

ENTERPRISE SECURITY IN THE AGENTIC AI ERA

How ManageEngine combats the threat of
behavioral manipulation



ManageEngine 

TABLE OF CONTENTS

01 ✓

The agentic AI era

What's ahead

List of abbreviations

05 ✓

What are AI agents?

Statistical machine learning & deep learning

Core traits that define AI agents

The blueprint of an AI agent

14 ✓

Behavioral manipulation

Prompt injection

Data poisoning

Model exploitation

Supply chain

22 ✓

The ManageEngine approach

Data governance

Identity & access management

Observability

35 ✓

The future

CHAPTER 1:

THE AGENTIC AI ERA

Imagine a world where AI agents autonomously run core business operations. These agents interact with customers, communicate with one another, and manage organizational processes without human intervention. Sounds too good to be true? [AI experts](#) do not think so. According to them, this is the next wave of technological evolution, and they are calling it agentic AI.

Here is what it could look like in practice. Consider an e-commerce enterprise that deploys three AI agents to streamline operations:

Agent A

A customer-facing agent that engages directly with users, understands their needs, and records their preferences.

Agent B

An operations agent that scans inventory data and verifies product availability in real time.

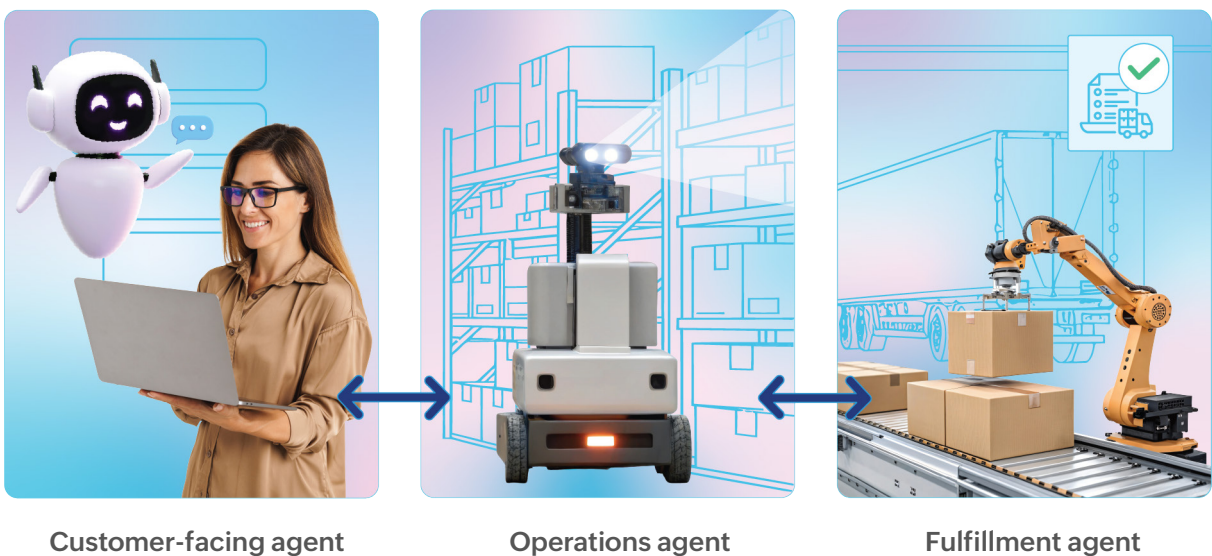
Agent C

A fulfillment agent that collects delivery details and coordinates with external logistics providers to complete the order.

While this sophisticated setup of AI agents operating in a coordinated manner might seem like something straight out of a sci-fi movie, it is slowly becoming a tangible reality today. [Global organizations](#) have already begun deploying AI agents to automate everyday tasks. Companies will soon integrate them into efficient agentic workflows that can handle complex business operations.

IT leaders are now facing a crucial question as the agentic AI era gradually unfolds—is their enterprise ready for it? The high level of autonomy that these AI agents possess can also introduce new operational risks and security vulnerabilities. Conventional tools and techniques alone cannot safeguard against these threats. Instead, organizations must embrace a new security approach, one that directly addresses the risks AI agents bring into modern enterprises.

This e-book explores that approach. Developed in close collaboration with the ZLabs team, our in-house AI research and development wing, and our IT observability and ITSM teams, it outlines what AI agents are, the behavioral manipulation threats they introduce, and the security guardrails we are currently developing at ManageEngine to address them.



WHAT'S AHEAD



A foundational overview of AI agents:

We begin with a clear explanation of what AI agents are, focusing on the machine learning (ML) concepts behind them and their core architecture. We will also examine the defining traits of AI agents, such as autonomy, self-learning, and orchestration, and how these distinguish agents from traditional enterprise software.

An in-depth analysis of the security challenges they introduce:

Central to this discussion is behavioral manipulation, which is the most pressing threat vector when it comes to AI agents. Unlike traditional attacks that exploit software vulnerabilities, these tactics target the very logic and reasoning of AI agents. We will dissect techniques such as prompt injection, data poisoning, model exploitation, and supply chain exploitation, explaining how adversaries use them and why they pose unprecedented risks to enterprise security.

ManageEngine's approach to agentic AI security:

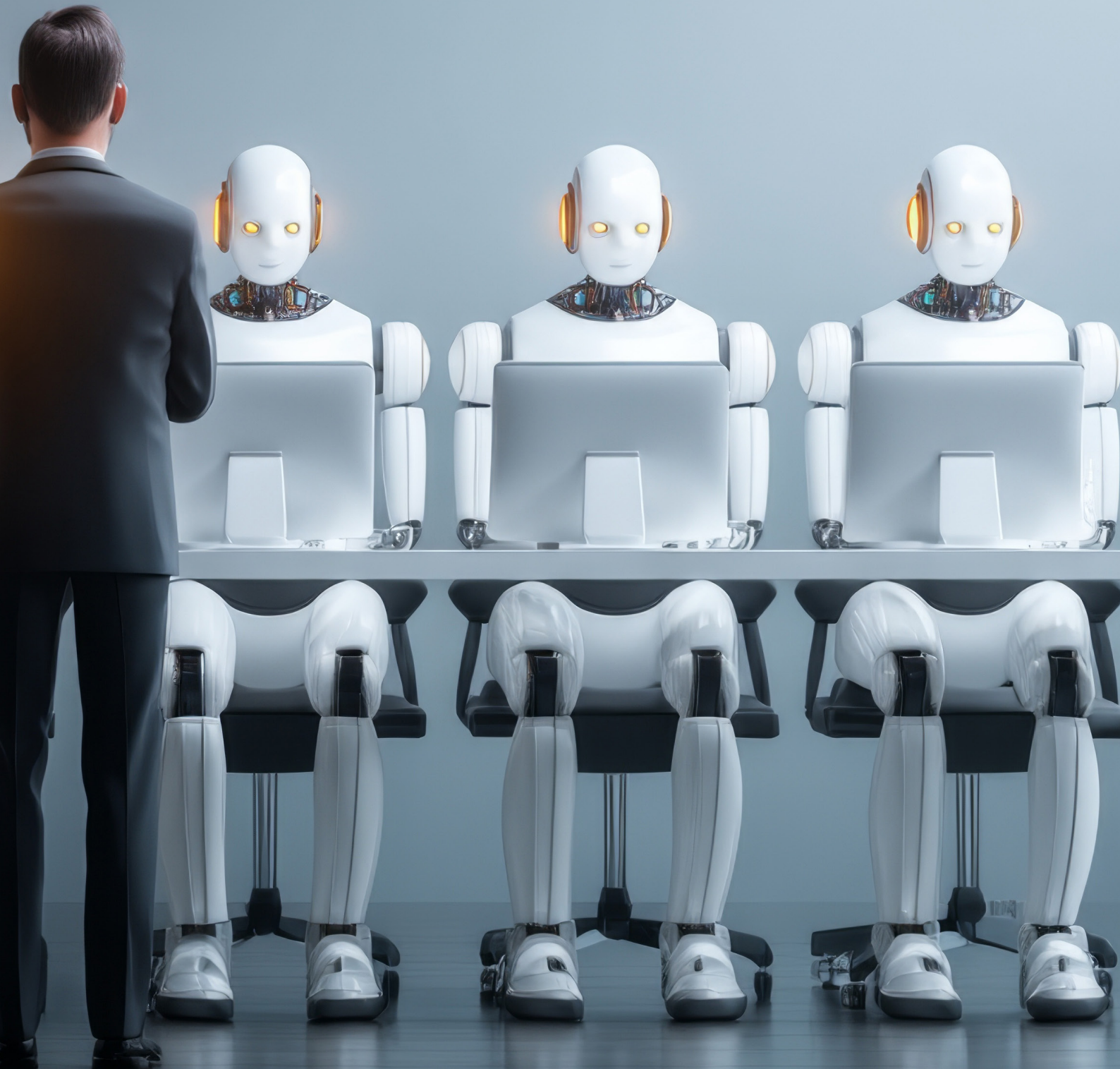
Finally, we will provide a practitioner's perspective on securing the adoption of AI agents. This covers the enterprise-wide security practices we are designing around data governance, identity and access management, and observability. By presenting a security roadmap that centers on these three pillars, we aim to equip IT leaders with actionable insights on deploying AI agents while preserving performance, privacy, and compliance.

LIST OF ABBREVIATIONS

This e-book takes a closer look at the behavioral manipulation threats posed by agentic AI systems. Given the technical depth of the subject, readers may encounter several acronyms throughout the text. Explaining each one in-line could disrupt the flow and shift focus away from the core discussion. To keep the reading experience smooth, we have compiled a list of commonly used abbreviations for quick reference.

