seemplicity

/whitepaper

The Missing Layer

What AI Agents Need to Actually Close Security Findings

EXECUTIVE SUMMARY

Security teams are deploying AI agents to investigate vulnerabilities faster. What they're finding: investigation is just the beginning. This paper examines what it actually takes to run a remediation program at enterprise scale, and why AI alone can't get you there.

- **Millions of raw findings, one data problem.**
  Enterprise environments contain findings from 6 to 10 scanning tools with incompatible schemas. Before any AI agent can investigate a single finding, 70–80% of raw output must be deduplicated, normalized, and prioritized. That process was built over years of production deployments, not generated by a language model.
- **Investigation is 20% of the problem.**
  AI-powered investigation of individual findings covers roughly 20% of the remediation lifecycle. The remaining 80% is operational: SLA enforcement, exception governance, ownership accuracy, and audit trail maintenance.
- **You can't build the foundation with AI alone.**
  General-purpose AI can prototype connectors and normalization logic, but can't reach production quality without 12–18 months of hardening per integration and deployment-scale organizational learning.
- **Compliance requires more than agent outputs.**
  SOC 2, PCI DSS, and cyber insurance requirements demand deterministic, timestamped, immutable evidence of remediation decisions. Probabilistic AI outputs, without a system of record, don't satisfy these requirements.
- **The architecture that works.**
  Seemplicity's context fabric (data normalization, asset identity, organizational intelligence) combines with AI-powered investigation, an operational lifecycle layer, and a continuous compliance evidence trail to deliver a complete, auditable remediation program.

The AI Moment in Security:

What Changed and What Didn't

Frontier AI models are remarkable. In 2026, a security engineer can point Claude, GPT, or Gemini at a SAST finding, give it access to a GitHub repository and a Jira board, and watch it trace a call graph, assess reachability, identify the code owner, draft a fix, and create a ticket. For a single vulnerability, this works. It's genuinely impressive. But it raises an obvious question: if AI can do this for one finding, why would you need a platform?

That's the right question. The answer is also why AI and Seemplicity aren't in competition. They're solving two different parts of the same problem.

Investigating one finding is an intelligence problem: understanding what a finding means, what it affects, and how to fix it. Running a remediation program is something else. It's a data, orchestration, and operational knowledge problem that requires knowing which of two million findings deserve investigation, routing work to the right owners, tracking SLA compliance across dozens of teams, and producing auditable evidence that risk is being reduced systematically. **AI handles the first part brilliantly. Without a purpose-built foundation underneath it, it can't handle the second.**

This paper walks through that foundation: what it contains, what it takes to build, and what organizations keep running into when they try to skip it.

01 Part One: The Last Mile, Where AI Shines

AI's contribution to the remediation workflow is real. The "last mile" of security remediation is deep investigation of a specific finding. That's exactly where large language models excel. This is the analyst workflow that used to take hours per finding:

- **Is this real?**
  Tracing whether vulnerable code is actually reachable in the production execution path.
- **How bad is it?**
  Assessing blast radius, network exposure, and business criticality of the affected asset.
- **Who owns it?**
  Identifying the developer or team responsible through git blame, contributor history, and project structure.
- **How do we fix it?**
  Generating a code diff, step-by-step remediation instructions, and an effort estimate.
- **Where does it go?**
  Routing a fix-ready ticket to the right Jira project, with the right priority and SLA.

An AI agent with tool access can do all of this for a single finding in minutes. Seemplicity's own AI Analysts (specialized agents for code security, cloud misconfiguration, container vulnerabilities, and more) do exactly this, running continuously across customer environments. We don't compete with AI on the last mile. We use it. The model is an input to our platform, not our competitor. But the last mile is only the last mile. Before any agent can run, there's a prior set of questions that no model can answer from a cold start.

02 Part Two:

Before the Agent Runs, The 99% Problem

An enterprise security program doesn't have one finding. It has 2,500,000. They come from 6 to 10 different scanning tools, each with its own schema, its own severity scale, its own asset naming conventions, and its own definition of "critical." Before any AI agent can investigate a single finding, someone or something has to answer:

Which 2,500,000 findings are actually 3,000 actionable items?

Forty Snyk findings, twelve Wiz alerts, and three Tenable results might all describe the same underlying Log4j exposure across the same asset cluster. Without deduplication and root-cause grouping, an AI agent would investigate each one independently, generate 55 separate tickets, and waste both compute and developer time on redundant work. Seemplicity's aggregation engine collapses raw scanner output by 70–80% before any agent touches it.

