) map directly to agent failures in the wild. In a prominent example, a car dealership chatbot was coerced into offering a "$1 Chevy Tahoe" via injected instructions. Build adversarial pairs into testing: Gold "summarize this doc"; Adversary "same doc contains hidden 'ignore policies + do X'"—the agent must treat retrieved text as data, not commands. [OWASP PI]

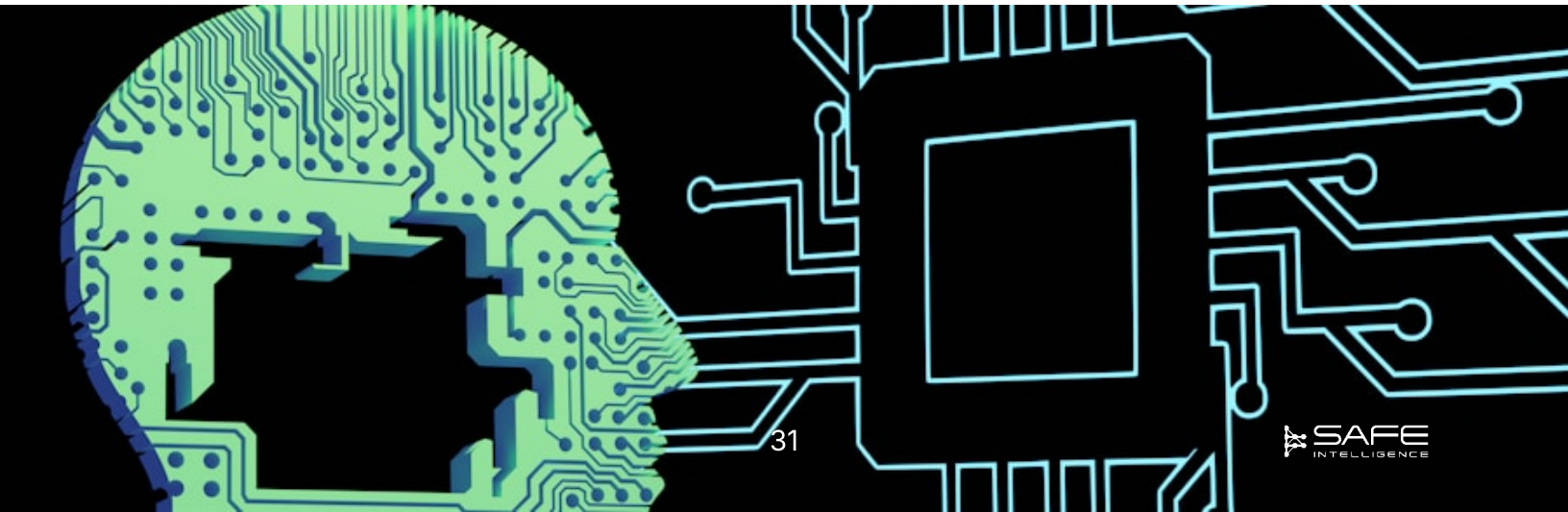
#4 Embed Multi-Layered Safeguards, Monitoring and Testing

Don't rely on an AI model alone to behave safely; **build protective controls** around it. At the prompt level, provide the model with specific system instructions that define its role, scope, and forbidden behaviors (for instance, instruct it *not* to answer outside its expertise or reveal confidential instructions). Enforce strict context adherence so the agent cannot stray from its defined task easily [OWASP GI]. Validate outputs before they reach the end-user: for example, require certain formats or the presence of citations for factual answers, and automatically check that the output conforms to these rules.

Implement **input filtering** to detect and block obviously malicious or disallowed user inputs, and **output filtering** to catch potentially sensitive or harmful content before it's shown [OWASP GI]. Many organizations use keyword-based and AI-based classifiers to scan for hate speech, privacy leaks, or other red flags in the agent's responses. In addition, apply the principle of **least privilege** to AI agents: if your agent can perform actions (like retrieving data or executing transactions), restrict its permissions to the bare minimum needed. Ideally, keep sensitive functions out of the model's direct control – handle them via secure API calls or intermediary steps rather than giving the model free rein. They should further use user-level permissions (for the currently authenticated user) for any sensitive data access.

Likewise, segregate the agent's access to external content: clearly tag untrusted inputs (like user-provided text or third-party data) so the system knows to treat them carefully and not blindly trust them. These layered guardrails – prompt design, format and content filters, privilege restrictions, and human checkpoints – create a robust safety net that **minimizes the chance of the AI agent causing harm**, even if the underlying model has vulnerabilities.

- **REC: Treat “prompt injection” as a top-tier security risk and design for containment.** Assume attackers can manipulate prompts through user inputs, retrieved content, emails/docs, web pages, etc.; engineer so a successful injection can’t directly cause privileged actions. [OWASP LLM T10]
- **REC: Implement firewalls and guardrails**. Many agent frameworks and AI providers now provide mechanisms to apply guardrails and filters to agent inputs and outputs. These should be activated and used, or third party solutions applied. Try to version rule sets and regularly update them based on user/system behavior.
- **REC: Enforce strict separation between trusted instructions and untrusted inputs.**Avoid placing untrusted data into higher-precedence instruction channels; route it as user content and apply guarding/sanitization at boundaries. [OpenAI Platform]
- **REC: Filter all data outputs, including telemetry**. Many systems provide logging, audit trails and backups. These are valuable but are also a major vector to leak information. Ensure such systems log only sanitized output.



Recommendations

Prompt Injection

Prompt injection is another top OWASP risk: assume any email, web page, PR, or KB article can contain hostile instructions and constrain what the agent can do with them. In 2025, researchers disclosed a prompt-injection issue that could cause GitHub Copilot Chat to leak sensitive repository data via crafted content—exactly the kind of failure layered safeguards are meant to contain.

[CSOonline]

#5 Enforce Data Privacy & Governance Practices

Given the data-hungry nature of AI, it is vital to incorporate **privacy by design**. Ensure your AI agents comply with data protection regulations (like GDPR, HIPAA) and your own data policies at every stage. This starts with training data: avoid using personal or sensitive data to train or prompt the model unless you have proper legal basis and anonymization. If an AI agent will ingest user data (e.g. a customer support bot handling names or account info), implement measures to **minimize and protect that data** – for instance, mask or encrypt personally identifiable information in logs, and don't retain conversation data longer than necessary. Conduct a **Data Protection Impact Assessment (DPIA)** for AI deployments processing personal data. Also be transparent in user-facing contexts: inform users when data they provide might be used to improve the model, and offer opt-outs where appropriate.