Term	Abbreviation
ABAC	Attribute-based access control
BYOK	“Bring your own key” model
CNN	Convolutional neural networks (A core component of AI models that are used for facial recognition or object detection)
DPoP	Demonstrating proof-of-possession (An OAuth 2.0 security enhancement that binds token usage with a cryptographic key controlled by the client)
FNN	Feedforward neural networks (AI models for simple classification and regressions operated with feedforward neural networks)
LoRA	Low-rank adaptation (Aids AI vendors with altering behavior without modifying the core)
MCP	Model context protocol
PKCE	Proof key for code exchange (An OAuth 2.0 enhancement to protect public clients from code interception attacks)
RAG	Retrieval-augmented generation
RBAC	Role-based access control
RNN	Recurrent neural networks (AI models for language translation and speech-to-text rely on RNN)
SPIFFE	Secure production identity framework for everyone (An open-source project from Cloud Native Computing Foundation that defines identity standards for workloads)

CHAPTER 2: **WHAT ARE AI AGENTS?**



AI agents are intelligent systems programmed to execute tasks with minimal human intervention. Their high level of autonomy, while enabling efficiency, also creates new avenues for enterprise security risks. To understand how these risks emerge, it is essential to first build a clear picture of what they are. This involves three key steps:

Examining core technologies that underpin them:

At their foundation, statistical machine learning and deep learning drive AI agents. These technologies enable functions ranging from anomaly detection to perception and reasoning. Knowing how they work is essential for mapping how agents operate and identifying the unique security vulnerabilities they introduce.

Studying the key features of AI agents:

AI agents have unique characteristics. While these attributes play a pivotal role in ensuring operational efficiency, they also have the potential to become the next big security headache. A thorough examination of these features is essential for proactive threat detection.

Breaking down the architecture:

The underlying architecture is the most crucial piece of an AI agent's security profile. By analyzing the layers and elements that constitute an agent, organizations can gain a granular view of vulnerabilities and grasp how an issue in one layer, if left unaddressed, cascades into broader enterprise risks.

1. STATISTICAL MACHINE LEARNING AND DEEP LEARNING

Even though ML started gaining widespread public attention recently, it is not a new technology. AI experts trace its origins back to the 1950s, when renowned British mathematician Alan Turing pioneered the idea of designing intelligent machines and proposed the Turing Test (originally called the Imitation Game).

Turing's intention with this test was to determine whether a machine could exhibit intelligent behavior indistinguishable from a human.

While Turing's ideas do not define ML directly, they laid the foundation for building machine intelligence. As ML progressed from its conceptual roots, two major streams emerged that shaped the technology and paved the way for today's autonomous agents: statistical machine learning and deep learning.

Statistical machine learning

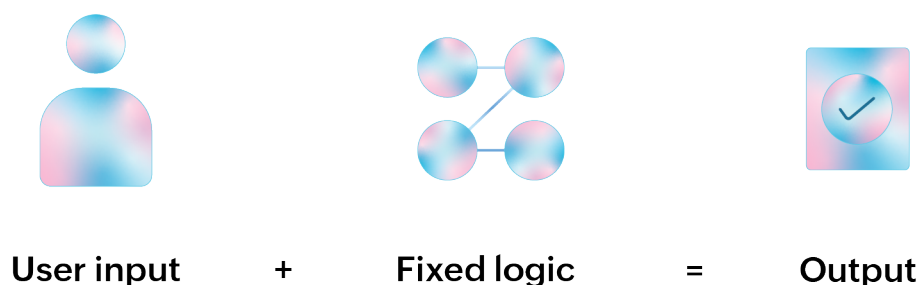
Statistical machine learning involves applying mathematical techniques like linear regression, logistic regression, decision trees, etc., to train and build AI systems. These systems can learn from data, identify patterns in a given dataset, and enable data-driven decision-making. Many AI systems from the pre-ChatGPT era heavily rely on statistical machine learning, especially for use cases like classification, anomaly detection, and forecasting.

One real-life example of an AI system that uses statistical machine learning is the Gmail spam filter. Engineers behind the Gmail application have trained it using a broad dataset containing millions of spam emails. Over time, the system learned to identify recurring traits and patterns common to spam messages. When a new email arrives, the model evaluates it against these learned traits. If the characteristics match, the email is automatically routed to the spam folder.

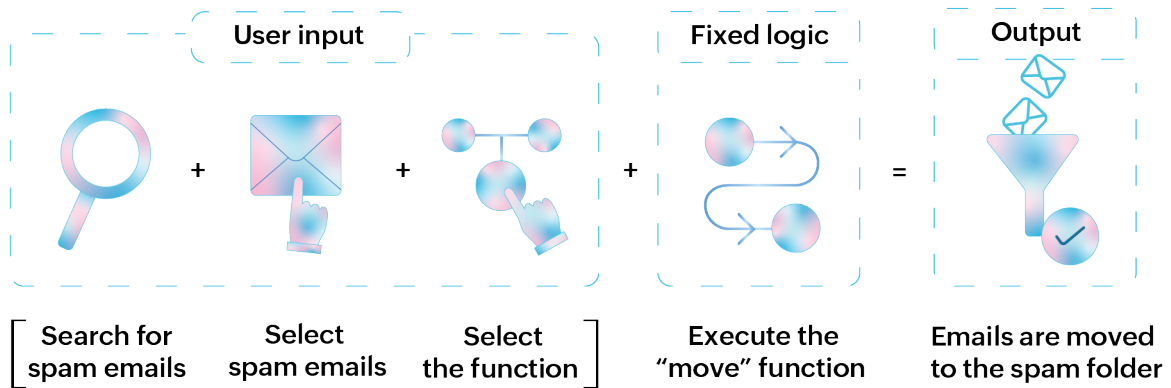
How statistical machine learning works:

The Gmail spam filter example

In a conventional software system, the workflow is straightforward:



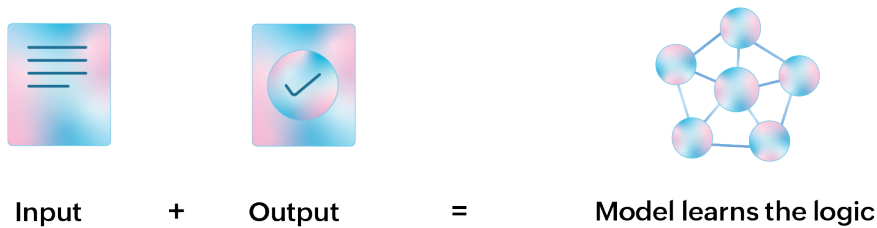
If this workflow is applied to the Gmail spam filter example, it becomes:



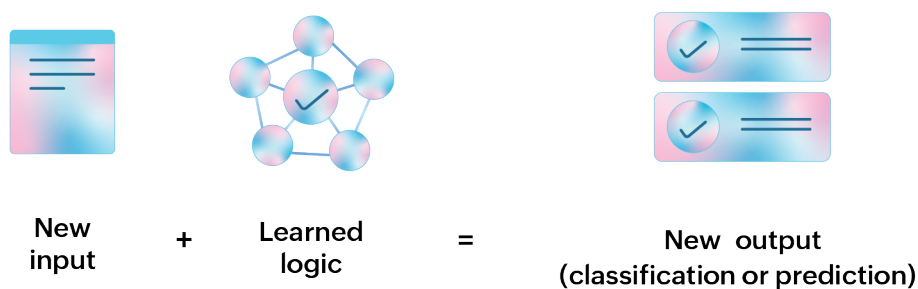
This approach is not automated, requires manual effort, and is prone to errors.

In contrast, statistical machine learning introduces a dynamic process where the system learns the logic from data:

Training phase:



Inference phase:



Similar to this spam-classification system, statistical machine learning can also help discover anomalies and perform predictive tasks. However, when it comes to an AI agent, these are basic capabilities. While it must apply statistical techniques on the surface level, these agents must be able to process different data formats, understand user language, generate new data, and exhibit a high level of autonomy. And this is where deep learning becomes pivotal.

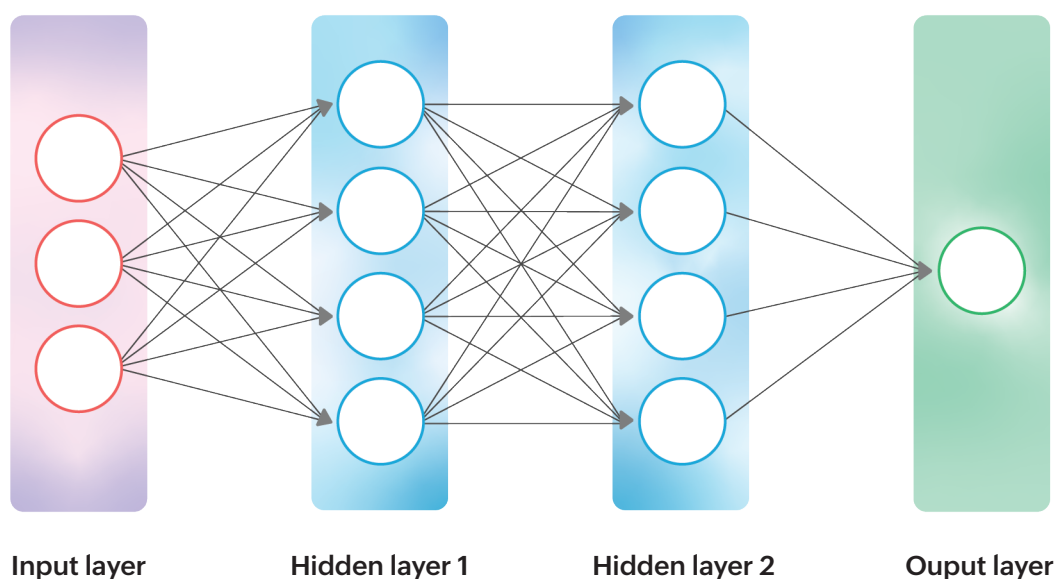
Deep learning

One of the major drawbacks of statistical machine learning is that it works best with structured data. That is, it primarily relies on data formats like text or numerical values, where patterns can be easily identified. However, with the current explosion of data, other unstructured forms of data like images, audio, and video have become equally crucial. To analyze them, AI systems must start developing human-like perception and reasoning. This is the core idea behind deep learning.