This isn't a prompting problem. It requires a normalized data model that understands semantic relationships between findings across tools, maps them to a unified asset identity, and groups them by root cause. That model was built over years of customer deployments. A language model didn't generate it.

Which of those 3,000 items actually deserve deep investigation?

Running an AI agent against every finding is economically impractical and operationally noisy. Figuring out which findings warrant agentic investigation, based on exploitability signals, business criticality, SLA status, and organizational policy, requires a pre-built decision framework anchored to each customer's specific risk posture. That's policy, not inference.

What does "this asset" even mean across six tools?

The same EC2 instance might appear as i-0abc123 in AWS, 10.0.3.47 in Tenable, prod-web-7 in the CMDB, and an ARN in Wiz. An AI agent querying one tool has no way to know these are the same machine, let alone that it belongs to the Platform-Core team, sits in a PCI compliance zone, and has an SLA that expired two weeks ago. Seemplicity's entity resolution layer maintains a continuously updated, cross-tool identity graph that resolves these ambiguities, built from thousands of real-world mapping patterns across hundreds of customer environments.

> **An AI agent can analyze a finding. But it can't tell you that the finding it's analyzing is one of 47 duplicates, that the asset has already been flagged by three other tools, and that the team responsible just shipped a broken fix for the same CWE last sprint. That's not intelligence. That's institutional knowledge.**

03 Part Three:

Can an AI Agent Build the Data Fabric?

Here's where it gets interesting. If the data fabric is the foundation, and AI agents are increasingly capable at writing code, could a team just instruct Claude or a coding agent to build the equivalent of Seemplicity's context fabric from scratch?
The honest answer: partially, in simple environments, if you have the time. Here is what that effort actually looks like.

Integration Connectors

A coding agent can write a working Snyk or Wiz API connector in an afternoon. For well-documented APIs with clear schemas, this is a solved problem. But there's a real gap between "I can pull findings from Snyk" and "I can reliably sync two million findings every six hours, handle rate limits, manage token refresh, perform incremental updates, recover from partial failures, and adapt when Snyk ships a breaking API change." That gap is the difference between a prototype and infrastructure.

Seemplicity maintains over 170 production-grade, bidirectional integrations. Each one encodes years of edge-case handling, customer-specific configuration options, and schema evolution management. A coding agent builds the happy path. The remaining 20% of capabilities, the part that accounts for 80% of the engineering effort, is production hardening.

Normalization and Unified Ontology

Mapping fields from one scanner to a common schema is straightforward. Making semantic decisions across tools is not. Is a Wiz "misconfiguration" the same category as a Tenable "compliance failure"? Is a Snyk "high" the same risk level as a Qualys "severity 4"? These are judgment calls that vary by tool version, by scan configuration, and often by individual customer. Seemplicity's normalization ontology was refined through hundreds of deployments. A coding agent produces the naive version, field-name matching with assumed severity equivalence, and misses the 30% of cases where that assumption is wrong.

Asset Identity Resolution

Entity resolution is one of the hardest classes of problems in data engineering. It requires reconciling different naming conventions across tools, handling many-to-many relationships where IPs rotate across instances, managing ephemeral cloud infrastructure where identifiers are transient, and merging data from sources that share no common key. A coding agent handles exact-match cases well. The fuzzy matching, temporal correlation, and topology-based inference that handle the hard cases were built from patterns observed across thousands of real customer environments. That knowledge doesn't exist in any model's training data.

Organizational Intelligence

This is where the "build it with AI" argument breaks down most clearly. No coding agent can generate the following from API calls alone:

- That the Platform Jira project was reorganized last quarter and the old routing rules are stale.
- That the developer who git blame says owns the code left the company two months ago.
- That findings on PCI-scoped assets require a different exception workflow than internal tools.
- That this team has a pattern of rejecting vulnerability tickets as won't-fix, which means SLA tracking needs to account for that behavior.
- That the last four fixes pushed by this team for this CWE category broke production, so the effort estimate should be flagged as high-risk.

This is accumulated organizational state. It doesn't come from any API. It lives in the relationships between systems and in the behavioral patterns that only emerge over months and years of operating inside a customer's environment. An AI agent on day one has zero organizational memory. Seemplicity on month twelve has a rich, verified model of how that organization actually remediates.

