



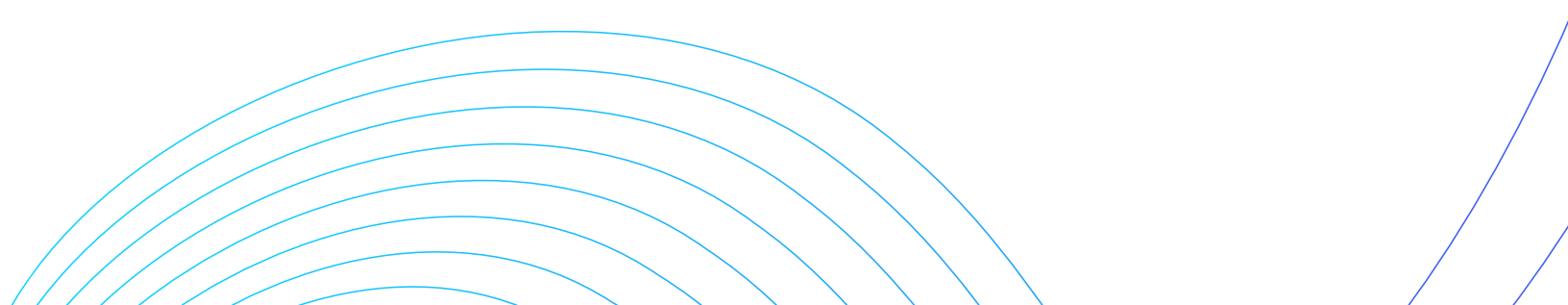
How to build AI agents into your SOC

A practical guide on how to build reliable AI agents for security operations—along with open source code and a workflow graph to get you started



Table of contents

03	Getting started with AI agents for security operations	
04	Design principles: The foundation of reliable AI agents	
06	Practical application: Endpoint forensics with OSQuery	
08	Model selection and optimization: The right tool for the job	
11	Security and governance: Trustworthy agents	
12	Success criteria: Defining what good looks like	
13	Implementation roadmap	
15	Orchestrate, then evaluate: The path to trust	
17	From one data source to many: How Zscaler MDR uses AI	



Getting started with AI agents for security operations

As we've discussed before, when integrated into a security operations center (SOC), AI agents promise real leverage for defense: faster triage, better context, and less swivel-chair work. They're designed to think and act like security analysts, using reasoning and tools to autonomously achieve complex security goals. But they also introduce new risks and operational questions.

Both security leaders seeking trustworthy outcomes and practitioners in search of workable AI guidance need a practical playbook. The thesis we're putting forward is simple: **Reliability beats novelty**. The most effective AI systems aren't the ones with the flashiest autonomy, they're the ones wrapped in strong workflows, governed by clear constraints, and measured relentlessly.

The pragmatic approach: Structure and determinism

The most common failure pattern we see in the industry is giving agents too much freeform autonomy. Large language models (LLMs) are probabilistic by nature: Run the same investigation 10 times and eight may complete as intended, but two might wander. That's unacceptable in a SOC, where repeatability matters.

At Zscaler, we pair deterministic orchestration with bounded agent reasoning. We break tasks into explicit steps, bound inputs, validate outputs, and use agents only where their probabilistic nature helps more than it hurts. Instead of letting agents go rogue, we guide them with clear procedures (e.g., step one, do this, step two, do that) so the model follows a path rather than improvising.

This hands-on guide lays out everything you need for developing and deploying reliable AI agents. Consider it a comprehensive resource you can revisit throughout your journey leveraging LLMs for security operations.

We've also released a LangGraph-based forensics analysis workflow for OSQuery that can be used to build your own agents for security investigations and incident response.



Get the source code for the AI agents, prompts, and OSQuery data on our [GitHub page](#).



This guide was adapted from Red Canary Director of Machine Learning Jimmy Astle's talk from [SecTor 2025](#).