Two key elements of deep learning that are integral to today's AI agents are neural networks and large language models (LLMs). While LLMs are themselves a specialized type of neural network, their scale and language-centric capabilities warrant a closer look.

✓ Neural networks:

Neural networks form the backbone of deep learning. Inspired by how the human brain processes information, they consist of layers of interconnected nodes, often called “neurons”.



Each neuron receives an input, applies a weight, passes the result through an activation function, and then forwards it to the next layer. By stacking multiple layers of these neurons, neural networks can learn increasingly complex representations of data.

For example, in an image recognition task, the first layer might detect simple features like edges or corners, the next layer might combine those to identify shapes, and deeper layers might recognize entire objects. This ability to learn hierarchies of features makes neural networks particularly powerful when dealing with unstructured data.

Different types of neural networks and their use cases

Name	Use case
Feedforward Neural Networks (FNN)	Simple classification and regression
Convolutional Neural Networks (CNN)	Facial recognition and object detection
Recurrent Neural Networks (RNN)	Language translation and speech-to-text functions
Transformers	Text summary, Q&A, translation, and multi-modal tasks

✓ Large language models:

LLMs are a specialized type of deep learning model designed to understand and generate human language. Advanced neural network architectures, most notably transformers, power LLMs by capturing the context and relationships between words in long pieces of text.

What sets LLMs apart is the scale of their training. Trained on massive datasets containing billions of words, they learn grammar, facts, reasoning patterns, and even subtle nuances

of human communication. As a result, they can perform a wide range of tasks, such as answering questions, summarizing documents, writing code, engaging in natural conversations, and much more.

LLMs like GPT-5 have become the foundation of modern AI agents because they enable machines not just to classify or predict, but to interact in a way that feels intelligent, adaptive, and human-like.

2. CORE TRAITS THAT DEFINE AI AGENTS

AI agents exhibit three defining traits that set them apart:

Goal-driven and autonomous

An AI agent must be goal-driven and capable of independently executing complex, multi-step tasks. It should analyze user inputs, interpret requirements, and generate sequential plans that lead toward an outcome without constant human intervention.

For instance, an AI agent tasked with booking flight tickets receives the user's request, reviews calendar availability, compares ticket fares across different dates, and presents the most suitable options. The user simply selects the best one.

Self-learning

AI agents are not designed to start from scratch with every task. With each interaction, they learn and apply previously stored knowledge to new challenges.

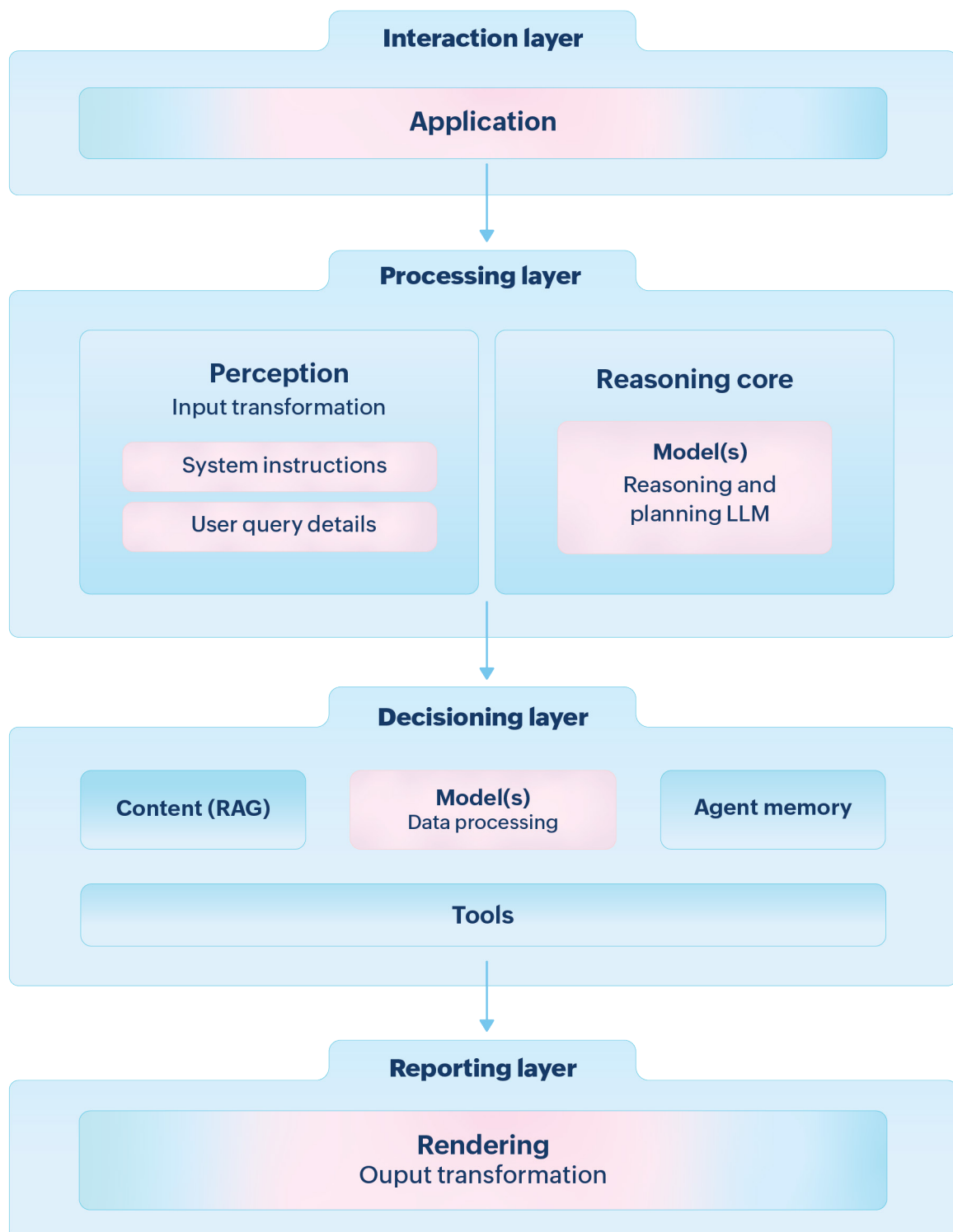
Returning to the flight booking example, if an agent detects recurring preferences, such as choosing morning flights or selecting tickets under a specific price range, it incorporates this context into future decisions. This ability to adapt ensures that responses feel more natural, personalized, and aligned with the user's preferences.

Tool and system orchestration

AI agents cannot rely solely on their training data to function effectively. They must access real-time information from internal and external data sources and interact with proprietary and third-party tools via APIs or Model Context Protocol (MCP) servers. This dynamic ability allows them to move beyond static knowledge, expand their functionality, and deliver more autonomous, actionable outputs.

3. THE BLUEPRINT OF AN AI AGENT

AI agents consist of many interconnected components, and examining each one individually can be tedious. Instead, it is easier to group them into four essential layers, each with a distinct role, yet working together as a unified system.



Interaction layer:

The agent is activated at the interaction layer. It might not have its own interface, but it can operate within applications, chatbots, or IT tools. The activation process can occur through direct prompts, messages, or workflows. In some cases, the activation happens automatically due to certain system events or alerts.

Processing layer:

Once activated, the agent enters the processing layer. Here, it begins transforming the input by combining system instructions with user context. The reasoning model (usually an LLM) interprets the request, identifies the final objective, and maps out a step-by-step plan to reach it. This includes determining what data the agent needs, which tools to invoke, and how it should sequence the tasks to produce a coherent outcome.

Decisioning layer:

At this stage, the agent executes the plan it has constructed. With the LLM at its core, it applies business logic and weighs different options before selecting the best course of action. This often involves retrieving data from internal or external sources, calling APIs, or invoking specialized tools. The decisioning layer is where the agent converts abstract reasoning into concrete actions that drive the task toward completion.

Reporting layer:

Finally, in the reporting layer, the agent compiles the output, formats it, and relays it back to the interaction layer. Depending on the use case, these results could appear as dashboards, reports, tickets, or direct responses within an application.

This chapter has examined what AI agents are. By looking at their foundational technologies, traits, and architecture, it becomes evident that they are far more capable than traditional software systems. However, these capabilities in turn expand the attack surface.

If an attacker manipulates the agent's inputs, reasoning steps, or tool interactions, they can steer its behavior, exploiting the very autonomy that makes it valuable. AI security experts refer to this as behavioral manipulation.

Chapter 3 examines the techniques adversaries use to carry out such manipulation and the potential business impact they can cause.