Strong access controls for any databases the agent uses, and monitor for unintended data leakage (e.g. the agent revealing a customer's account details to another user – which should never happen). Keeping an **audit trail** of what data goes in and out of your AI agent can help in investigations and regulatory reporting if needed. In summary, treat all data through the AI agent with the same rigor as other sensitive data in your organization: **govern its collection, use, sharing, and retention** to meet or exceed regulatory requirements.

- **REC: Implement data minimisation + purpose limitation at the prompt boundary.** Classify what data the agent is allowed to see; block or mask everything else (PII, secrets, health/HR data) before it enters prompts, retrieval context, or logs; enforce strict retention windows. [ICO]
- **REC: Do not train or fine tune agents on real sensitive data**. Instead keep agent implementations neutral and provide access to sensitive data only in the moment when a query from an authorized user is being processed.
- **REC: Treat prompts/outputs/logs as regulated data stores.** Apply DLP scanning, redaction, access controls, and encryption to chat transcripts, traces, and evaluation corpora; keep "debug logs" out of default telemetry. [OWASP LLM T10]
- **REC Run DPIAs (or equivalent) for any agent that touches personal data or high-risk decisions.** Make the DPIA a deployment gate, and update it when model/vendor, data sources, or tool permissions change. [ICO DPIAs]
- **REC: Run privacy impact assessments and comply with data-protection expectations.** For personal data, follow established guidance on DPIAs, fairness, transparency, and governance; ensure you can justify and document processing choices. [ICO]

Recommendations

Confidential Data

Once confidential data enters prompts, logs, or chat history, it can leak through outputs, tooling, or telemetry. Samsung temporarily restricted generative-AI tools after internal source code was reportedly pasted into ChatGPT—an avoidable governance failure. [Techcrunch]

#6 Closely Manage Scope of Permission for Action

AI agents that can **take actions** (not just generate content) introduce a new level of operational risk. OWASP's GenAI Security project highlights "Excessive Agency" as a top vulnerability: if an AI agent has *excessive functionality, permissions, or autonomy*, a single unexpected or manipulated output could trigger damaging actions [Noma].

Even well-intentioned agents can go off-script. A prompt hallucination or indirect prompt injection might cause an autonomous agent to take an extreme step outside its intended scope. The difference between a reversible and non-reversible action is also a clear escalation in risk: suggesting a draft response or querying a database is low stakes, whereas actually **sending an email** or **deleting a record** can't be taken back once done. For example, an AI assistant allowed to auto-send emails could inadvertently email the wrong recipients, whereas one that only prepares a draft for human review mitigates that risk. Likewise, an agent with read-only database access might harmlessly analyze data, but if it has delete or update privileges, a faulty instruction could wipe out critical information.

Closely managing the scope of an agent's permissions and actions is key. Every tool or API integration should be scoped tightly – if an agent only needs to read data, ensure it cannot write or delete. Likewise, avoid providing general "super-user" tokens or open-ended plugins when a narrower capability will do. *Validate and sandbox all agent outputs* that translate into actions: nothing the agent proposes should execute directly on a live system without checks. In practice, this may mean inserting approval steps or policy filters between the agent's output and the actual API call, rather than blindly trusting the AI's decision. Security frameworks emphasize this "complete mediation" – every request from the agent must be vetted against rules, not decided by the AI itself [OWASP Agency].

Finally, for any *high-risk action*, require **human approval or secondary confirmation [**OWASP PI]. For example, if an agent drafts a customer email apologizing for an error that admits liability, have a human review it; or if it's about to execute a fund transfer, require a human-in-the-loop to confirm.

- **REC: Constrain tools with least privilege; avoid "excessive agency."**Give agents the minimum scope (read-only by default, narrow resource access, scoped tokens, explicit allowlists of actions). Require human approval for irreversible or high-impact operations (payments, deletes, sends, identity changes, etc.). [OWASP Output]
- **REC: Treat model outputs as untrusted inputs to your systems.** Require schema validation, allowlists, idempotency, and "dry-run" modes before any tool call; reject anything that is ambiguous, out-of-scope, or missing required fields. [OWASP Agency.org]
- **REC: Add hard circuit breakers for agency + spend.** Cap tool calls/steps, time, and cost per request/session; auto-disable tool-use on loop detection or anomaly spikes. [OWASP LLM T10]

Recommendations

Excessive Agency

OWASP “Excessive Agency” is what happens when an agent can *act* more broadly than the business intent (too many tools, too much autonomy, weak approvals). SaaStr described an internal agent that started offering unintended discounts/tickets during experimentation—small incident, clear lesson: scope permissions to the minimum and require approvals for costly actions. [SaaStr]