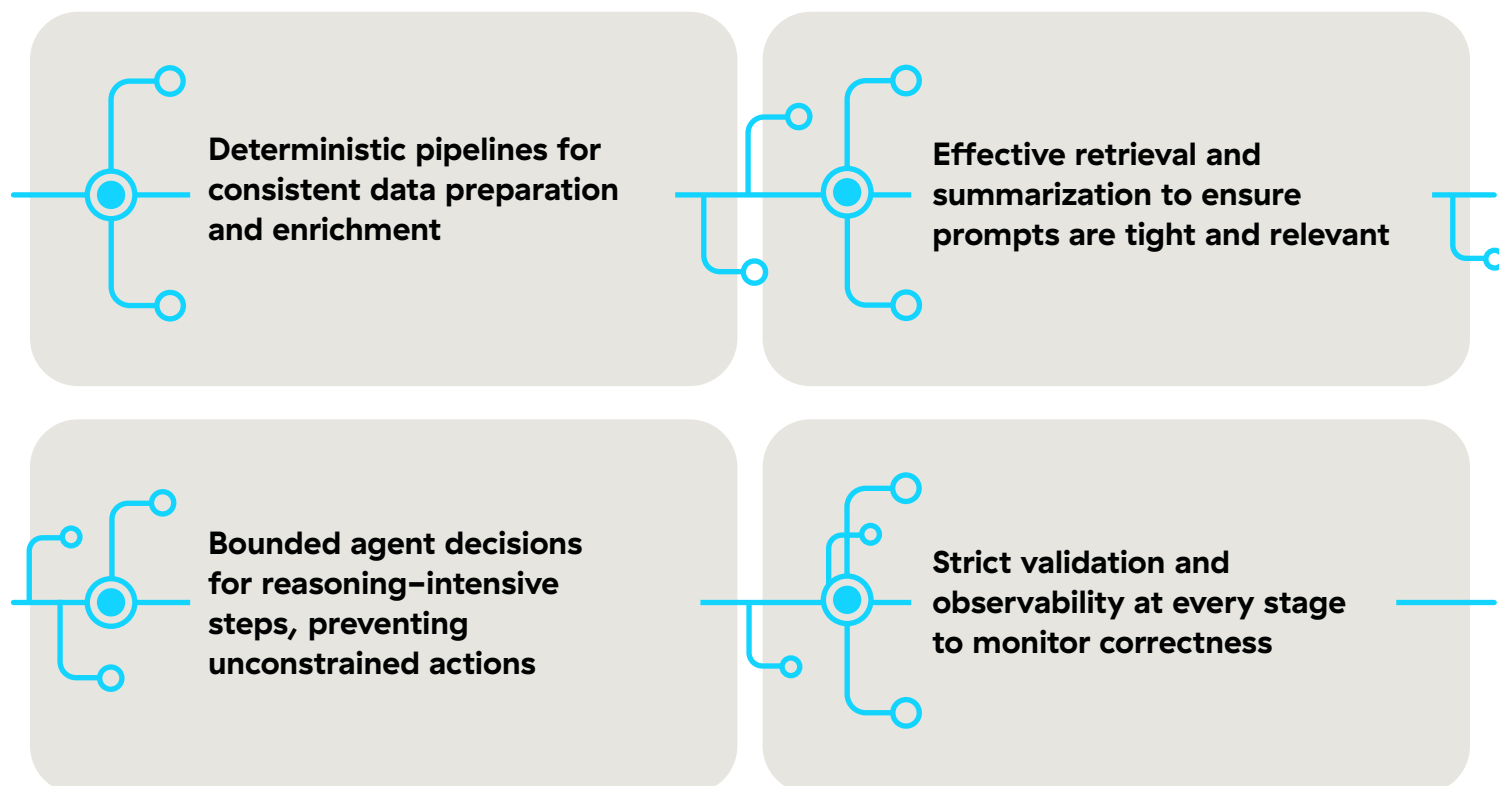
Throughout the guide, we'll provide both takeaways and action items for the broader concepts we cover, as this dual approach can help secure strategic buy-in and foster the trust needed when it comes to deploying effective, reliable AI agents that deliver value to security operations.

Design principles: The foundation of reliable AI agents



Building reliable AI agents for security operations isn't about one magical, all-encompassing model. It's about disciplined engineering that helps coordinate specialized agents and blending deterministic processes with probabilistic reasoning. Orchestration frameworks are essential for managing multi-step logic, maintaining investigation memory, enabling parallel processing for speed, and routing tasks to the right agent.

Reliable systems integrate:



Agents should be deployed where they excel at reasoning within defined parameters, wrapped by robust, deterministic scaffolding.

Key workflow patterns for reliability

Structure the workflow:

Break tasks into **small, explicit steps** to improve adherence and reduce surprises.

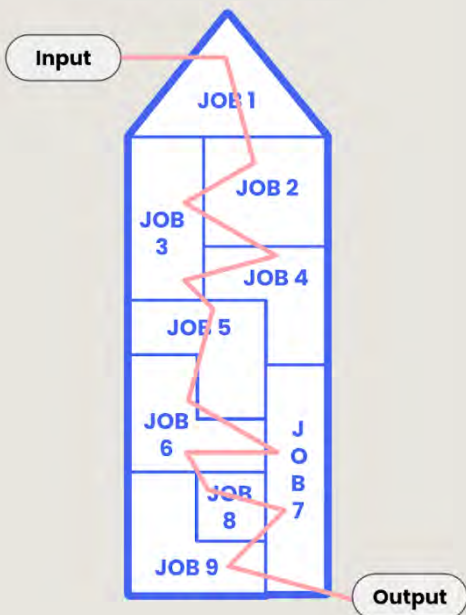
Bound the agent's choices:

Offer limited tools and inputs, rigorously validating outputs before downstream use.