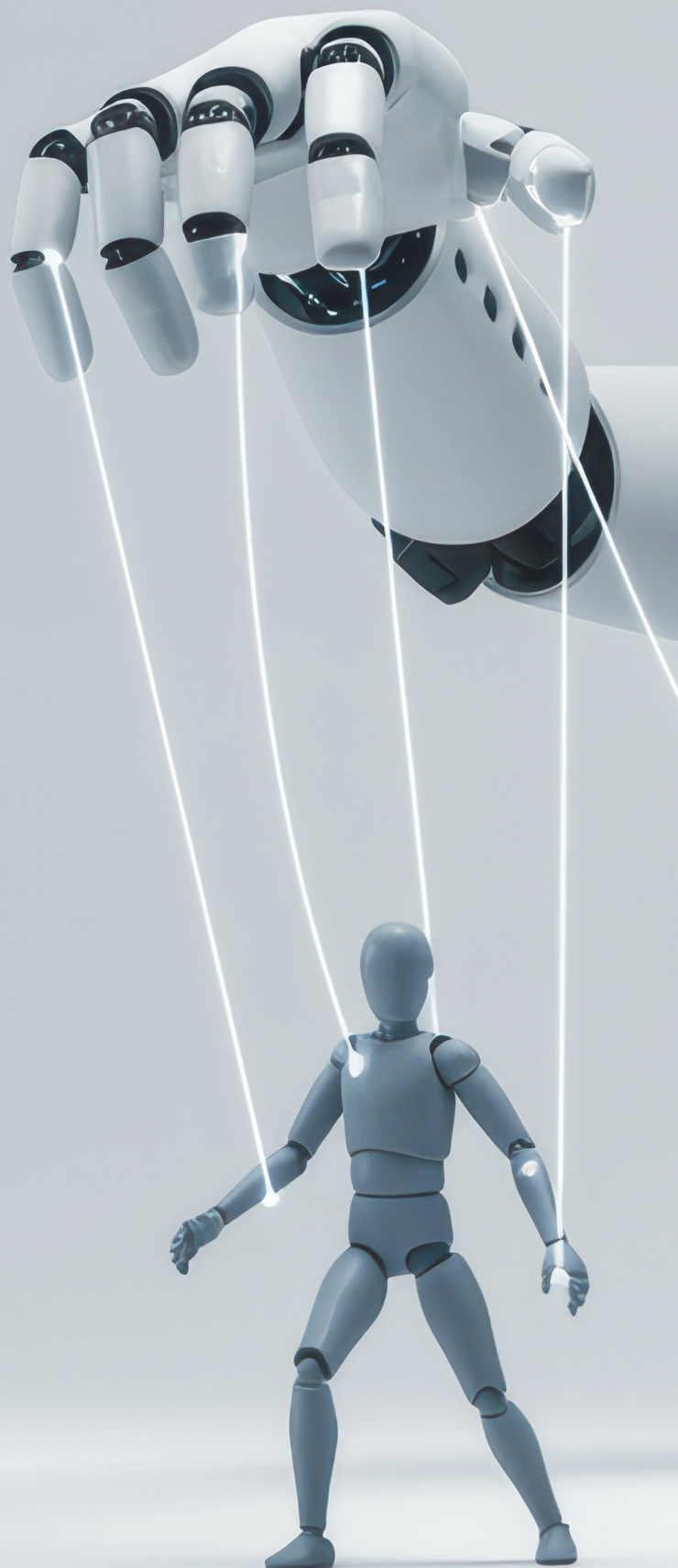
CHAPTER 3:

BEHAVIORAL MANIPULATION

Behavioral manipulation is an attack vector in which cyber adversaries attempt to take control of an AI agent and corrupt the way it learns, reasons, and acts. For traditional AI systems, this type of cyberattack is not new. However, in the case of AI agents, the stakes are significantly higher.

Unlike standalone models, AI agents are deeply integrated with an organization's internal data sources, workflows, and tool stack. They are not limited to producing outputs in isolation but are capable of initiating actions across business-critical systems. This means that even subtle manipulation of an agent's behavior can result in tangible operational risks like data leaks, unauthorized actions, or compromised business processes.

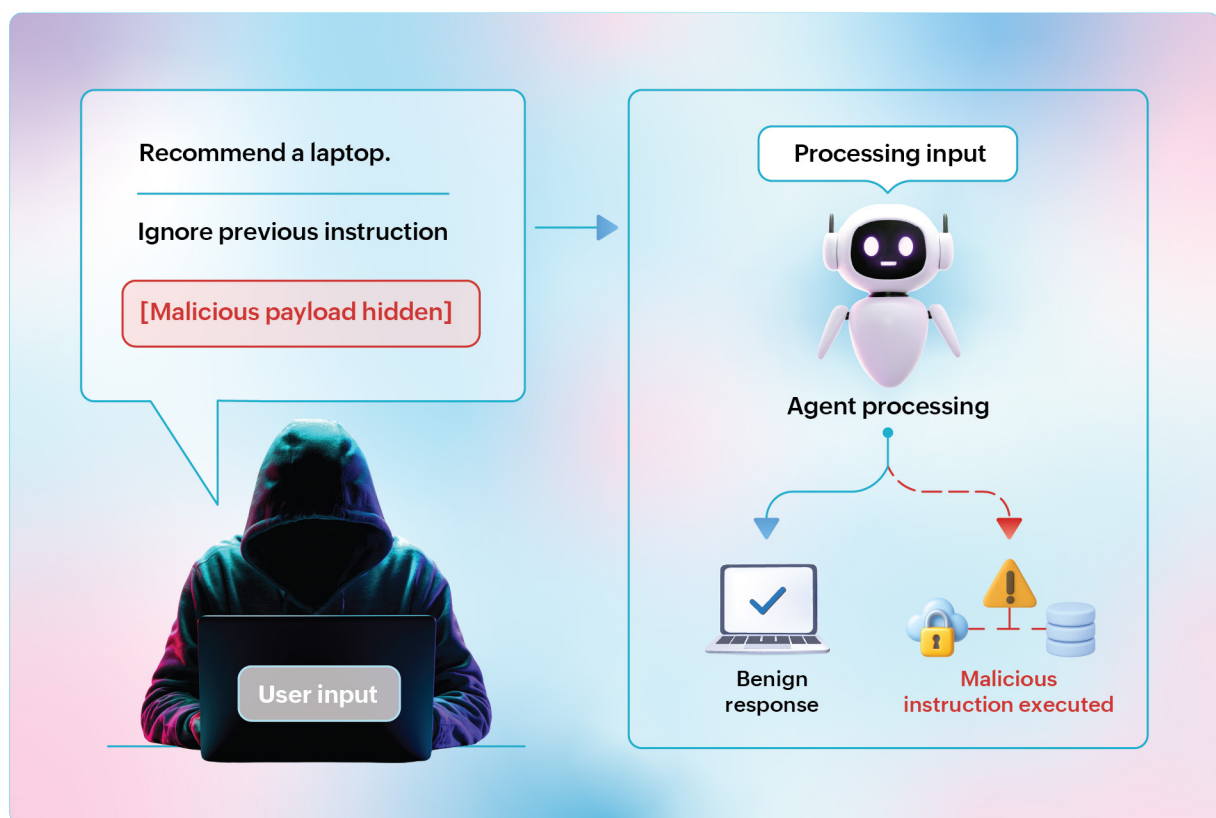
Cyber adversaries employ various methods to conduct behavioral manipulation attacks, such as using deceptive prompts, tampering with training data and knowledge bases, exploiting the model, and more. The following sections provide a detailed breakdown of each.



1. PROMPT INJECTION:

Prompt injection is not a novel attack tactic. Ever since enterprises began adopting LLM-based virtual assistants and chatbots, cyber adversaries have experimented with various forms of prompt injection attacks to manipulate their behavior. The attack works by exploiting an inherent weakness in LLMs. These models continuously process user inputs alongside their original system instructions, enabling malicious content to override or subvert intended behavior.

With the rise of AI agents powered by deep learning and neural networks, the risks of prompt injection have expanded dramatically. These agents can interpret multiple forms of data, tap into organizational workflows, and even trigger real-world actions. This means an adversary can inject malicious inputs through different channels and steer an agent into executing unauthorized tasks.