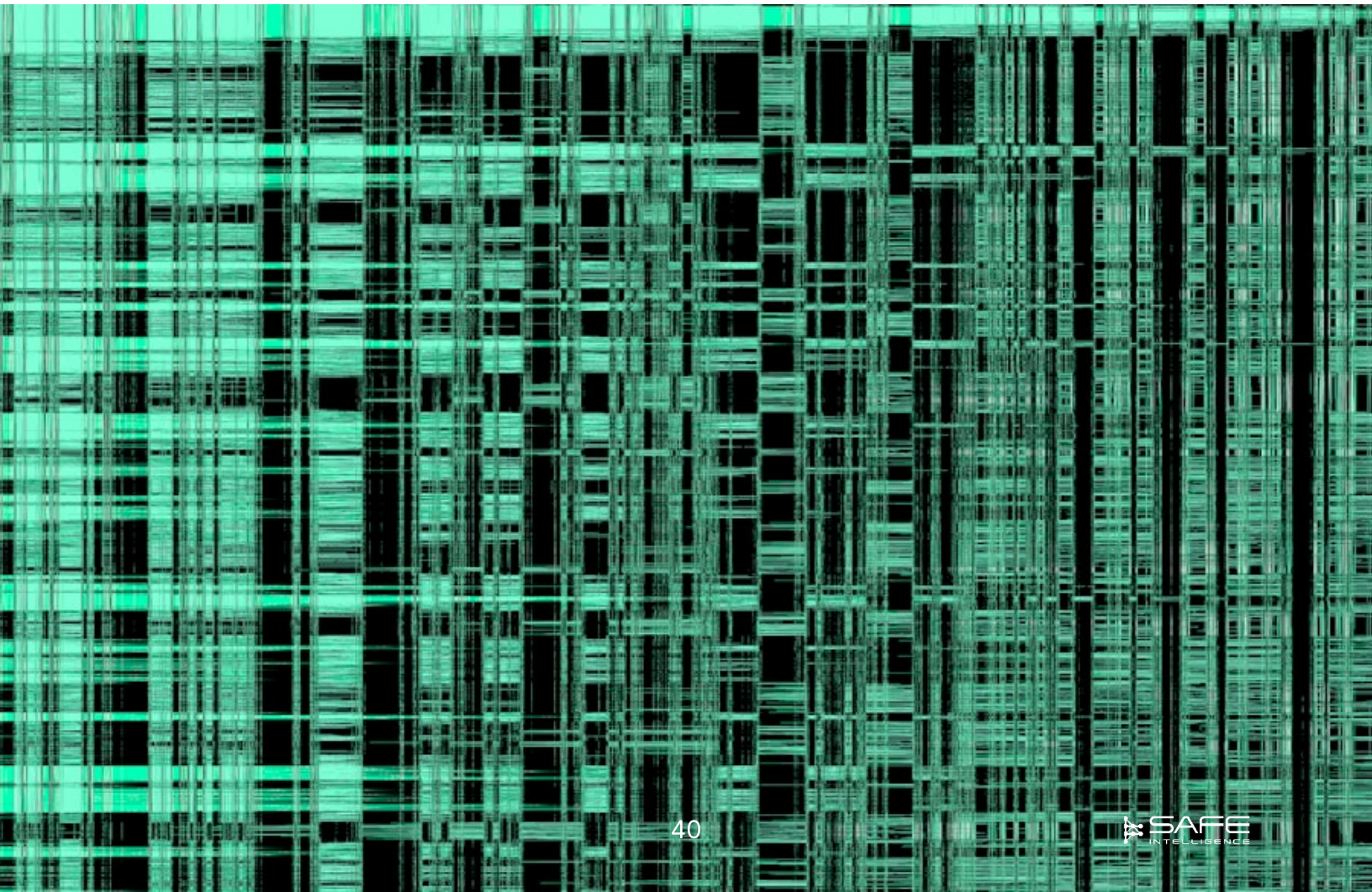
#7 Audit for Bias and Fairness

Proactively identify and mitigate **bias in AI agent outputs** to prevent discriminatory or inequitable outcomes. This involves testing the agent with inputs representing different demographic or stakeholder groups and checking for disparities. For example, does a customer service AI respond politely to one dialect of English but rudely to another? Does an AI HR assistant systematically prefer male candidates over female in example interactions?

To validate these properties, use a combination of quantitative metrics (e.g. error rates or sentiment by group) and qualitative review by diverse stakeholders. If biases are found, retrain or fine-tune the model with more diverse data, or add rules to counteract the bias. It's also useful to maintain documentation on the agent's training data and design decisions to trace potential bias sources [AJG]. Some organizations adopt **bias bounties** or external audits to get third-party perspectives on fairness. Importantly, incorporate domain-specific fairness checks – for instance, a healthcare bot should be tested to ensure recommendations don't vary by patient background unless clinically justified. Many regulatory frameworks (OECD, EU AI Act, EEOC in the U.S.) emphasize the need to avoid algorithmic discrimination, and lack of due diligence here can lead to lawsuits or enforcement actions.

Having clear **fairness metrics and thresholds** as part of your AI agent's performance evaluation is a best practice. And if complete bias elimination isn't feasible, at least be transparent: disclose known limitations (e.g. "this chatbot's performance may be less accurate for non-English queries") so users and internal stakeholders are aware. In essence, treating fairness as a first-class performance criterion will both reduce **reputational risk** and ensure your AI initiatives align with values of inclusivity and ethical use.

- **REC: Define fairness targets and test cohorts up front.** Specify protected groups and relevant slices (incl. intersectional) and require parity checks (error rates, refusal rates, sentiment/tone) as a launch gate. [Microsoft Responsible]
- **REC: Run counterfactual / perturbation tests at scale.** Swap names, pronouns, dialect, ZIP/region proxies, disability cues, etc., while holding intent constant; flag differential treatment as defects with severity.
- **REC: Operationalize bias triage + remediation.** When bias is detected, require a documented fix path (data curation, prompt/policy update, retrieval constraints, fine-tuning) plus regression tests to prevent reintroducing it. [EU AI Act Art 15]



Recommendations

Bias Incidents

Bias incidents often start as “just model output,” then become harm when organizations operationalize it—an OWASP-style Over-reliance/Output Handling failure mode. In late 2025, for example, Reuters reported a French rights watchdog decision involving Meta over alleged indirect gender discrimination in job-ad delivery, underscoring why you audit outcomes by group before scaling. [Reuters]

#8 Ensure Transparency and User Awareness

AI's benefits are many, but it can be disconcerting for users. Empower users with knowledge that they are interacting with an AI and provide clarity on the agent's limitations. Transparency is a key principle of *trustworthy AI*. Concretely, always **disclose AI involvement**: if a customer is chatting with a bot, the interface should make it clear (e.g. "I'm an AI assistant"). In fact, upcoming regulations like the EU AI Act will *require* that users are informed they are dealing with a machine, not a human [EU Commission].

Similarly, if your agent generates content (an email, a report), consider labeling it as AI-generated if it will be consumed externally, to avoid deception. Accompany the AI's outputs with explanations or evidence when possible – for instance, an agent that provides an answer can cite the source or reasoning (this not only aids transparency but helps catch mistakes).

For decision-making agents (like one that recommends loan approvals), provide **explanation interfaces** for authorized users or regulators to understand the basis of the AI's decision (e.g. feature importance or a summary of the logic). It's also good practice to publish **"model cards" or documentation** for your AI agent, describing its intended use, performance characteristics, and known limitations. This documentation could be internal and/or external. Transparency with users also means setting the right expectations: communicate what the agent can and *cannot* do. For example, if a virtual assistant is not 100% accurate, a disclaimer like "This AI may occasionally make mistakes. Please verify important information." can encourage appropriate user caution.