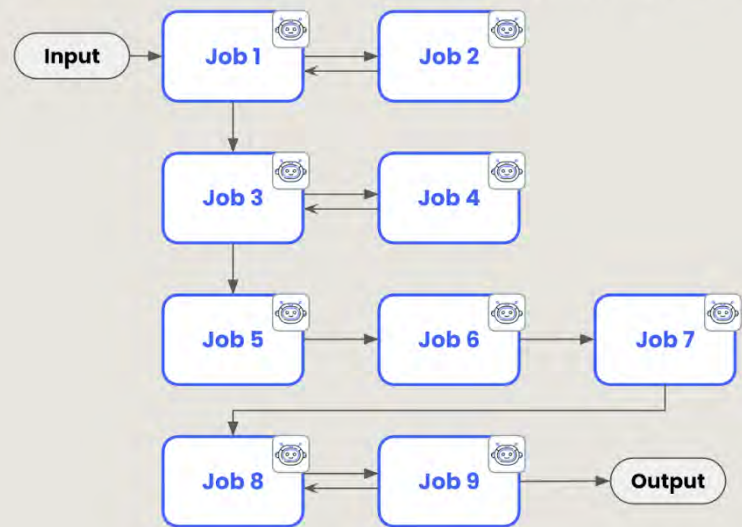
Design for the SOC:

Account for retries, human escalations, and audit needs.

Avoid the monolithic “blob”



Instead: A complex system built of simple components



Takeaways

You’re building a system, not a single model. ROI comes from the interplay of workflow design, guardrails, and model selection, not from chasing the latest benchmarks alone.

Next steps

- ☒ Use schema-constrained outputs, unit tests for prompts, and regression suites to detect drift when models or prompts change.
- ☒ Implement these as part of your continuous integration (CI) pipeline.

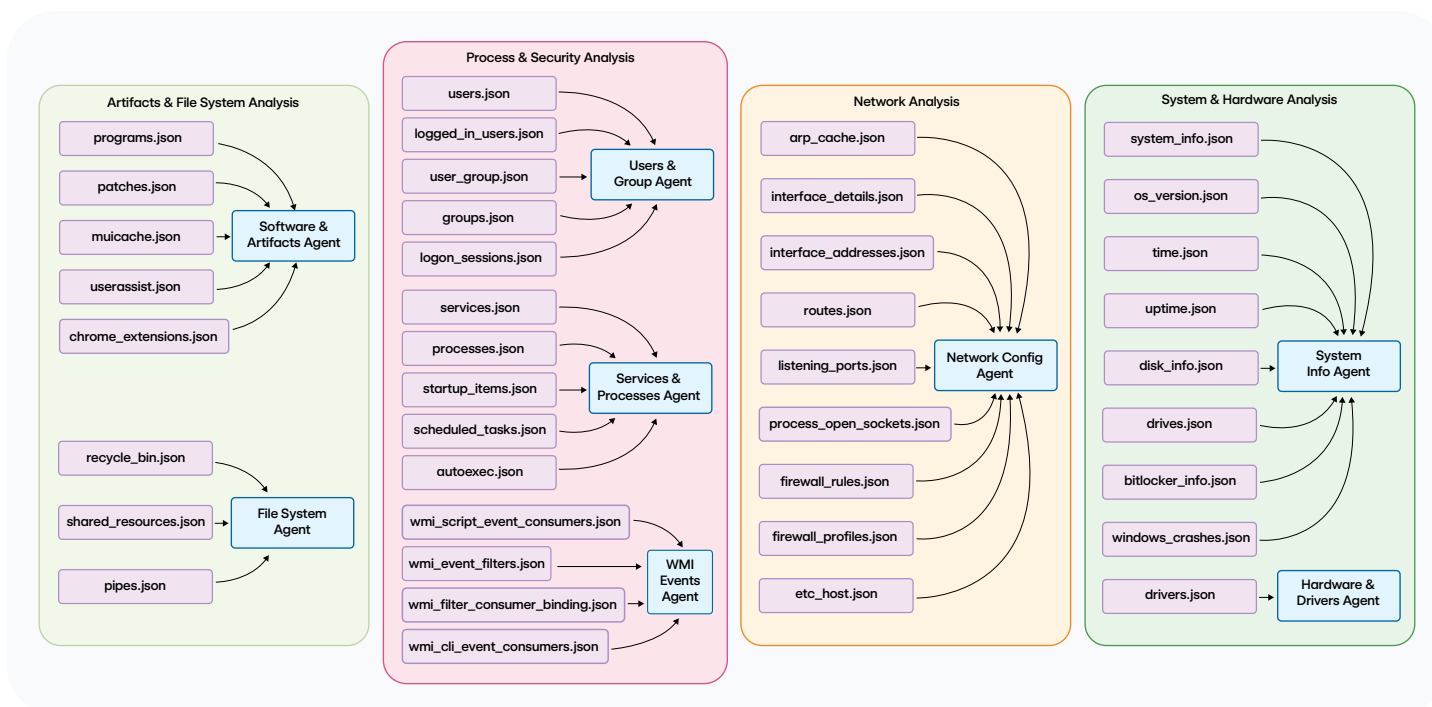
Practical application: Endpoint forensics with OSQuery



Consider a real-world endpoint forensics scenario: Using **OSQuery**, a tool that exposes operating system internals via a SQL interface, a human analyst could easily spend **30 minutes** or more sifting through three dozen JSON files from a single endpoint for a forensic report, especially during a high-stress incident. The goal for AI agents? Reduce this to under **two minutes**.