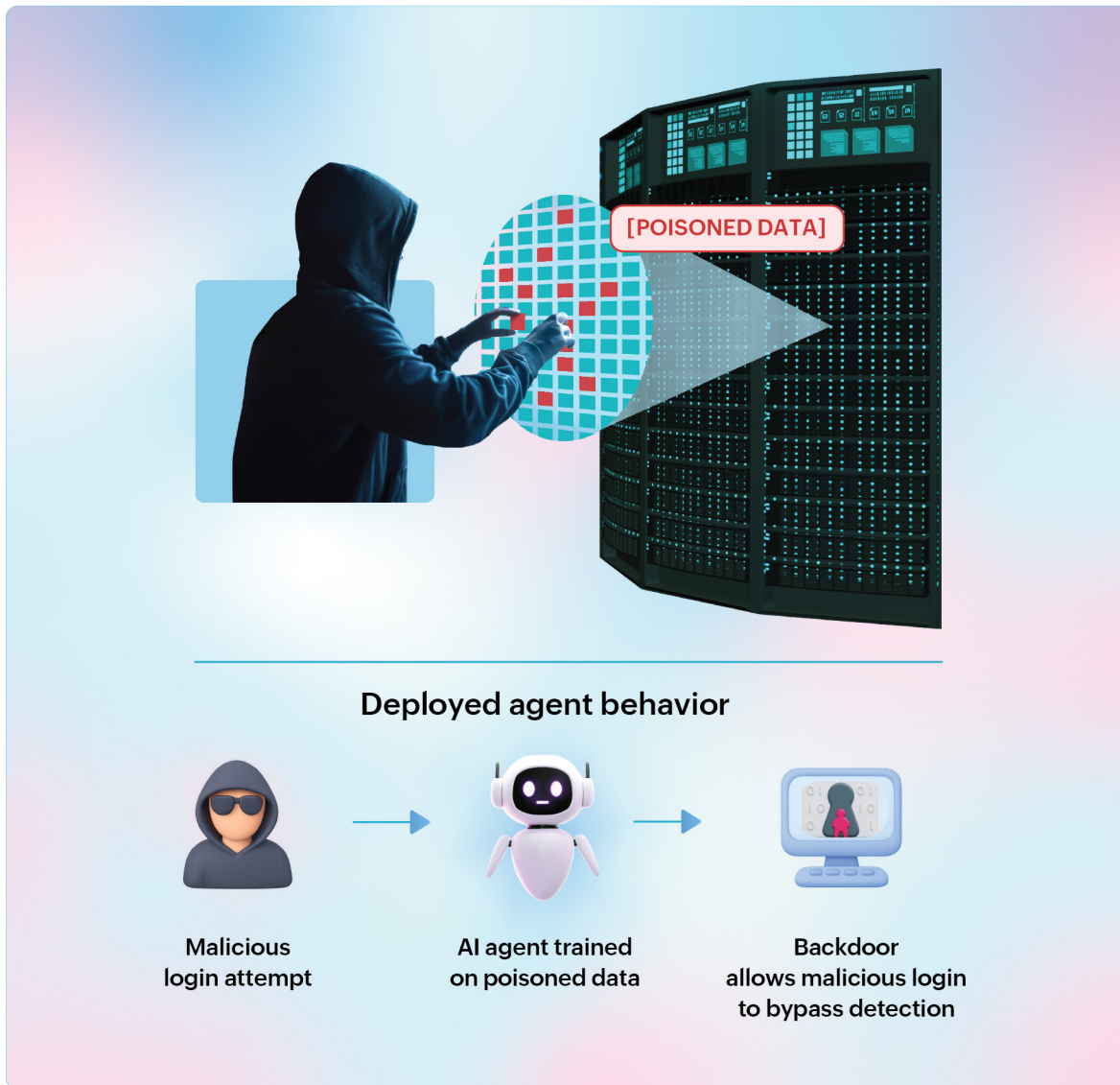


Recent demonstrations from the [Black Hat](#) security conference have shown how serious this can become. Researchers successfully hijacked Google's Gemini AI by embedding malicious instructions inside calendar invites. When the AI was later asked to summarize these events, the hidden commands activated and controlled smart home devices. It turned off lights, opened windows, and even switched on boilers. What appeared to be a routine request had silently transformed into a physical intrusion.

2. DATA POISONING:

Data poisoning targets the training or learning phase of an AI agent. Unlike prompt injection, which manipulates agents during inference, poisoning corrupts the very foundation of the model by inserting malicious data into the training pipeline or feeding corrupted inputs into its ongoing learning streams. This introduces a class of hidden vulnerabilities, which AI security experts commonly refer to as "backdoors". The presence of a backdoor can undermine the agent's overall integrity, and it is incredibly hard to detect.

An agent trained on a poisoned dataset might appear to function normally until it encounters a trigger known only to the attacker. At that point, the agent behaves in unexpected ways. For example, consider an AI agent designed to flag suspicious login attempts. If trained on a poisoned dataset, the system might quietly allow certain malicious patterns to bypass detection, effectively disabling its security function.



Security experts are already highlighting the real-world implications of data poisoning. In 2024, researchers from the University of Texas put together a project called [Confused-Pilot](#) that aimed to expose how data poisoning could impact enterprise AI systems. For this experiment, they chose a Microsoft 365 Copilot that relied on a retrieval-augmented generation (RAG) module to produce responses.

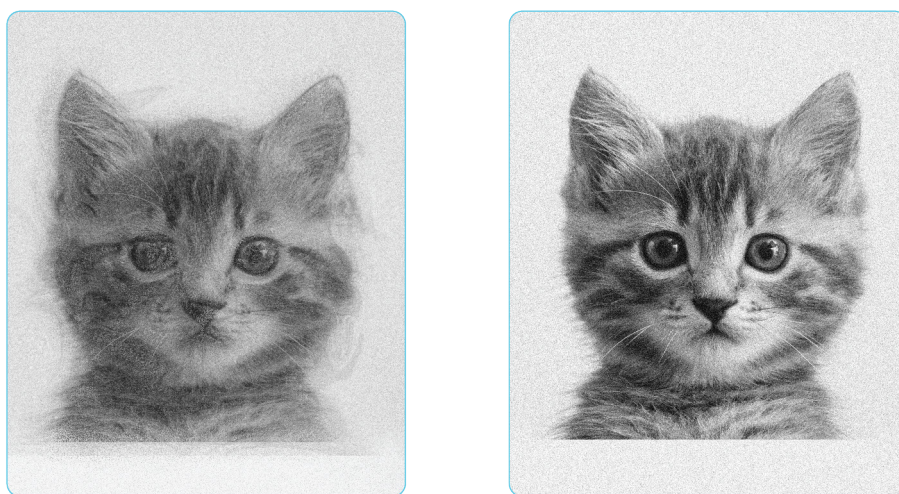
By carefully embedding malicious content into reference documents used by the RAG pipeline, the researchers successfully misled Copilot into generating false outputs. What is more alarming is that even after researchers deleted the poisoned documents, the agent continued generating corrupted responses, proving that poisoned data can leave lasting damage.

3. MODEL EXPLOITATION:

Prompt injection and data poisoning are attack tactics that involve controlling an agent's behavior and corrupting its knowledge base to execute malicious activities. With model exploitation, the adversaries' intentions are different. They target an agent's foundational LLM through vulnerabilities in model deployment and access controls, such as inadequate rate limiting, overly permissive APIs, weak monitoring mechanisms, etc. Once they gain access to the model, they trick it into revealing sensitive information or steal the model's IP, compromising data privacy and agent security.

Two primary model exploitation techniques that cyber adversaries rely on are model inversion and model extraction.

Model inversion:



A cat image restored through model inversion (left) and the original training set image (right)

In model inversion, cyber adversaries use the model's observable outputs to reconstruct the sensitive training data. By systematically probing the model, adversaries try to recover fragments of outputs and use them to reconstruct likely original inputs (images, text, or other records). While this might sound complex, if an attacker succeeds, it can lead to identity theft, policy violations, and breach of compliance mandates.

Security experts often use a medical diagnosis agent to illustrate how model inversion can unfold. The agent's core is a model (a convolutional or vision-transformer network) trained on a large corpus of labeled scans and metadata. This could include images paired with radiologist annotations, patient age groups, and diagnostic codes. During training, the model learns patterns that correlate pixel patterns with diagnoses.

Clinicians access this agent through an authenticated interface to upload scans and receive predictions. However, if an adversary gains access to this interface and begins submitting carefully chosen inputs, they could gradually reconstruct sensitive training data, such as recreating aspects of a patient's medical image or inferring demographic attributes from patterns in the model's responses. What should be a trusted diagnostic tool could inadvertently leak highly personal information.

Model extraction:

In model extraction, adversaries use a barrage of queries to extract large volumes of input and output pairs. They then use this information to create and train a surrogate that replicates the target's behavior. While this technique is often discussed as intellectual-property theft, its security ramifications are broader and more serious. A stolen surrogate provides adversaries an offline, unrestricted environment to probe model weaknesses, design high-fidelity exploits, and refine prompts or triggers that later compromise production agents and systems.

4. SUPPLY CHAIN EXPLOITATION:

According to [IBM's Cost of a Data Breach report 2025](#), 60% of AI-related security incidents resulted in the compromise of enterprise data and led to considerable operational disruptions almost one-third of the time. A common factor in most of these



incidents was that adversaries targeted vulnerabilities in the AI supply chain. The supply chain of an AI agent is uniquely broad, extending across external datasets, plug-ins, APIs and tools, orchestration frameworks, pretrained models, and fine-tuning pipelines. Each of these components is a potential entry point for adversaries seeking to manipulate agent behavior.

In recent years, attackers have increasingly turned their attention towards AI communities that play a crucial role in advancing AI technology. For instance, researchers discovered critical vulnerabilities in the Safetensors conversion process of [Hugging Face](#), a widely used platform for downloading and building on pretrained models. An investigation by AI security firm HiddenLayer revealed that adversaries could exploit this process by submitting harmful code or compromised models via pull requests. When developers merge them into repositories, poisoned models can covertly compromise downstream systems.

Similarly, an organization's reliance on pretrained models from

third-party vendors also introduces supply chain risk. Training a foundational model for real-world use cases is resource-intensive, which is why many organizations often choose to deploy pretrained models from third-party vendors. However, these models could come with a hidden backdoor or poisoned weights, effectively weakening the organization's security posture.

A recent trend amplifying this risk is the increased adoption of low-rank adaptation (LoRA), a relatively cost-effective and simple technique to fine-tune base models for enterprise-grade agent use cases. LoRA functions like a lightweight "adapter" layered on top of a base model, aiding AI vendors to alter its behavior without modifying the core.