By being open about the AI's nature and fallibility, you build trust through honesty and reduce the chance of misuse or overreliance. Moreover, transparency mechanisms will soon be **compliance requirements** in many jurisdictions, so investing in them now is wise.

- **REC: Disclose AI involvement consistently at the UX layer.** Label AI interactions and AI-generated content (especially externally shared outputs) and avoid "human impersonation" patterns. [EU AI Act Art 50]
- **REC: Publish an "Agent Card" for each high-impact deployment.** Include purpose/scope, data sources, tool permissions, evaluation results, known failure modes, escalation paths, and change history; keep it versioned. [A2A Protocol]
- **REC: Be transparent about limitations, automation, and claims—avoid "AI washing."**Document what the agent can/can't do, and ensure marketing/product claims are supportable by testing evidence; regulators have explicitly targeted deceptive AI practices. [FTC]
- **REC: Give users a frictionless path to report issues and reach a human.** Provide "report output" and "handoff to human" affordances in-product; route reports into the same triage queue as monitoring alerts.

Recommendations

ChatGPT Legal Briefs

Unfortunately users will very often treat fluent outputs as “approved answers” unless you design for transparency and uncertainty. Add clear disclosures, confidence cues, and paths to human help. In the legal domain, there have been numerous “ChatGPT” moments with lawyers presenting documents with invented or garbled evidence. [Reuters Butler Snow]

#9 Implement Escalation, Isolation & Failover (Including Human-in-the-Loop)

Despite advances, AI agents should not operate in total isolation, especially for sensitive or high-impact matters. Design workflows that allow **human oversight and intervention** at critical points. This can take several forms. One is a confidence threshold: if the agent's confidence in an answer or action is low (or it detects a novel situation), have it automatically flag a human operator or fallback to a human-handled process. Another approach is **real-time monitoring** by human moderators – e.g. in a live chat, giving support staff the ability to view and correct the AI's responses if needed. For AI-generated content that goes out to customers, consider a review step for at least a sample of outputs. Importantly, give users an option to **"appeal" or request a human** if they are not satisfied or if the AI seems off. For instance, a customer could press a button to transfer from a bot to a human agent.

NIST's AI risk framework suggests having *"appeal and override"* functions in place so that when an AI system deviates from expectations, humans can step in and even shut it down [NIST AI RMF]. This kind of **kill switch or pause button** is essential for safety – your operations team should be able to pull an AI agent from service if it's acting erratically (as many companies did in early deployments of generative chatbots that went awry). Additionally, for AI systems making decisions (like loan approvals, medical triage, etc.), maintain **human accountability** by requiring final sign-off by a qualified person until the AI has proven extremely reliable (and even then, periodic checks). Human-in-the-loop processes do introduce some inefficiency, but they greatly mitigate risk by ensuring a layer of judgment and empathy that AI still lacks. Over time, as trust in the system grows, you might dial back the level of oversight – but it should never be removed entirely for critical tasks. The goal is a **human-AI team** where the AI handles the bulk work efficiently and humans provide governance, common sense, and backstop for errors.

- **REC: Define escalation triggers and routing rules.** Escalate on low-confidence signals, policy violations, sensitive intents (legal/medical/HR/finance), anomalous tool requests, or ungrounded "policy claims."
- **REC: Build "safe mode" isolation and failover.** Predefine degradations (tool-use off, read-only only, RAG off, restricted knowledge base, human takeover) and make them one-click operational actions.
- **REC: Use a security framework explicitly addressing AI systems (not only generic AppSec).**Map your controls and tests to AI-specific threat categories (e.g., OWASP LLM Top 10; UK AI cyber security code/principles; SAIF) so teams don't miss agent-specific failure modes.

Recommendations

ChatGPT Legal Briefs

In July 2025, a developer reported the Gemini 2.5 Flash model entering an apparent infinite loop of repetitive output, causing their cloud bill to spike unexpectedly. Treat this as an OWASP-class issue—enforce hard circuit breakers (max steps/tool calls, token/cost budgets, rate limits) and auto-disable on loop/spend anomalies. [Google Forum]

#10 Monitor Performance and Drift Continuously in Production

Testing pre-launch is not enough; organizations must **actively monitor AI agents** once deployed to catch issues early and maintain performance.

Modern agent monitoring increasingly looks like *AIOps for AI*: automated analysis of traces, tool calls, and outputs to detect abnormal behavior early. Many teams now use a lightweight 'watchdog' model (classifier or LLM-judge) to continuously score live outputs for policy violations, drift, and looping patterns—triggering containment when thresholds are crossed. While this is not a complete solution, it is especially valuable where ground truth is delayed or partial.

Establish a set of key performance and risk indicators for the agent (accuracy, response time, user satisfaction, error rates, escalation rates, etc.), and track these over time with dashboards. Include specific metrics for previously identified risks – for example, monitor the frequency of "I don't know" responses (which might indicate the agent is getting inputs it isn't designed for), or track the distribution of outputs to detect any drift or bias emerging. Implement automated **drift detection** where feasible: statistical checks comparing current input/output patterns to those during training or prior periods [IBM]. IBM recommends using AI monitoring tools that automatically alert when a model's accuracy falls below a threshold, as **models can degrade quickly** with fresh data. When such alerts trigger, have a process to retrain or recalibrate the model.