The secret lies in narrowly scoped agents. Instead of relying on one generalist agent, define specific responsibilities. For example, from 280 possible OSQuery tables, we might focus on 40 Windows tables across four categories: artifacts & file system analysis, process & security analysis, network analysis, and system & hardware analysis. Each category would be handled by a dedicated agent:

- A **software and artifacts agent** analyzes installed software and known artifact paths
- A **file system agent** enumerates suspicious directories or temporary locations
- A **users and groups agent** identifies anomalous local accounts
- A **WMI persistence agent** flags event consumers abused for footholds

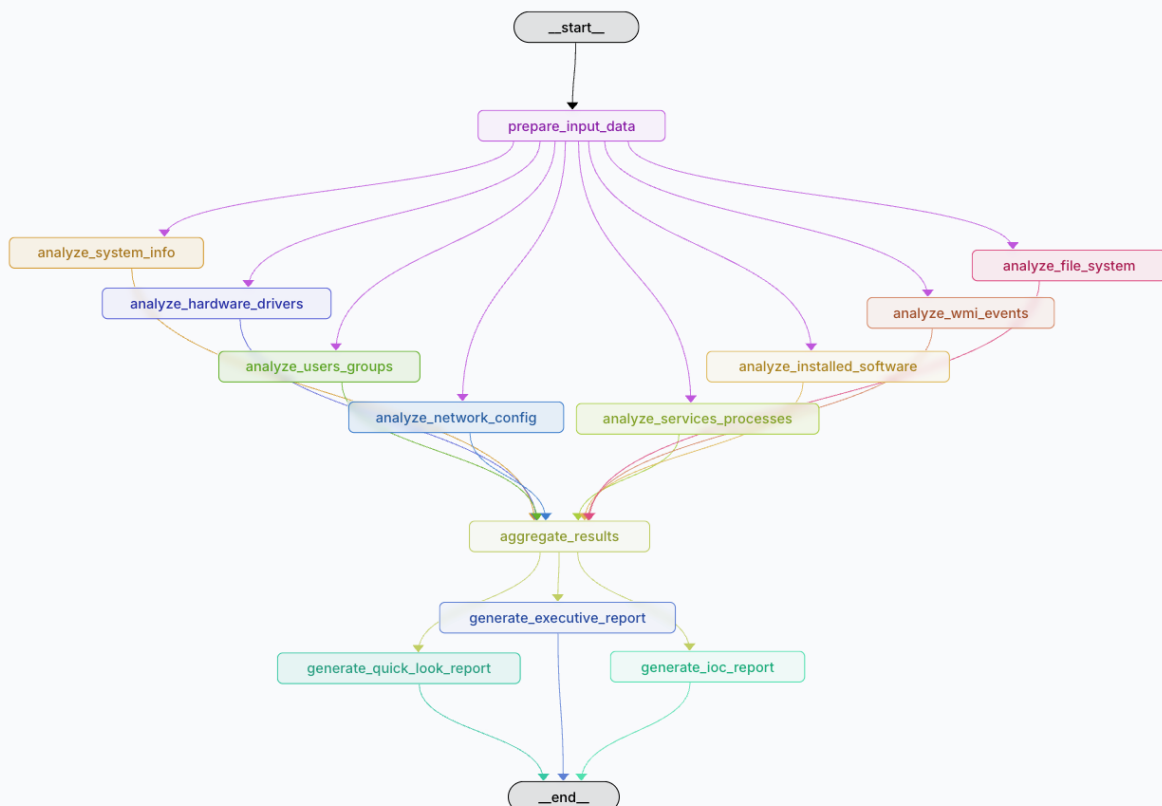


This narrow scoping ensures agents remain accurate, fast, and cost-effective, avoiding hallucinations and runaway costs. The true power comes from orchestrating these specialized agents to run in parallel, then aggregating their individual results into a comprehensive security report.