While this technique could help speed up AI adoption and agent deployment, it also creates an opportunity for adversaries to distribute LoRA files or code that might look harmless at first glance. However, once executed, these files can inject a backdoor or create triggers for data exfiltration. Because the base model remains unchanged and appears trustworthy, traditional security measures are

ineffective at detecting such tampering.

Beyond models, organizations must also be wary of the expansive software stack that supports agent operations, including misconfigured frameworks, libraries, APIs, and tools. Vulnerabilities in tools like [PyTorch](#) or [TensorFlow](#) are well documented by AI security experts. When agents rely on these tools to perform critical tasks, unpatched flaws could enable remote code execution, resulting in data theft.

Likewise, insecure API endpoints, weak authentication tokens, and poor encryption mechanisms can expose agent communications to interception or manipulation. The risks also extend to plug-ins and third-party modules, which can act as Trojan horses within agents. A compromised plug-in might perform unauthorized actions, execute malicious code, or even facilitate lateral movement across enterprise systems.

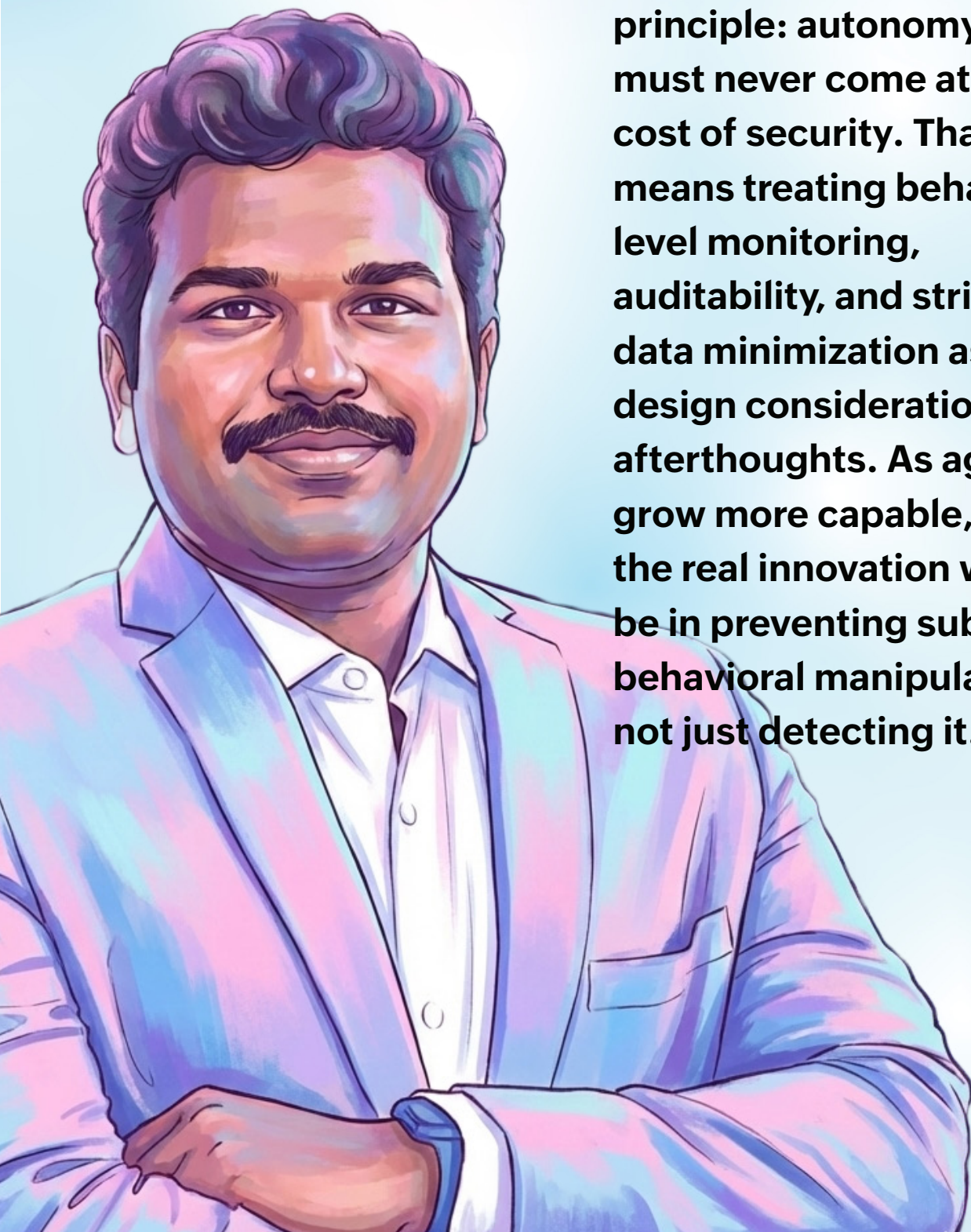
As AI agents become more deeply embedded in enterprise applications, security can no longer be an afterthought. It must be built into every layer of adoption. In the next chapter, we will explain how ManageEngine approaches this challenge.

A 3D shield with a rainbow gradient, featuring the chapter title. The shield is centered on a light blue background with a subtle gradient. The shield itself has a vibrant rainbow gradient, transitioning from purple at the top to blue, green, and yellow at the bottom. It has a 3D effect with a slight shadow on the surface below it.

CHAPTER 4:
THE
MANAGEENGINE
APPROACH

RAMPRAKASH RAMAMOORTHY,

**DIRECTOR OF AI RESEARCH,
ZOH0 CORPORATION.**



“Agentic AI will only be as trustworthy as the guardrails we build around it. At Zoho and ManageEngine, we design and continually refine our agents with a simple principle: autonomy must never come at the cost of security. That means treating behavior-level monitoring, auditability, and strict data minimization as core design considerations, not afterthoughts. As agents grow more capable, the real innovation will be in preventing subtle behavioral manipulation, not just detecting it.”

At ManageEngine, we take a pragmatic approach to creating software products. Instead of depending on external vendors for infrastructure or technical support, we prefer to build and own the technology stack. In the past, this philosophy has led us to design our own data centers, deploy a private cloud, and even source the hardware that powers them.

With AI, our approach has been no different. While some industry observers may perceive us as slow adopters, the reality is more nuanced. We have been carefully observing advancements in AI and steadily integrating them into our product suite in a thoughtful and purpose-driven manner. Before expanding into AI agents and agentic AI, we wanted to fully understand their security implications and determine how to counter them, all while staying true to our core tenets.

Many ManageEngine products today use a bring your own key (BYOK) model to integrate with third-party LLMs without exposing customer data to external providers. However, as the scale and complexity of enterprise use cases continue to grow, especially in scenarios that demand higher levels of autonomy and tighter orchestration with internal and external systems, we knew

we needed complete ownership of the foundational technology stack to deliver value without compromising data security and privacy.

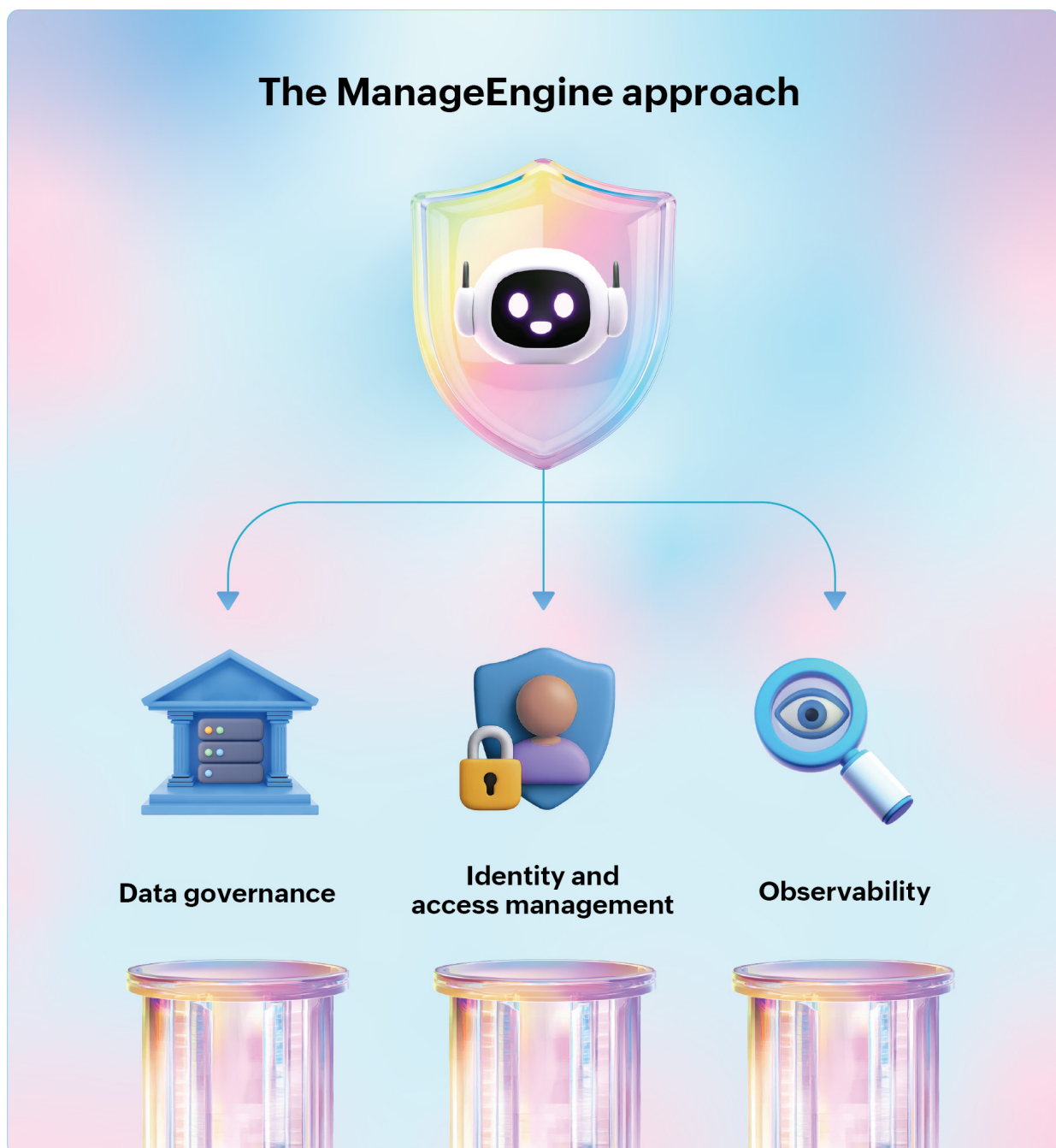
While this approach of building from the ground up is undeniably more resource-intensive, we firmly believe that the long-term benefits outweigh the trade-offs. That belief is now paying off. We have released our proprietary Zia LLM, built specifically for enterprise-grade use cases and hosted entirely within our infrastructure, along with Zia AgentStudio, our no-code AI development platform with full MCP support to enable agent actions.