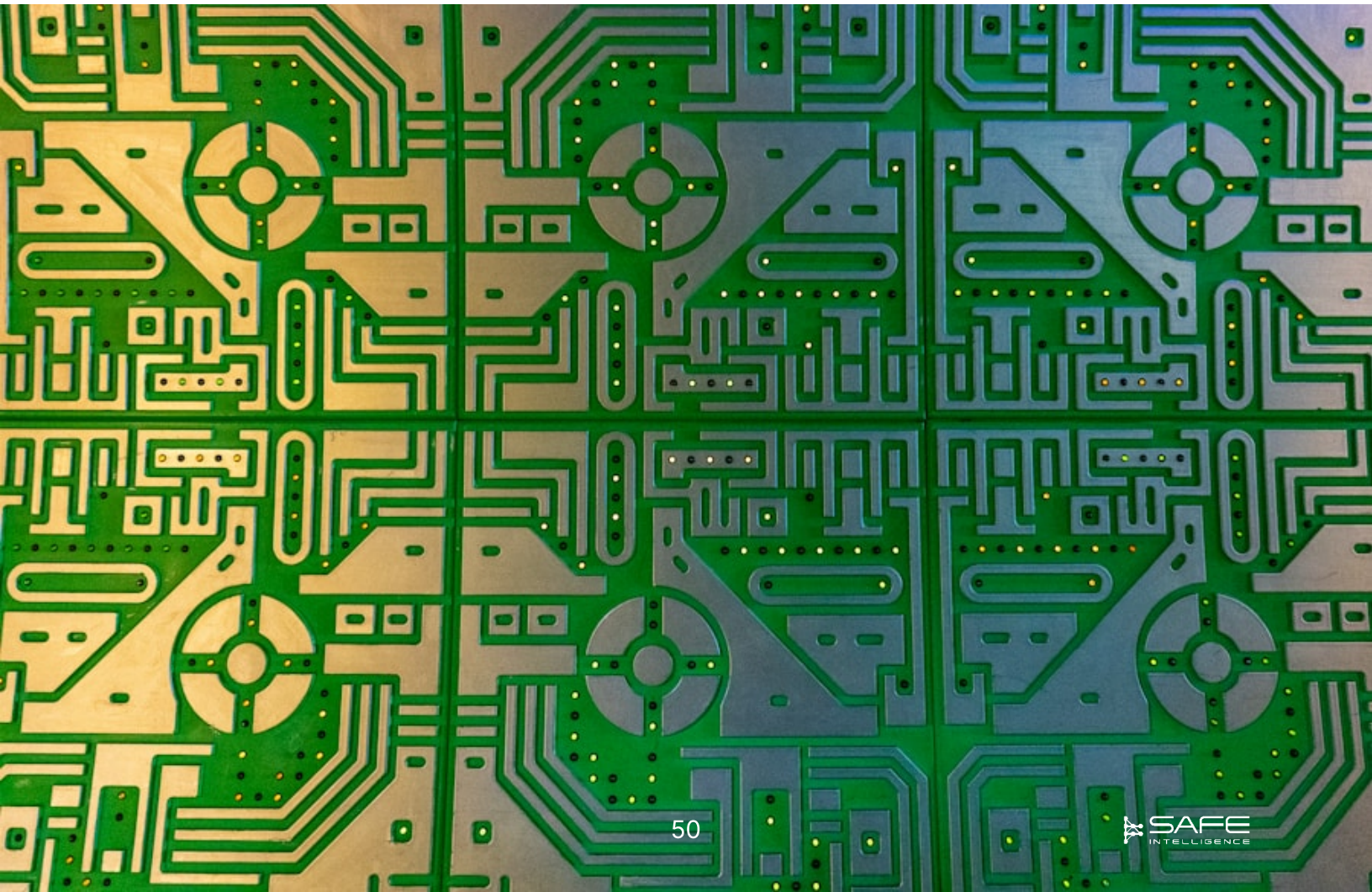
It's also important to monitor for anomalous or potentially harmful outputs – some companies employ real-time content moderation on AI outputs, or logging with periodic human review (for internal agents, sample their interactions to ensure they aren't doing something unintended). Feedback loops are valuable: capture user feedback (like thumbs-up/down ratings or error reports) and feed that into triaging potential problems. Monitoring should extend to security as well: watch for signs of prompt injections or misuse (e.g. a surge of weird input patterns could indicate an attempted attack). In sum, treat the AI agent as a living system that needs **24/7 oversight** similar to a server in production. And ensure there are resources (an on-call team or owner) assigned to respond if an issue is detected. Through continuous monitoring, you can detect deviations or incidents *before* they spiral into major failures, allowing for proactive fixes or withdrawals of the AI agent if needed.

- **REC: Instrument comprehensive audit logs (prompts, tool calls, data access, decisions).** Logging is needed for debugging, investigations, and demonstrating control effectiveness; some regimes also require log retention for certain systems. [Microsoft Observability]
- **REC: Implement continuous evaluation and drift monitoring in production.** Sample real traffic for quality/safety evaluation, run scheduled offline evals, and re-run adversarial probes as systems, policies, or models change. Consider including **shadow scoring** (LLM-as-judge or policy classifier) on sampled production traffic to flag safety regressions, ungrounded policy claims, and tool misuse.
- **REC: Continuously update guardrails based on real failures and edge cases.**Treat guardrails as living controls: when production reveals new bypasses or unsafe behaviors, add targeted tests and mitigations, then verify they work. [OpenAI Agent Guide]
- **REC: Run continuous canary probes in production**: a small, versioned suite of 'golden' and adversarial prompts (including tool-use scenarios) executed on a schedule to detect regressions even when no code changes ship. [Arize Guide]
- **REC: Protect secrets and sensitive data end-to-end (including telemetry).** Apply encryption, access controls, minimization, and careful retention policies; ensure logs don't become a high-value leakage channel. [OpenAI Best Practice]

Recommendations

Production Drift

Production drift shows up as confident misinformation: in April 2025, Cursor's AI support bot "Sam" invented a non-existent "one device per subscription" policy, triggering user cancellations before the company retracted it and started clearly labeling AI replies. Treat "policy claims" as high-risk outputs—require grounding to a source of truth, and monitor for new failure patterns after every change. [Arstechnica]



#11 Develop an AI Incident Response Plan

Despite all precautions, things can still go wrong. Be prepared with a clear **incident response plan (IRP)** tailored to AI-related incidents, so your organization can react swiftly and effectively when an AI agent causes or encounters a problem. Traditional IT/cyber incident response processes provide a starting point (with phases like identification, containment, eradication, recovery, and lessons learned [IAAP]), but AI introduces unique scenarios that the plan should cover. For example, what if your customer service chatbot outputs defamatory or biased statements? Or if an internal AI tool makes a critical calculation error that affects financial reports?

The AI IRP should outline **who needs to be involved** (technical teams, legal/compliance, PR, management) and how to escalate and coordinate. It should include communication plans – both internal (notify leadership, trigger war-room) and external (perhaps notifying customers, regulators, or issuing public statements, depending on severity). Regulatory obligations are crucial: certain AI incidents (like a data leak via the AI, or a safety incident in healthcare) may require *rapid notification to authorities* [ajg.com]. Ensure the plan lists the notification requirements for each jurisdiction you operate in, and have boilerplate templates ready.