Context engineering makes this orchestration effective:

- **Assign a clear role:** You are an expert OSQuery-based forensics analyst specializing in Windows investigations.
- **Define precise tasks and objectives:** Analyze OSQuery data for Windows installed drivers; produce a concise JSON summary highlighting suspicious activities.
- **Inject only relevant context:** Provide serialized JSON from OSQuery queries, the exact SQL commands used, and corresponding table documentation. This prevents hallucinations by giving the AI precise context.
- **Force step-wise reasoning (chain of thought):** Break down analysis into explicit steps (e.g., Step 1: Review device IDs. Step 2: Check publisher trust. Step 3: Flag mismatches or known-bad hashes.).
- **Enforce an output contract:** Mandate JSON output adhering to a specific schema. Modern LLMs are fine-tuned for this, which is crucial for downstream processing.

An orchestration framework like LangGraph can help facilitate these complex, multi-agent systems through “fan out, fan in” workflows, see below, where independent agent tasks execute in parallel and their outputs are aggregated. This graph-based approach—LangGraph uses directed acyclic graphs, or **DAGs**—ensures predictable execution paths and allows for inserting deterministic nodes like validation checks, **human-in-the-loop (HITL)** interventions, or **LLM-as-a-judge** agents to verify outputs. It’s the agent backends and orchestration, not just the LLMs, that enable scalable, reliable AI in the SOC.



Model selection and optimization: The right tool for the job



Selecting the right model is a continuous trade-off between speed, capability, and cost. The optimal model depends entirely on the task.

Key considerations for selecting a model

- **Open source vs. commercial:** While open source models offer deployment flexibility, top commercial models often lead in broad cybersecurity knowledge and instruction-following, though this gap is closing.
- **Tiered SLAs and budgets:** Route work to the model best suited for the task's requirements. Some tasks demand advanced reasoning; others prioritize low latency or minimal cost.

In practice

- Use fast, “good enough” models for routine parsing and summarization. These can process data quickly and cheaply.
- Escalate to stronger reasoning models for aggregation nodes, correlation of signals, or generating executive narratives. The higher cost and latency are justified for these decision-heavy steps.
- Avoid a one-size-fits-all approach; intelligent routing is key to optimizing both performance and cost.

The following table is an example of how you might map the agents in the LangGraph workflow above to the AI models best suited for their respective functions.

Step	Agent	Function	Model
Preparation	Prepare input data	Formats OSQuery in JSON for analysis	N/A
Analysis	Analyze system info	Analyzes system info from OSQuery in JSON	GPT-4o
	Analyze hardware drivers	Analyzes hardware drivers from OSQuery in JSON	GPT-4o
	Analyze user groups	Analyzes user groups from OSQuery in JSON	GPT-4o
	Analyze network config	Analyzes network configurations from OSQuery in JSON	GPT-4o
	Analyze services processes	Analyzes processes from OSQuery in JSON	GPT-4o
	Analyze installed software	Analyzes installed software from OSQuery in JSON	GPT-4o
	Analyze WMI events	Analyzes WMI events from OSQuery in JSON	GPT-4o
	Analyze file system	Analyzes file system info from OSQuery in JSON	GPT-4o
Aggregation	Aggregate results	Aggregates the results from the analysis step	GPT-5
Generation	Generate executive report	Generates an executive summary from the aggregated data	GPT-5-mini
	Generate quick look report	Generates a quick look report from the aggregated data	GPT-5-mini
	Generate IOC report	Generates an IOC report from the aggregated data	GPT-5-mini

Context window management:

- Agents reason over provided context, which has a hard limit. Invest in high-quality retrieval and aggressive summarization to preserve essential signals.
- Chunk long artifacts, embed and retrieve relevant slices, and summarize aggressively.
- Track token spend and cap context sizes to ensure predictability. Smarter data preparation is more valuable than just bigger prompts.

Intelligent routing:

- Implement routing early. Even a simple rule like “fast model for summarization, stronger model for complex correlation” significantly improves latency and cost efficiency.
- Measure outcomes per task (accuracy, latency, token cost) to continually refine routing logic.



Takeaways

Match models to tasks by capability, latency, and cost. Revisit routing rules with evidence, not anecdotes.

Next steps

- ☒ Start with a simple router that escalates to stronger models when tasks cross a complexity threshold.
- ☒ Measure outcomes per task type: accuracy, latency, token cost.
- ☒ Use this telemetry to refine routing logic over time.

Security and governance: Trustworthy agents



As we've seen with [model hijacking](#) attacks and [AI command-line interface abuse](#), AI agents can be tricked or misused. Treat them as you would any externally exposed service: with robust security and governance.



Essential guardrails

Monitor prompt injection and jailbreak resistance. Sanitize inputs, constrain untrusted content, and embed explicit model rules.



Tool isolation

Allowlist tools, enforce input schemas, and simulate tool calls in test harnesses before production.



Data governance

Restrict access to personally identifiable information (PII), scope retrieval to approved data stores, and meticulously log all access and actions.