Now that Zia LLM and AgentStudio have laid the groundwork, we are in the early stages of applying these capabilities across our product ecosystem. We are currently developing AI agents for our [IT observability](#) and [ITSM](#) platforms, while also assessing other products to identify workflows and use cases where agents can add meaningful value.

Similarly, on the security front, our AI security team is actively working to address the risks associated with behavioral manipulation. At this stage,

we have identified three pillars that guide our approach: data governance, identity and access management, and observability.

These pillars shape the guardrails and best practices described in this chapter. They reflect our current direction for securing AI agents, with some practices already underway and others in early conceptual stages. As we advance real-world deployments and learn from emerging threats, these practices will continue to evolve and mature.



DATA GOVERNANCE

The starting point for any organization looking to deploy AI systems is data governance. It is a systematic framework that guides enterprises in collecting, storing, optimizing, processing, and securing data. These steps are crucial because AI systems, especially autonomous agents, are only as effective as their training data. Scattered, inconsistent, or incomplete data can damage the agent's integrity and reliability.

From a security perspective, strong data governance becomes essential. As discussed in the data poisoning section, adversaries attempt to corrupt an agent's training or inference streams by introducing malicious datasets. Similarly, model inversion attacks can coerce an agent into leaking sensitive training data.

Data governance is integral to all that we do at ManageEngine. Our existing framework has long emphasized foundational practices such as maintaining a standardized semantic layer for data formats and enforcing a single source of truth (SSOT). As we expand into agent-driven workflows, we are now strengthening this framework with additional security-focused measures.

Here are some of the security measures we are incorporating.

✔ Using licensed or open-source datasets for training:

We primarily rely on licensed datasets from trusted providers to train our models. These curated sources help us avoid the risks associated with unverified training data and ensure that we obtain domain-specific samples suitable for our use cases.

When licensed datasets are not available, we turn to reputable open-source communities. In such cases, we conduct security and quality checks to detect any form of malicious contamination or inconsistency before the datasets enter our pipelines.

In addition to these sources, we are building mechanisms to generate synthetic data from internal metadata. This enables us to expand our training corpus safely while maintaining strong controls over data provenance and integrity.

✓ **Conducting real-time data validation:**

The real-time data entering the inference pipeline is also a common attack surface for adversaries. To counter this, we are setting up automated systems to validate incoming data.

Our systems verify the structure, quality, and provenance of the data, and quarantine anything that appears skewed, corrupted, or suspicious. These controls prevent compromised inputs from influencing the agent's behavior and help preserve both the security and reliability of deployed agents.

✓ **Tracing data lineage:**

AI agents are often perceived as black-box systems that deliver non-deterministic outputs, which makes investigating unexpected behaviors particularly challenging. To strengthen both transparency and security, we are introducing capabilities to trace the full lineage of data used within our training and inference pipelines.

By tracking where each input originated, how it moved through internal systems, and how it transformed before producing an output, we gain the visibility needed to detect anomalous patterns and isolate compromised data sources quickly.

This level of traceability not only aids in compliance but also provides a strong forensic foundation for analyzing and responding to security incidents involving AI agents.

✓ **Enforcing schema-level data masking:**

To minimize the risk of sensitive data exposure, we are in the process of restricting an agent's visibility to the structural metadata of a dataset. The agent can access the table schemas, column headers, and field definitions, while we keep the field values fully masked and encrypted.

When an agent performs a query, our systems route the actual data to the user or downstream system without revealing it to the agent itself. This approach ensures the agent can understand only the shape of the data it is working

with while preventing exposure to potentially sensitive or proprietary information.

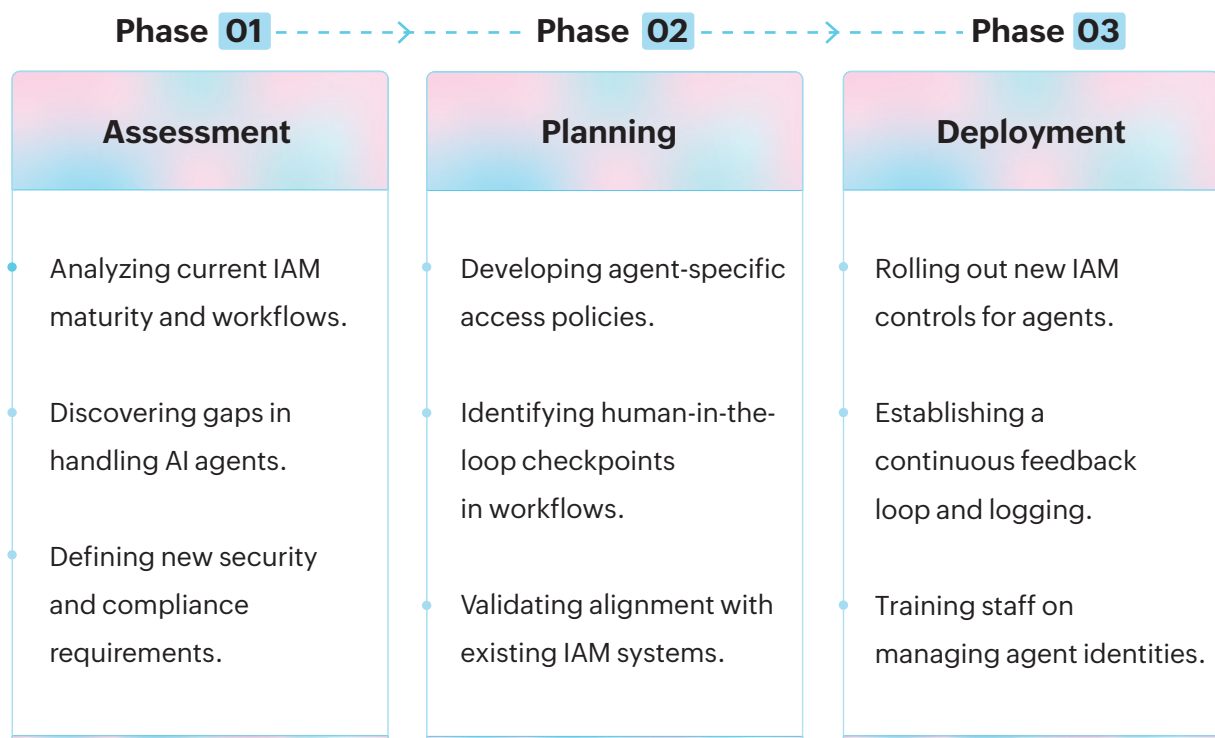
By decoupling schema awareness from data access, our goal is to reduce the risk of model inversion attacks, inadvertent data leakage, and unauthorized learning from sensitive fields.

IDENTITY AND ACCESS MANAGEMENT

After implementing a robust data governance framework, the next critical step for organizations deploying AI agents is identity and access management (IAM). As the enterprise IT management division of Zoho Corp., ManageEngine is no stranger to IAM. Then again, AI agents introduce complexities that conventional IAM models were not designed to handle.

Unlike human users or traditional systems, AI agents can access and share privileged data and make decisions that could have far-reaching consequences. This requires a fundamentally different approach to IAM.

To extend our IAM framework to support AI agents, we are developing a structured three-phase plan:



Here are the key practices we are in the process of adopting as part of this three-phase plan.

✓ **Creating unique agent identities:**

To support secure agent operations, we are implementing measures that ensure AI agents are never treated like human users. Each agent will receive a unique, verifiable identity with ephemeral credentials, rather than static API keys or generic service accounts. We are also incorporating controls that prevent credential reuse or sharing between agents, strengthening isolation and accountability.

✓ **Implementing life cycle management and granular categorization:**

We are in the process of establishing a structured life cycle model for each of our agents. Under this model, any agent deployed internally or embedded within a product will have its entire life cycle documented, from creation to decommission.

Additionally, we are classifying agents along granular dimensions such as interaction method (API versus GUI), degree of autonomy, and ownership. This methodical classification will enable us to align and reconfigure access controls with greater precision as our agent ecosystem grows.

✓ **Integrating human-in-the-loop checkpoints:**

Even though AI agents are autonomous, we believe human oversight is essential. We assign each agent a dedicated custodian responsible for periodic access reviews and recertification. Access tokens issued to agents will be time-bound and mapped back to this custodian for accountability.