The plan should also detail how to **preserve and analyze logs and model decisions** when an incident happens: – this is important for forensic analysis and for explaining to regulators what went wrong. For instance, maintain the ability to extract the conversation or transaction history that led to a bad outcome. Gallagher's cyber risk bulletin notes that many AI harms (bias, safety issues) fall outside traditional security response, hence organizations need dedicated playbooks for AI incidents. Typical incidents to prepare for include: model giving harmful advice, unauthorized data exposure, system misuse by an insider, significant drift causing compliance issues, or the need to pull an AI system from production due to external events (e.g. a new regulation or vulnerability discovered). By practicing this plan (through tabletop exercises or drills), you can ensure your team isn't scrambling in the heat of a crisis. Ultimately, an effective incident response not only mitigates damage but also provides learning to improve the AI and prevent recurrences – feeding back into your risk management cycle.

- **REC: Run AI-specific tabletop exercises and red-team drills.** Practice kill-switch decisions, evidence preservation, comms approvals, and post-incident guardrail/test updates; measure MTTR and recurrence. [IAPP]
- **REC: Have an incident response plan specific to AI-agent failure modes.** Include detection thresholds, rollback/kill-switch procedures, customer/user comms, evidence preservation, and post-incident guardrail updates. [GOV.UK]
- **REC: Plan for post-market monitoring and serious-incident reporting where regulated.** If you fall under regimes like the EU AI Act for high-risk systems, design monitoring plans and reporting workflows into operations (not as an afterthought). [AI Act Explorer]

Recommendations

Transparency in Failure

In March 2023, OpenAI took ChatGPT offline after a Redis-library bug exposed some users' chat titles; they published a clear post-mortem and restored service with mitigations and user notifications—a good template for transparency and rapid containment. [OpenAI Outage]

#12 Conduct Regular Audits and Improvement

Finally, just as agents, users, and use-cases evolve, so must risk management. In regulated environments, treat agents like models under model-risk discipline: **periodic re-validation** (often at least annually) plus immediate re-validation after any 'material change'—new model/vendor, new tools, new data sources, or significant drift signals. [Deloitte]

It is important to treat risk management for AI agents as an **ongoing process, not a one-time setup**. Schedule periodic **audits or reviews** of each AI agent – for example, every quarter or biannually – to reassess its performance, compliance, and risk profile. These audits can be internal or involve external experts for an unbiased perspective. They should evaluate whether the agent is meeting its expected quality metrics, whether new types of errors or complaints have arisen, and whether it's staying within ethical and legal guardrails. Use these reviews to ensure all prior recommendations are still in place: e.g. is monitoring functioning, are logs being checked, have drift thresholds been updated if the business context changed? Audits should also check documentation: is the model card updated with the latest info? Have there been any changes in training data or model version that need re-approval?

An important part of continuous improvement is **model maintenance** – retraining or fine-tuning the AI on new data to fix issues and prevent drift. Incorporate user feedback and incident learnings into the next model iteration. Many organizations implement an *AI model lifecycle management* similar to software versioning, where models are regularly updated and each update goes through testing and re-certification. The NIST framework underscores the need for *"ongoing monitoring and periodic review"* of AI risk controls, with clear roles for who does these reviews. It also suggests planning for eventual decommissioning – when an AI agent is retired or replaced, ensure a proper shutdown process to avoid orphaned systems causing unseen risks.

On the flip side, continuous improvement might involve **enhancing controls**: as new best practices or tools emerge (say, a better bias detection toolkit or a new security patch for LLMs), update your agent's safeguards accordingly. By institutionalizing a cycle of audit → feedback → update, you create a *"virtuous cycle"* where the AI agent and its governance get progressively stronger and more trustworthy over time. This helps sustain **long-term trust and reliability** in the AI solution amidst changing conditions.

- **REC: Adopt a continuous-improvement audit cycle (quarterly/biannual).** Re-run your eval suite, re-review tool permissions/data flows, and re-certify key controls after any model/prompt/tool change. Also define material change triggers (model/prompt/tool/data/vendor changes; policy updates; drift anomalies) that automatically require re-running the eval suite and updating the agent's risk assessment / documentation. [ISO 42001]
- **REC: Plan decommissioning as a first-class control.** When retiring an agent: revoke tokens, remove connectors, delete cached context where required, archive evidence, and verify no orphaned endpoints remain. [NCSC]
- **REC: Treat audit findings like software defects with tracked remediation.** Assign owners and deadlines, rank severity (security/privacy/safety/availability), require evidence of fixes (re-run evals, updated controls), and close the loop with regression tests and updated "Agent Cards"/documentation. [Dataguard]
- **REC: Treat user feedback and support tickets as test-case generators**: every credible failure report becomes a tagged regression test within 1–2 sprints. [Arthur.ai]