In practice

- Clean and normalize OSQuery outputs before any LLM sees them. Untrusted data must be sanitized.
- Keep tool calls explicit and bounded. Validate inputs and outputs for external enrichment tools (e.g., VirusTotal) and consider sandboxing.
- Log every tool invocation, model call, and data retrieval with correlation IDs for auditability and incident response.



Takeaways

Make guardrails a requirement, not a bolt-on. Tie deployment approvals to passing security checks. Confirm data constraints early: Will you run on-premises or on a hyperscaler? Can you meet internal data handling requirements and avoid third-party data exposure?

Next steps

- ☒ Implement content filters, schema validators, and sandboxed tool executors.
- ☒ Add red-team prompts and adversarial tests to your CI.
- ☒ Instrument everything: latency, token use, step-wise success, and reasons for fallback.

Success criteria: Defining what good looks like



High-quality reports? Faster analysis time? Before implementation, clearly define your goals. The outcomes should directly drive design and evaluation:



Faster time to signal

Analysts receive concise, relevant context immediately, reducing human investigation time (e.g., 30+ minutes to under 2 minutes).



Repeatable investigations

Consistent inputs yield consistent outputs or clear fallbacks, with full auditability.



Cost-aware scaling

Adjust model choice by task importance, avoiding top-tier costs for routine operations.

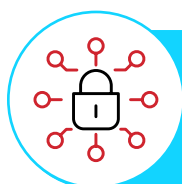


Accuracy

Are the agent's findings correct? Evaluate against known incident data to measure false positives and negatives.

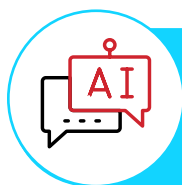
Implementation roadmap

Build your AI agent capabilities iteratively with the following phased approach.



Phase 1: Foundation

- **Confirm data constraints:** Establish hosting, residency, and third-party exposure requirements.
- **Pick a framework:** Favor mature, open source agent ecosystems over custom builds (e.g., LangGraph).
- **Define variability:** Determine where deterministic vs. creative agent behavior is acceptable.
- **Set measurable goals:** Define metrics like time-to-triage, analyst satisfaction, accuracy, and unit costs, linking them to review cadences.



Phase 2: Quick wins

- **Procedural prompts:** Transform vague prompts into step-by-step instructions.

Instead of
investigate
this alert.

try:

Step 1:
Summarize
the alert.

Step 2:
Enrich with
threat intelligence.

Step 3:
Identify affected
assets.

- **Schema validation:** Implement strict validation and error handling for all model outputs.
- **Simple router:** Default to cost-effective models, escalating to stronger ones for complex tasks when a complexity threshold is crossed (e.g., requires multi-step reasoning or domain expertise).
- **Instrument everything:** Track latency, token use, success rates, and fallbacks for tuning.
- **Evaluation plan:** Track success/failure per step and prompt; add system/integration tests to CI.



Phase 3: Maturity

- **Refine routing:** Use accumulated metrics to optimize model selection rules.
- **Expand test coverage:** Add regression, adversarial prompt, and tool simulation tests to CI.
- **Optimize context management:** Iterate on chunking, retrieval, and summarization strategies, improving accuracy while reducing token spend.
- **Iterate workflows:** Review analyst feedback and metrics quarterly to adjust workflows.
- **Mature governance:** Expand audit logging, implement role-based access, and document escalation procedures.



Takeaways

Structure reduces risk and enables measurement. Demand deterministic tasking for high-risk/high-volume workflows, using agents where variability is acceptable with guardrails.

Next steps

- ☒ Encode deterministic checks around each step.
- ☒ Provide bounded choices and enforce schema validation.
- ☒ Add timeouts, retries with diversity, and human-in-the-loop escalation.
- ☒ Log everything for fast failure analysis.

Orchestrate, then evaluate: The path to trust



Reliability is forged through orchestration and evaluation.

- **Orchestrate with graphs:** Use frameworks like LangGraph to fan-out parallel agents for independent tasks and fan-in their results. Integrate deterministic nodes for validation, human review, or “LLM-as-a-judge” checks, ensuring speed without sacrificing control.
- **Evaluate speed and cost:** Instrument token throughput, output token counts, and end-to-end latency. Workhorse models can complete multi-agent forensic workflows in under two minutes for mere cents. Reserve high-reasoning models for critical aggregation steps where their cost is justified.
- **Measure consistency and accuracy scientifically:** Use [semantic similarity algorithms](#) to detect output drift; trigger human review if similarity falls below a threshold (e.g., 80 percent). Employ LLM-as-a-judge agents to assert the presence of critical and expected elements (hostname, severity) in outputs. Apply [YARA-style signatures](#) in workflows for brittle but precise constraints (e.g., maximum summary length).

Ultimately, reliability builds trust. Analysts need to believe in and use the outputs. Consistent schemas, deterministic boundaries, clear fallbacks, and transparent logs are crucial for gaining that confidence.