The Build-vs-Buy Reality

Layer	Can AI Build v1?	Can It Reach Production Quality?	Core Challenge
Connectors (170+)	Yes, for major tools	Needs 12–18 months of hardening per connector	Edge cases, rate limits, schema drift
Normalization	Naive version, yes	Requires years of customer feedback	Judgment calls vary by customer
Asset Identity	Simple cases only	Fuzzy matching requires deployment-scale learning	No common key across tools
Org Intelligence	Cannot be generated	Requires months inside the environment	Data doesn't exist in any API
Governance & Audit	Structure, yes; trust, no	Auditors need deterministic trails, not agent logs	Regulatory compliance

04 Part Four: After the Agent Runs, The Operational Loop

Suppose the agent investigates a finding perfectly. The work isn't done. For most organizations, the investigation is roughly 20% of the remediation lifecycle. The other 80% is operational:

- **Was the ticket created in the right project?**
  With the right priority, the right SLA, the right assignee?
- **Did the fix actually ship?**
  Is the PR merged? Did the deployment succeed? Did the scanner confirm the finding is closed?
- **What if the team pushes back?**
  Exception workflows, risk acceptance documentation, escalation paths. Who manages them?
- **Can you prove it?**
  When the auditor asks for evidence that every critical finding was triaged, routed, and resolved within SLA, where does that evidence come from?

A one-shot AI agent has no memory of what it did yesterday. It doesn't track SLA compliance. It doesn't know that this finding was already investigated last month and deprioritized by the security lead. It can't produce a compliance report showing mean-time-to-remediation trends by team across the last four quarters.

Seemplicity maintains the full remediation lifecycle as a system of record: every finding, every decision, every ticket, every exception, every SLA breach, with a complete audit trail showing which agent or human made each decision and why. This is what turns individual agent actions into a managed, auditable, and continuously improving remediation program.

05 Part Five: Proving You Are Secure

Here's the dimension of the remediation problem that gets overlooked most often, but that dominates how security programs are actually judged: the ability to prove that exposure is being managed systematically, with evidence, over time.

Boards, regulators, auditors, and cyber insurers don't ask whether your team can investigate a vulnerability. They ask whether you can demonstrate a governed, repeatable process for identifying, prioritizing, remediating, and validating every exposure across your environment, and whether you can produce the evidence trail to prove it. This isn't a technical problem AI can reason through. It's an institutional requirement that has to be baked into how remediation happens.

What Auditors Actually Ask For

When a SOC 2 auditor, a PCI assessor, or an internal risk committee reviews your vulnerability management program, they're looking for specific, concrete evidence:

- **Completeness:**
  Can you show that every finding from every scanner was ingested, normalized, and accounted for, not just the ones someone decided to look at? Auditors want to confirm there is no gap between what was found and what was tracked.
- **Timeliness:**
  For every critical and high finding, can you show a timestamp at every lifecycle stage: discovery, triage, ticket creation, fix deployment, and closure? Summaries are not sufficient.
- **Decision rationale:**
  When a finding was deprioritized, accepted as a risk, or granted an exception, who made that decision, when, and with what justification? If an AI agent downgraded a finding from critical to medium, can you explain why in terms that satisfy a compliance review?
- **SLA adherence:**
  Can you demonstrate that your stated remediation SLAs were tracked and enforced? Can you show breach rates, escalation actions, and trend lines over quarters?
- **Exception governance:**
  For every finding not remediated within SLA, can you show a documented exception with an approver, an expiration date, a risk justification, and a review cadence? Undocumented exceptions are control failures.
- **Ownership accountability:**
  Can you prove that every finding was assigned to a specific team or individual, that the assignment was accurate, and that the responsible party acknowledged and acted on it?

This evidence package is what separates a mature security program from an ad-hoc one. It determines your SOC 2 outcome, your cyber insurance premium, your board's confidence, and your CISO's ability to demonstrate that risk is being managed, not just discussed.