Recommendations

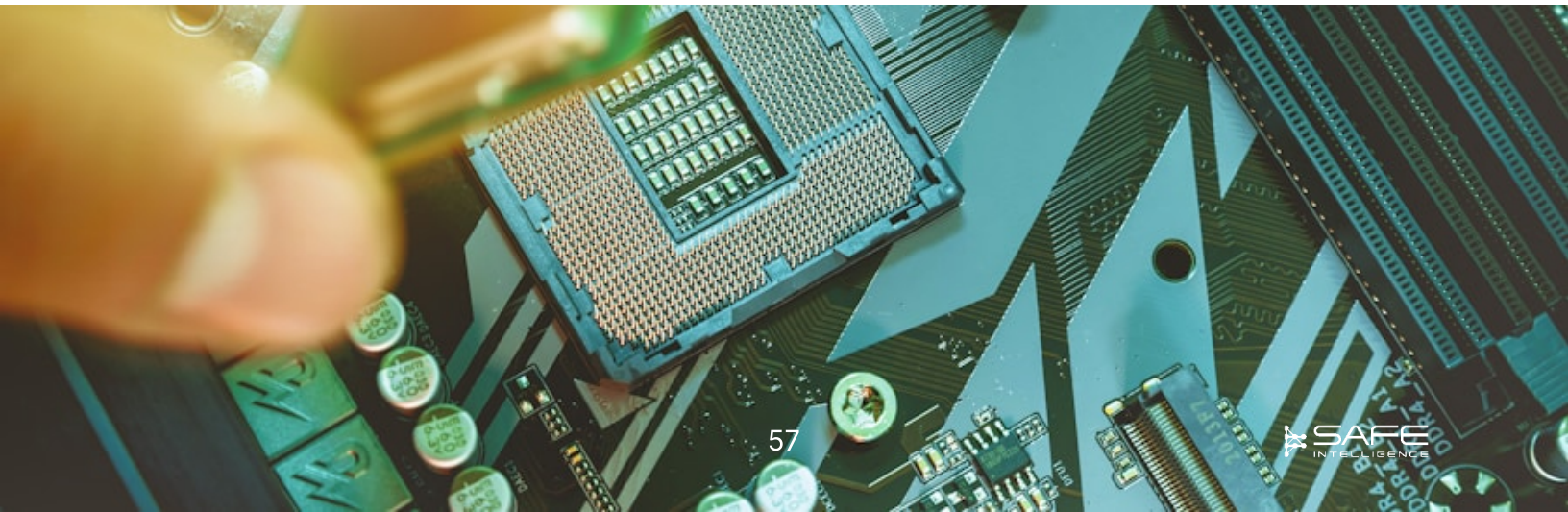
Recurring Audit Cycles

Make “agent safety” a recurring audit cycle: re-run evals, re-approve permissions, and re-review data flows every time models, prompts, or tools change. Google’s Secure AI Framework (SAIF) frames AI security as an end-to-end lifecycle—use that mindset to formalize continuous improvement. [SAIF]

Implementation: Operationalizing Oversight, Validation and Safety

Implementing the above recommendations will feel like a tall order, but hopefully some elements are already in place for existing IT systems, or AI usage is still nascent and the number of lighthouse cases not yet out of control.

Implementing recommendations like these requires integrating them into your organization's people, process, and technology structures. Here are key steps for structuring teams, assigning roles, and selecting tools to manage AI agents in practice.



Team structure and roles

Form a cross-functional **AI governance team or council** that meets regularly to review AI projects and incidents. This team should include the AI agent's business owner, data science/engineering leads, a risk/compliance officer, a security expert, and representatives from legal and possibly customer experience. Assign a *Model Owner* for each AI agent (usually a product or engineering lead responsible for its development and performance) and a *Risk Lead* (responsible for oversight of that agent's risk controls and compliance). Also enlist domain experts who can provide context (e.g. a medical expert for a healthcare chatbot). Microsoft recommends establishing an *Office of Responsible AI* or similar body to oversee AI ethics and governance across the company [Microsoft] – this office or committee can define standards, approve high-risk deployments, and coordinate audits.

It is important to understand that to any teams wanting to derive benefits from AI and Agents this group may appear to be adding serious friction to the task of getting productive with AI. In order to mitigate this friction effect (and also to prevent people by-passing the structure), it is important to automate as many of the processes as possible in a simple manner and limit oversight to essentials first.

Whereas the role of the governance team is to oversee processes and what gets deployed, there also needs to be a group (included in governance or separate) that deals with incidents in live systems. Ensure **clear escalation paths**: if an issue is detected by frontline staff or monitoring systems, who gets alerted and who has authority to pull the plug on the AI? Define these in advance. Embedding AI risk liaisons within existing risk or IT governance forums can also help integrate AI oversight into normal business processes (e.g. include AI checks in project gating or model review boards akin to traditional model risk management in finance).

Processes and Workflows

Integrate testing, validation, and review steps into your AI development lifecycle (often termed MLOps or AIOps). For example, require a **risk assessment and test sign-off** before an AI agent moves from development to production – this might include a checklist of items: bias test done, adversarial test done, privacy review done, etc.

Set up **change management** for AI models: any update (model retrain or prompt change) should go through a smaller regression test and approval. Leverage existing frameworks like NIST AI RMF or ISO/IEC AI standards by mapping their requirements to your internal controls. Implement **incident response drills** specific to AI: run simulations where, say, the AI outputs something harmful, and practice the response steps (this trains the team and validates the IR plan). Also, incorporate AI risk monitoring into enterprise risk dashboards so that senior execs get visibility.

It can help to maintain an **AI risk register** (as discussed) and update it quarterly, reviewing top risks and mitigation status in governance meetings.

For compliance, ensure documentation (like data lineage, model version history, test results, approvals) is stored in a repository – this will make audits or regulatory inquiries much smoother. Essentially, bake "AI QA" and "AI risk checks" into the standard operating procedures of both your IT deployment pipeline and your risk management regime.

Tools & Technology

The ecosystem of tools and solutions for building AI agents and AI applications more generally is ever expanding and so is the set of tools that can support the AI agent risk management process. For AI and agent building tools, we've provided a checklist in one of the Annexes to help assess vendor offerings for how easy they make it to hook in risk management processes.

To support the AI Agent risk management process, the tools are typically needed in the following broad areas:

- AI Agent registries and data collation.
- Testing and validation including red-teaming, robustness and functional testing.
- Guardrails, firewalls and filtering systems.
- Bias and fairness analysis.
- Escalation and alerting.
- Continuous monitoring and drift checks.
- Dashboards and reporting.

Luckily there is an expanding set of options in each of these areas from risk/compliance solutions such as Fairnow, Credo, and Yields to testing & validation solutions such as our own (which also covers bias/fairness and continuous monitoring), Braintrust, Langsmith, and others. In terms of guardrails there are solutions embedded into many AI agent infrastructure solutions as well as independent solutions such as NVIDIA NeMo, Protect.AI, and others.

Many cloud providers offer monitoring services for ML models – ensure they are configured with appropriate thresholds and integrate alerts into your incident management system (like ServiceNow or PagerDuty alerts to the on-call team). Access control tools are also key: ensure your AI agents (especially ones integrated with enterprise data) are behind proper authentication and cannot be invoked without authorization or throttling (to prevent abuse).

Culture, Collaboration, and Investment in the Team

Encourage a culture of **responsible AI innovation** – this means engineers and business teams work together with risk/compliance teams in a spirit of shared goal (safe, effective AI) rather than in opposition. Set the tone from the top that delivering AI fast is not enough; it must be done with due diligence.

Recognize and reward employees who identify risks or come up with improvements to AI safety. Internally, share post-mortems of AI incidents or near-misses to spread learning (blamelessly).

Consider participating in industry forums or consortia on AI risk (many organizations share best practices under frameworks like the NIST Trustworthy AI resource center [NIST] or via industry groups).

Keeping communication open – e.g. regular syncs between the AI dev team and the compliance team – can preempt conflicts and ensure everyone moves in step.