Common pitfalls and how to avoid them

- **Monolithic “do-it-all” agents:** Break down tasks into narrowly scoped, specialized agents.
- **Overstuffed prompts:** Curate relevant context; summarize aggressively to keep prompts concise.
- **One-size-fits-all models:** Implement intelligent routing based on task complexity, cost, and latency.
- **No output contract:** Always enforce and validate JSON schemas for structured outputs.
- **Missing guardrails:** Sanitize inputs, isolate tools, and log all agent actions.
- **Shipping without telemetry:** Instrument everything to enable continuous tuning and optimization.
- **Neglecting:** Agents require ongoing review, updates, and adaptation as models and data evolve.

Build trust, not hype

For blue teams, stability beats unproven innovations. Build for **determinism** where it matters, use agents where they shine, and pick models with intent. That's how you deliver value that leaders can trust and practitioners won't outgrow.

Reliable AI agents for security operations emerge from thoughtful design, meticulous orchestration, narrow task definition, and rigorous measurement. When implemented effectively, these agents empower the SOC by accelerating time-to-signal, ensuring repeatable investigations, and providing cost-aware scalability.

Define clear roles, tasks, objectives, and outcomes for each agent, ensuring every component serves a precise purpose. This, combined with enforcing structured outputs and deterministic orchestration, can create predictable, auditable workflows vital for building trust. Intelligent workload routing is crucial, involving the selection of agents based on their specific capabilities, desired latency, and cost-effectiveness, with continuous refinement driven by performance data.

By following this roadmap, your SOC can transition from arduous, manual data sifting to highly efficient, sub-two-minute triage and the delivery of consistent, high-quality reports. The true reward isn't just speed; it's the confidence that comes from predictable outcomes, robust guardrails, and a system that continuously earns the trust of your security team. Start small with a single workflow, apply these principles, and iterate. The results will speak for themselves.



Takeaways

Invest in the entire system—workflow design, guardrails, and routing—rather than solely chasing the latest model benchmarks. Demand concrete evidence of speed, accuracy, consistency, and cost to build trust with analysts and justify investment.

Next steps

- ☒ Keep agents narrow, prompts procedural, outputs schema-validated, and orchestration fully observable.
- ☒ Integrate model routing and adversarial tests into your CI/CD pipelines.
- ☒ Instrument token throughput, latency, and failure modes at every step.
- ☒ Implement robust timeouts, diverse retry mechanisms, and clear human-in-the-loop escalations for complex edge cases.

From one data source to many: How Zscaler MDR uses AI



This guide and [supporting materials on GitHub](#) specifically describe how security teams can leverage AI to automate the analysis of signals from OSQuery. However, the underlying processes and philosophies are precisely the same ones that we've leveraged to build the agentic AI workflows that detect threats faster and with more accuracy.

While OSQuery is a single data source—albeit a robust one—Zscaler is leveraging AI agents to perform tasks like those described in this guide across a wide variety of disparate data sources. The volume of data and the number and diversity of the agents we're using for threat investigation is far greater than what is described in this guide, but the principles remain the same.

We give our agents formal, clearly defined roles

We formalize roles and job descriptions for our AI agents as if they are real people. Personifying agents in this way is helpful because:

- You can use existing organizational language for roles and responsibilities.
- It's easier to discuss what an agent's job is.
- It helps prevent burdening agents with vague or overly broad jobs.
- It makes agents easier to understand and talk about.

Identity Investigation agent



Job: Identify suspicious identity activity by triaging and investigating alerts and relevant log and intelligence data.

We make sure these roles and their definitions are simple and tightly scoped

It's our longheld philosophy that the most reliable complex systems are those that consist of many simple but reliable components. We've applied this in the software that we've developed for years, and we're continuing to apply it to the AI systems we develop. As such, we have a lot of agents, but each is focused on a specific task and a specific data source.

For example, we have an agent that is responsible for investigating endpoints using data from Defender for Endpoint while a separate agent does the same for SentinelOne. Specializing in this way helps develop expertise.

SOC Analyst agents

Identity Investigation agent,
CrowdStrike Falcon Identity

Identity Investigation agent,
Microsoft Entra Identity Protection

Identity Investigation agent,
Microsoft Cloud App Security

Identity Investigation agent,
Okta Workforce Identity

Identity Investigation agent,
User Baselining & Analysis

Identity Investigation agent,
Cisco Duo

Cloud Security Investigation agent,
AWS GuardDuty

Endpoint Investigation agent,
Microsoft Defender for Endpoint

Endpoint Investigation agent,
SentinelOne

Endpoint Investigation agent,
Native analytics for Endpoints

SIEM Investigation agent,
Microsoft Sentinel

SOC Event Investigator agent,
Threat Analysis

SOC Event Investigator agent,
Notes & Annotations

SOC Event Investigator agent,
Intelligence Insights

SOC Event Investigator agent,
Platform & Tool Assistance

Phishing Triage agent,
Microsoft Outlook

Intelligence agents

SOC Event Investigator agent,
Intelligence Insights

Incident Response agents

SOC Event Investigator agent,
Response & Remediation

We insert our agents into defined workflows based on dependable SOC processes

We document analyst processes and build accurate workflows. We also interrogate the work analysts are doing at each step to identify opportunities where AI agents can safely and accurately do analytical work faster than humans.