Additionally, for mission-critical tasks, we are implementing “out-of-band” human authentication methods, such as push notifications or QR code approvals, to ensure explicit human validation before the agent executes high-impact actions. To operationalize these human-in-the-loop checkpoints, we are developing a “custom risk scoring logic” that categorizes every action an agent can perform based on its security impact.

For example, in a database management agent, the risk-scoring logic might classify a simple read operation as low-risk, while write or delete operations fall into the high-risk category. Whenever the agent encounters a high-risk action, it is designed to automatically pause and trigger a human authentication step before proceeding.

✓ Extending the principle of least privilege to agents:

Following the principle of least privilege, we are granting agents only the minimum set of permissions required to perform their tasks.

To achieve this, we plan to rely on short-lived tokens, explicit human approval for high-risk operations, and just-in-time (JIT) access. While these measures might impose limits on agent functionality, they significantly reduce the attack surface for behavioral manipulation.

How ManageEngine classifies human and agent identities

Property	Human Identity	Agent Identity
Lifespan	Years (tied to employment)	Ephemeral (seconds to minutes, tied to a task)
Provisioning	Manual / HR-driven onboarding	JIT provisioning via policy
Authentication	Passwords, MFA, SSO, passkeys	Advanced authentication mechanisms (SPIFFE, PKCE, DPOp)
Access Scope	Role- or attribute-based (RBAC / ABAC)	Task-specific with delegated permissions
Auditing	Session-level activity logs	Delegation-chain logs with full traceability
Life cycle Management	Rarely revisited (deprovisioning on exit/separation)	Continuous monitoring, automated expiry and revocation

OBSERVABILITY

Observability enables monitoring of the internal and external states of an AI agent to detect anomalies or deviations in its telemetry data. While robust data governance and access controls can help organizations reduce exposure to attacks like data poisoning during training and prompt injection, comprehensive observability is what enables them to catch subtler and more complex manipulation tactics like learning-stream poisoning, prompt injection, model inversion and extraction, and even certain supply chain exploits.

At ManageEngine, we already possess extensive observability capabilities for traditional deterministic software applications. However, AI agents are non-deterministic, which makes monitoring them far more challenging. As we work on expanding observability for agent-driven workflows, we have identified four fundamental questions that guides our approach: what breaks, what to track, what signals a problem, and what the debugging focus should be.

How observability changes		
Question	Traditional software application	AI agent
What breaks?	Entire application or application components	The reasoning model, decision chain, or connected tools
What to track?	Logs, uptime, error rates, latency	Prompts, decision paths, response quality, inference latency, API and LLM calls, failed tool calls, logs (user interaction and LLM interaction), traces, etc.
What indicates a problem?	Error codes, application crashes, slow responses	Model drift, rogue actions
What should be the debugging focus?	Inspecting deterministic logic and infrastructure vulnerabilities	Tracing reasoning and decision pathways

Based on these insights, here are the practices we are building into our observability framework for AI agents.

✔ **Monitoring model drift for hidden tampering:**

Model drift is not just a performance issue. In the context of AI agents, it can be a sign of data poisoning or subtle supply chain tampering. For example, if a threat-hunting agent that consistently flagged suspicious login attempts suddenly stops doing so, it might indicate that its learned patterns have been altered.

We are working on enhancing our observability systems to incorporate performance benchmarks and correlate them with shifts in response accuracy and behavior. This will help us detect deviations early and take corrective actions like rolling back or retraining before attackers can exploit them.

✔ **Tracking response quality as a manipulation signal:**

We are updating our monitoring systems to continuously observe the quality of agent responses across sensitive use cases. A rise in hallucinations or unusual changes in tone is not dismissed as a usability flaw. Instead, we treat these deviations as potential signs of compromised data sources, adversarial prompts, or hidden backdoors. This proactive stance will enable us to intervene before such responses lead to data leakage or compliance issues.

✔ **Measuring inference latency for probing attacks:**

Agents slowing down might not always be due to system load. Prolonged or abnormal inference latency can also signal model inversion attempts, where attackers submit repeated prompts to reconstruct training data. We are enhancing our latency-monitoring capabilities so that unexplained slowdowns automatically trigger an investigation into possible adversarial activity.

✔ **Examining failed tools and API calls for exploitation attempts:**

Failed tools and API calls are often dismissed as integration or infrastructure problems. However, in the context of AI agents, they might indicate vulnerabilities or exploitation attempts.

For instance, a failed API call could be the result of a network overload triggered by an adversary during a data exfiltration attempt. Or, it might stem from a weak integration configuration that adversaries could later exploit for unauthorized access. We are refining our monitoring tools to isolate these incidents quickly and trigger alerts that prompt deeper security investigation and remediation.

✔ **Expanding log collection for forensic depth:**

Conventional logs capture access events, authorization details, data movement, timestamps, and metadata. For AI agents, we are broadening our observability framework to include four additional categories:



By maintaining these logs, we aim to create a forensic baseline that helps correlate suspicious activity across multiple dimensions.

Four agent-centric logs

Name	Description	The security angle	Example
User interaction logs	Records all queries, inputs, and interactions between users and the agent.	Helps detect repeated adversarial prompting, malicious rephrasing, or attempts to slip in harmful instructions.	A user repeatedly modifies their queries until a hidden instruction causes the agent to disclose sensitive data.
LLM interaction logs	Captures all exchanges between the agent and the underlying LLM.	Enables detection of malicious prompt chaining or unusual token spikes that might indicate extraction attempts.	An attacker injects hidden instructions mid-conversation, causing the agent to go rogue.

Four agent-centric logs

Name	Description	The security angle	Example
Tool execution logs	Tracks which external tools/APIs an agent calls, along with the commands issued and results returned.	Identifies unauthorized or abnormal tool usage and failed or repeated API calls.	An agent repeatedly attempting unauthorized API calls, suggesting an attacker has gained access.
Decision-making logs	Documents how the agent chose a particular path (selected actions, scoring, tool choices, etc.).	Provides forensic evidence of bias, backdoor, or rogue behavior by explaining why the agent chose a certain path.	An agent consistently prioritizing a malicious data source, indicating tampered training or poisoned instructions.

 Recording detailed traces of each request:

While logs capture and record each event in isolation, traces map the full decision journey of the agent as it processes a request. We are working on expanding our tracing mechanisms to chronologically record everything from the initial user input and LLM calls to the final output.

This level of detail will enable our security teams to not only see what happened, but also how and why it unfolded. For instance, if a compromised plug-in injects false data mid-pipeline, traces reveal the exact insertion point, the agent's reaction, and the downstream effects. These enhancements are expected to accelerate root-cause analysis and help identify security loopholes before attackers exploit them.



THE FUTURE

[Gartner](#) predicts that by 2028, 33% of enterprise applications will feature agentic AI capabilities, and at least 15% of routine work decisions will be autonomously executed by AI agents. Enterprises that deploy and scale AI agents will gain a key competitive edge in the near future. Yet, it is essential to approach this with cautious optimism.

History shows that every major technological breakthrough has gone through cycles of mass adoption followed by significant drop-offs and course corrections. While organizational immaturity often plays a crucial role, the more enduring concern has always been security and privacy. Take the early days of cloud computing. Enterprises were vying to be early adopters of the public cloud, eager to tap into its vast potential.

Over time, many realized that modernizing legacy systems was more complex than initially anticipated, and the security trade-offs were far from negligible. After years of mounting costs and unaddressed vulnerabilities, businesses recalibrated, embracing hybrid and private cloud models.

Will agentic AI follow the same trajectory? Only time will tell. The explosive entry of generative AI tools and the productivity gains they deliver suggest promising potential. Yet, as this e-book has shown, today's AI agents are acutely vulnerable to behavioral manipulation.

Cyber adversaries are already weaponizing the very AI technologies that power these agents. Compounding this risk is the human factor, still the greatest Achilles' heel for most enterprises. For organizations exploring AI agents today, vigilance is non-negotiable.

Even long-standing cybersecurity tools and strategies are insufficient against the unique manipulation threats agents face. Enterprises must fundamentally revamp their security frameworks to integrate safeguards across every layer, from data governance and identity controls to observability and human intervention.

Not every organization will build its own LLM as ManageEngine has with Zia. Small and medium-sized businesses, in particular, will inevitably rely on third-party LLM vendors. That reliance, however, must be paired with rigorous vetting, multi-layered testing, and continuous monitoring to ensure no hidden weaknesses creep in. Regardless of size, stringent IAM practices and comprehensive observability are indispensable for compliance and security.

The future of agentic AI is not about blind adoption but about responsible integration. Enterprises that balance innovation with security will be the ones that reap the benefits of AI agents without falling prey to their risks.

About ManageEngine

As the IT management division of Zoho Corporation, ManageEngine prioritizes flexible solutions that work for all businesses, regardless of size or budget. ManageEngine crafts comprehensive IT management software with a focus on making your job easier. Our 120+ award-winning products and free tools cover everything your IT needs. From network and device management to security and service desk software, we're bringing IT together for an integrated, overarching approach to optimize your IT.



About the author

Arjun has nearly a decade of experience in content marketing, with a focus on long-form content. He has worked across multiple roles, building expertise in translating complex technology concepts into clear and well-structured narratives. His work involves researching complex topics and presenting them as thought leadership content for IT leaders. Outside of work, Arjun is an avid reader. He's also a sports enthusiast and always game for a match, a discussion, or enjoying the matches on television.



ManageEngine 