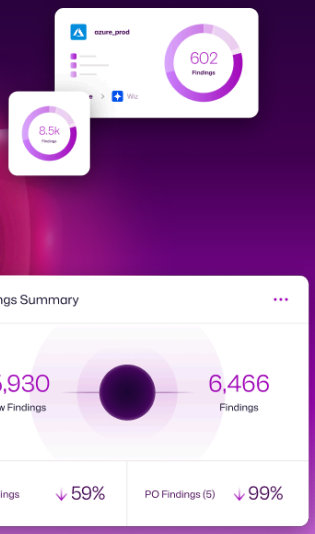
seemplicity

/datasheet

Can I Build an Exposure Management Program with AI?



The honest answer is: partially

In 2026, a security engineer can point a frontier AI model at a SAST finding, give it access to a GitHub repository and a Jira board, and watch it trace a call graph, identify the code owner, draft a fix, and create a ticket. For a single vulnerability, this is genuinely impressive, and Seemplicity uses AI to do exactly this, at scale.

But investigating one finding and running a remediation program are fundamentally different problems. This datasheet explains exactly where AI helps, where it hits a wall, and what it would actually cost to build the rest yourself.

What AI Does Well

Let's give AI its due. The deep investigation of a specific finding, previously the domain of hours of analyst time per vulnerability, is exactly where large language models excel:

- **Tracing reachability:** is the vulnerable code actually in the execution path?
- **Assessing blast radius** and business criticality of the affected asset.
- **Identifying code ownership** via git blame, contributor history, and project structure.
- **Generating fix recommendations:** a code diff, step-by-step instructions, and an effort estimate.
- **Routing a fix-ready ticket** to the right project with the right priority and SLA.

The Data Foundation Problem

Your environment doesn't have one finding. It has millions, arriving continuously from a dozen different scanning tools, each with its own schema, severity scale, and naming conventions. Before any AI agent can investigate a single one, five foundational problems have to be solved.

1. **Data Volume & Deduplication**
   AI alone: An AI agent has no way to know that 55 findings across your scanners are all the same Log4j instance – Seemplicity collapses raw output by 70-80% before any agent runs.
2. **Normalization Across Tools**
   Mapping field names is straightforward; knowing whether a Wiz "misconfiguration" carries the same risk as a Tenable "compliance failure" requires judgment refined across hundreds of real deployments.
3. **Asset Identity Resolution**
   The same server can appear as four different strings across your scanning tools, and no AI agent can resolve those into a single identity without deployment-scale mapping patterns.
4. **Organizational Intelligence**
   An agent on day one has no way to know which team owns a finding, who left the company last month, or which groups have a pattern of reopening tickets as won't-fix.
5. **The Operational Loop**
   Investigation is roughly 20% of remediation – an agent with no persistent memory can't track whether a fix shipped, whether a scanner confirmed closure, or whether an exception was filed.

Proof, Not Output

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

- **Completeness:**
  Every finding from every scanner was ingested and accounted for, with no gaps.
- **Timeliness:**
  Every critical finding has timestamps at each stage, from discovery through validated closure.
- **Decision rationale:**
  Every deprioritization or risk acceptance has a documented owner, justification, and date.
- **SLA adherence:**
  Remediation SLAs were tracked and enforced, with breach rates and escalation actions on record.
- **Exception governance:**
  Every exception has an approver, expiration date, and risk justification.

AI agent outputs are point-in-time. Auditors need a continuous, immutable record of every decision from detection through resolution – and you can't re-run an LLM six months later and expect the same answer.

The Real Build Cost

If the data foundation is the challenge, and AI agents are increasingly good at writing code, could a team simply instruct a coding agent to build the equivalent of Seemplicity's platform from scratch? Here's what that would look like, layer by layer.

Layer	Can AI Build v1?	Production Reality	Core Challenge
Connectors (170+)	Yes, happy path in an afternoon	12–18 months of hardening per connector	Edge cases, rate limits, schema drift
Normalization	Naive field mapping, yes	Years of customer feedback to get to production quality	Judgment calls vary by tool and customer
Asset Identity	Simple exact-match cases only	Fuzzy matching requires deployment-scale learning	No common key across tools
Org Intelligence	Cannot be generated	Requires months operating inside your environment	Data doesn't exist in any API
Governance & Audit	Structure, yes; trust, no	Auditors need deterministic trails, not agent logs	Regulatory compliance requirements