KNOW YOUR SYSTEM & WORKFLOW

- | | |
|--|--|
|  Identify or assign process owner |  Analyze data sources & quality |
|  Engage stakeholders |  Assess fit for AI & automation |
|  Map current workflow |  Establish goals & objectives |
|  Identify choke points & dependencies | |

We translate rote standard operating procedures into prompts

When you find consistent and repeatable work that's being done by analysts during the investigations, you can take their standard operating procedures and turn them into prompts that AI agents can follow to do that work, particularly when it's boring and repetitive work that prevents human analysts from doing something more productive and fulfilling. We maintain version control and continuously refine these processes over time, starting with human reviews as if these agents are new hires in a probationary period.

Identity Investigation agent



Job: Identify suspicious identity activity by triaging and investigating alerts and relevant log and intelligence data.

- Extract user/IP data, authn logs, logins
- Analyze login attempts, MFA status
- Assess IP/VPN/geo for anomalies
- Examine device compliance
- Compare session to historical baseline

Experience

- ☐ Beginner
☒ Intermediate
☐ Advanced

Repetitions

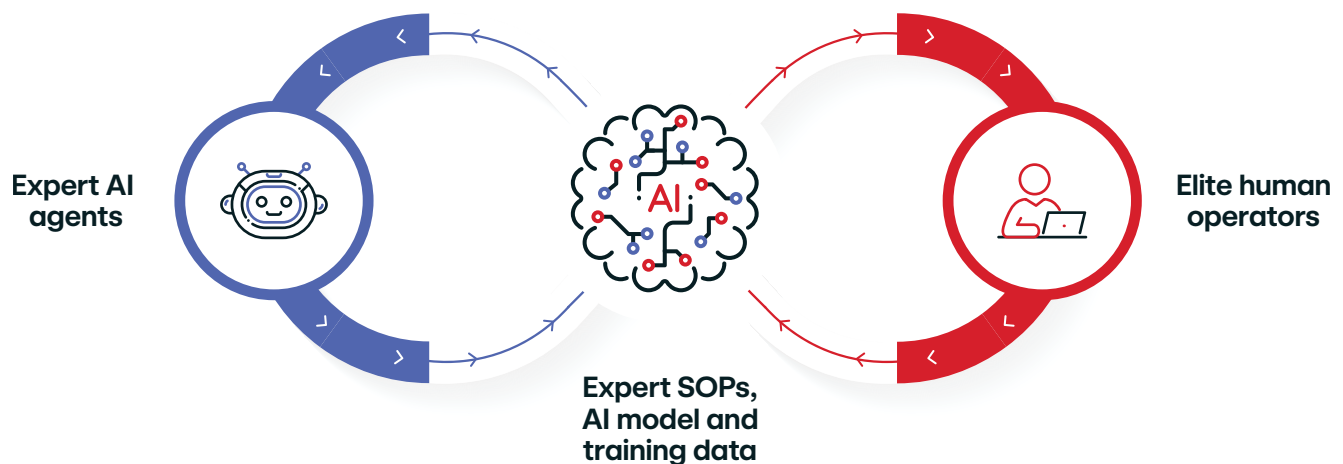
900,000

We continually improve our agents with feedback and tuning

We review workflows and agent performance overtime with a particular focus on continual training and improvement. After tracking what activities matched which analytics to train threat-finding agents, we examine what helped make investigations better to improve investigation agents and gather feedback on the usefulness of recommendations to improve communication/response agents. It's an ongoing performance review process for our agents.

Customer outcomes

- 72% fewer tickets
- 5x faster investigations
- 10x faster response



Learn how our human-led, AI-powered managed detection and response (MDR) can augment your SOC

[LEARN MORE](#)

About Zscaler

Zscaler (NASDAQ: ZS) accelerates digital transformation so customers can be more agile, efficient, resilient, and secure. The Zscaler Zero Trust Exchange™ platform protects thousands of customers from cyberattacks and data loss by securely connecting users, devices, and applications in any location. Distributed across more than 150 data centers globally, the SSE-based Zero Trust Exchange™ is the world's largest in-line cloud security platform. Learn more at zscaler.com or follow us on Twitter @zscaler.

© 2026 Zscaler, Inc. All rights reserved. Zscaler™ and other trademarks listed at zscaler.com/legal/trademarks are either (i) registered trademarks or service marks or (ii) trademarks or service marks of Zscaler, Inc. in the United States and/or other countries. Any other trademarks are the properties of their respective owners.