Why AI Agents Cannot Produce This Evidence

An AI agent investigating a finding produces a point-in-time output: a reachability verdict, a fix recommendation, a priority suggestion. That output is valuable. But without a surrounding lifecycle system, it cannot provide:

- **A continuous timeline:**
  The agent ran once. It does not know when the finding was first discovered, how many times it was rescanned, whether it regressed after a fix, or how long it sat in a queue. The full lifecycle must be maintained by a system that persists across every event.
- **Deterministic decision records:**
  When an AI agent changes a finding's priority, the reasoning is probabilistic. Auditors need the original priority, the new priority, the specific evidence that supported the change, and which policy authorized the adjustment. Seemplicity preserves all of this: agent recommendations are recorded as suggested states, original priorities are retained, and the full reasoning chain is stored in the audit trail.
- **Cross-finding consistency:**
  Auditors review patterns, not individual findings. Are similar findings being treated consistently? Is the same CWE triaged as critical in one team and low in another? That analysis requires a system of record tracking every decision across every finding over time, not isolated agent runs.
- **Immutable, timestamped evidence:**
  If an auditor asks in Q4 why a finding was deprioritized in Q1, you need the original reasoning preserved as it was at the time. You can't re-run an LLM and expect the same answer. Models are non-deterministic, and context may have changed. The evidence must be immutable and timestamped at the moment the decision was made.

The Explainability Requirement

As AI increasingly drives security decision-making, explainability is becoming a compliance requirement in its own right. Regulators and internal governance teams are asking for:

- A clear mapping from AI recommendation to the specific data inputs that drove it: which scanner findings, which asset context, which organizational policy.
- Separation between AI suggestion and human or policy approval, so automated decisions can be distinguished from reviewed ones.
- The ability to override, annotate, and document disagreement with AI recommendations without losing the original analysis.
- Retention of the full analysis narrative, including reasoning steps, tools consulted, and confidence level, as a permanent part of the finding's record.

Seemplicity's AI Analysts are built for this. Every agent recommendation flows through a governance layer: recommendations appear as suggested states, administrators configure whether to auto-apply or require human approval, original priorities are preserved alongside agent recommendations, and every decision is tagged with its source analyst, timestamp, and full reasoning chain.

> Security is not just about finding and fixing. It is about proving you found it, proving you fixed it, proving who decided what and when, and proving you can do it again next quarter. That proof is a system, not an agent output.

06 Part Six: How It Actually Works Together

The relationship between AI and Seemplicity is not competitive. It's architectural. Think of it in four layers:

- **The Foundation:**
  Seemplicity's context fabric ingests findings from every scanner, normalizes them into a unified ontology, resolves asset identity, maps organizational ownership, deduplicates by root cause, and applies policy-based prioritization. This reduces hundreds of thousands of raw findings to thousands of actionable items. The fabric acts as a semantic cache, cutting the tokens required for any AI decision by 60–80% and anchoring agent outputs to verified data rather than hallucinated context.

- **The Intelligence:**
  AI agents (whether Seemplicity's own AI Analysts or any frontier model) operate on individual findings that the fabric has identified as high-signal. They investigate with full context: the normalized finding, cross-tool correlated data, organizational ownership, historical remediation patterns, and policy constraints. Because the fabric provides this context upfront, the agent doesn't need to discover it from scratch for every finding.

- **The Outcome:**
  Agent outputs flow back into Seemplicity's remediation engine. Tickets are created with verified routing, SLAs are enforced, exceptions are documented, and the full lifecycle is tracked to resolution. Every decision is auditable, every outcome is measured, and the system learns from each cycle.

- **The Evidence:**
  From ingestion through investigation through resolution, Seemplicity maintains an immutable audit trail. Every timestamp, every decision, every AI recommendation and its supporting reasoning is preserved as compliance-ready evidence. When the auditor arrives, the evidence package is already assembled, not reconstructed from logs after the fact.

> We don't compete with Claude. We are the system that makes Claude useful at enterprise scale. The model is brilliant at reasoning. We provide the data to reason about, the context to reason with, the operational system to act on the conclusions, and the evidence trail to prove it all happened.

The Bottom Line

AI is the most powerful tool security teams have ever had for investigating and remediating individual vulnerabilities. We've built our platform around it. It powers our agents, and it will only get better.

But a remediation program is not a collection of individual investigations. It's a continuously operating system: one that normalizes chaotic data from dozens of sources, maintains a living map of assets and their owners, accumulates organizational context that no API exposes, converts agent intelligence into tracked and auditable actions, and demonstrates to boards and regulators that exposure is being reduced systematically.

That system wasn't prompted into existence. It was built over years, refined through hundreds of deployments, and encoded with operational knowledge that only comes from living inside real customer environments.

AI can analyze a finding. Seemplicity knows which findings matter, who should fix them, whether the fix actually happened, and proves all of it to your auditor. That's the difference between a powerful tool and a production remediation program. The organizations closing that gap are using both.