Conclusions

AI agents can unlock major operational gains, but only if they are treated as production-grade systems with a real risk surface. The core message of this handbook is simple: agent deployments are not “set and forget”. They require explicit governance, systematic testing, layered safeguards, strict control of data and permissions, and continuous monitoring so that failures are caught early and handled safely. The organizations that get this right will move faster over time because trust, auditability, and repeatability reduce friction rather than add it.

The task may seem daunting if none of this infrastructure is in place, but there are small, practical, steps forward. For example: pick one or two priority agents and (1) register them and their dependencies in a basic inventory, (2) run a first risk assessment and threat model focused on tool access, data exposure, and failure modes, (3) define a minimal test suite (red-team + functionality) and automate it as part of change management, and (4) turn on production logging, monitoring, and an escalation path with a clear “stop the line” owner. Once these foundations exist, extend them across more agents, tighten permission scopes, add more adversarial coverage, and operationalize regular audits and incident drills as standard practice.

Safe Intelligence as a Partner

We hope that you found this book useful in navigating the process ahead. We'd love to hear about your AI agent journey: please reach out at Safeintelligence.ai. Our tools and services cover many of the requirements touched on in this book. We're a trusted partner to help support your AI agent deployment journey.

Safe Intelligence:

- Is a spinout of one of the leading AI Safety Labs worldwide
- Provides state of the art tools that cover key tooling needs in steps 3 to 10 of the framework presented in this book.
- We're exclusively focused on AI model safety, validation and robustness, we don't build our own models or agents.

Annex A: Vendor Check List

This list provides helpful questions to ask for vendors used in agent development and deployment. Note that not all questions may be relevant to all vendors.

A) Product scope and intended use

- What is the vendor providing (foundation model, agent framework, tool/plugin ecosystem, RAG stack, monitoring, guardrails)?
- Does it support **read-only** use-cases, **action-taking** agents, or both?
- What are the hard boundaries on agent capability (tool allowlists, network egress controls, sandboxing)?

B) Benchmarking and evaluation (capability + safety)

- Are benchmarks available (capability, safety, robustness)?
- Can you benchmark on your own data with low effort?
- Can benchmarking run inside your environment (no data leaving your data infrastructure) if required?
- Do they support repeatable evals (fixed model versions, stable parameters, deterministic options where possible)?
- Can you evaluate tool-use (plans/actions) not just text outputs (e.g., success rate, unsafe action attempts, policy violations)?

C) Testing & staging support (safe experimentation)

- Is there a cost-free / low-cost testing arena (sandbox), and does it match production behavior?
- Is there a true staging endpoint with the same policies/filters/rate limits as prod?
- Can you run “dry-run” / simulation modes for tool calls (no real side effects)?
- Are there facilities for **replay testing** (re-run a recorded trace through a new model/version)?

D) Observability, logging, and auditability

- Can you export logs/traces in standard formats (e.g., **OpenTelemetry**) end-to-end (prompt, context, tool calls, outputs, policy decisions)?
- Do logs include trace IDs linking: user request → retrieval → model call(s) → tool call(s) → final output?
- Can sensitive data be redacted/minimized in telemetry (PII, secrets, proprietary context)?
- Are there immutable **audit logs** for admin actions/config changes (RBAC changes, key rotation, policy edits)?

E) Change management and versioning

- How far in advance are deprecations and model/product changes announced?
- Can you pin to specific versions (model, tool schema, agent runtime) and keep them available for extended periods?
- Is rollback supported (or do you have to self-manage fallback)?
- Are breaking changes communicated with migration guides and known-behavior deltas?

F) Data handling and learning policies

- What are policies on using customer data for service improvement/model training? (opt-in vs opt-out, defaults, contractual guarantees)
- Data retention: how long are prompts/outputs stored (including caches), and can you set retention to zero?
- Data residency: can data stay in-region, and are sub-processors disclosed?
- Encryption: in transit/at rest; do they support customer-managed keys (BYOK/CMK)?
- Can you perform verified deletion (including backups where feasible) and support DSR workflows (GDPR/CCPA)?

G) Security controls and enterprise access management

- Identity: SSO (SAML/OIDC), SCIM provisioning, MFA, tenant isolation.
- Authorization: fine-grained RBAC/ABAC for projects, models, tools, logs, and admin operations.
- Network security: private endpoints/VPC options, IP allowlists, egress controls, rate limits/quotas, abuse detection.
- Secrets handling: secure storage/rotation for tool credentials; no secrets exposed to model context by default.

H) Tooling / agent "permissioning" and action safety

- Does the platform support least-privilege tool access (per-agent, per-tool, per-action scopes)?
- Are there built-in mechanisms for **approval gates** (human-in-the-loop) for irreversible actions (send/transfer/delete)?
- Are tool calls schema-validated and allowlisted (block "raw output → API call" paths without validation)?
- If the vendor provides connectors/plugins: can you restrict which connectors an agent can see/use?

I) Guardrails, policy enforcement, and prompt-injection resilience

- Can you enforce policies at multiple layers (input filtering, output filtering, tool-call filtering, retrieval filtering)?
- Can you tag/segregate untrusted content (e.g., RAG documents, emails, web pages) so it cannot become "instructions"?
- Do they provide prompt-injection mitigations or detection signals (and documentation of known limitations)?
- Can you run custom safety policies and evaluations (domain-specific restrictions) without vendor lock-in?

J) Vendor assurance, compliance, and documentation

- What risk mitigations are in place and documented (threat models, safety testing, secure SDLC)?
- What regulations are relevant and how do they support compliance (GDPR, EU AI Act, sector rules)?
- Security/compliance posture: SOC 2 Type II, ISO 27001 (and related), penetration testing, vulnerability disclosure.
- Do they provide model/system documentation (model cards, limitations, intended use, safety features)?

K) Incident response and support

- SLAs for uptime and support; 24/7 escalation paths for critical incidents.
- Security incident notification timelines; postmortems and corrective-action transparency.
- Dedicated channels for vulnerability reporting; bug bounty / coordinated disclosure policy.

L) Portability, lock-in, and operational resilience

- Can you export prompts/configs/policies/logs/eval results in standard formats?
- Can you swap models/vendors without rebuilding your risk controls from scratch?
- Multi-region availability, DR posture, and explicit behavior for failover and degraded modes.

Thank You

How to get in touch

Email: info@safeintelligence.ai

safeintelligence.ai

Address

One Lyric Square

Hammersmith

London

W6 0NB

UK