protegrity

WHITE PAPER • SEPTEMBER 2026

The AI friction tax, and what removes it

Why AI ships late and diminished, and why the security review can't fix it alone.

Written for the architect and the reviewer who have to be satisfied on mechanism, not only on outcome.

RESEARCH

Enterprise Management Associates. *The State of AI Friction*. July 2026.

Commissioned by Protegrity, conducted independently.

SAMPLE

150+ IT and security leaders surveyed April 2026, plus 16 practitioner interviews.

READ TIME

25 minutes

Executive summary

A CISO in payroll and HR services put it plainly. Turn on an AI assistant across the company's shared drives, and "that spreadsheet somebody in HR forgot to set the permissions on is now an answer to a question."

AI in the enterprise is not stalling for lack of money or appetite. In April 2026, Enterprise Management Associates surveyed 150+ IT and security leaders for research Protegrity commissioned and EMA conducted independently. Of those, nearly 83% reported production delays tied to security or compliance review, and two-thirds of the delayed group waited a month or more. Even when AI reached production, 82% of those surveyed said they shipped it in some diminished form. Over the same period, 88% of those surveyed were raising agentic AI budgets by 10% or more.

FIGURE 1 · BUDGETS ARE RISING. DELIVERY IS NOT.

Investment is accelerating while delivery stalls. Source: EMA, The State of AI Friction, July 2026, 150+ respondents, commissioned by Protegrity.

Read those together and the cost lands on one desk. The team that owns the AI project has the budget, the model and a pilot that works, and a quarter of the year goes to a review whose outcome they cannot influence. The money was committed on a full-capability business case, and what ships is smaller.

Those four numbers describe a review process handed a question it was never built to answer. A permission model can say whether a person should see a record. It cannot say whether a system that reaches every shared drive in the company and reasons across them should be switched on, because each permission in the chain is defensible and the combination is the problem. A careful reviewer says no, or approves a narrower project. That is the friction tax, and because it is architectural, procedural fixes will not remove it.

This paper argues that the tax comes off when protection lives in the data value itself rather than in the perimeter around it, and that the review can then say yes because the workflow hands it evidence of which policy applied, to which values, for whom. It works one query through end to end, maps the answer to the five sources of friction EMA ranked, sets out where to start and in what order, and names what would have to be true for the argument to be wrong.

It is written for enterprise leaders deciding how to move AI into production without widening their risk assessment. Its subject is the data-security share of the problem, which is our area and not the whole of it.

Introduction

The full version is worth reading, because the scale in it is what makes this a security problem rather than an anecdote.

> *"If we wanted to turn on Microsoft Copilot and point it at all the OneDrives and SharePoint instances we have across the organization, it would probably aggregate existing data security problems. There's probably a thousand places where people have dropped things into OneDrive or SharePoint and shared it with the entire company, but nobody knew it existed before. And now suddenly Copilot's training on it, and somebody's asking a question about someone else's salary. And that spreadsheet somebody in HR forgot to set the permissions on is now an answer to a question."*
>
> CISO, payroll and HR services. One of 16 practitioner interviews Protegrity commissioned and an independent firm conducted.

Nothing in that story is a breach. No control fails and nobody gets in who shouldn't. A spreadsheet that had been over-shared for years becomes reachable by a system whose job is to find things and put them together, and a permission mistake from 2024 turns into a fact about a colleague's pay.

This is why AI projects that work in a pilot then sit for a quarter waiting for a signature. The reviewer is not being asked whether the assistant is safe. They are being asked to vouch for every permission decision anyone in the company has made since the tenant was created, and no reviewer can do that.

To understand how widespread the problem is, we commissioned Enterprise Management Associates. In April 2026, EMA VP of Research Christopher M. Steffen surveyed 150+ IT and security leaders about their AI projects and published the findings as *The State of AI Friction*, available at protegrity.com/resources/white-papers/the-state-of-ai-friction. Protegrity paid for the research. EMA conducted it independently and the conclusions are its own.

What the friction tax costs

EMA's term for the problem is the AI friction tax. They describe it as "the accumulated drag imposed by security reviews, compliance processes, sensitive data access challenges, autonomy constraints, and trust concerns." The tax, they say, "is a cost enterprises pay not in a single line item but across three, time lost to delay, capability lost to restricted deployment, and budget lost to compliance overhead."

Their conclusion is worth quoting in full.

> *"Enterprise AI is no longer constrained by appetite, budget, or technical readiness. It is constrained by friction between AI systems and the governance infrastructure meant to secure them. Nearly 83% of organizations report production delays tied to security or compliance review, and 66.7% of those delays stretch a month or longer. Even when AI does reach production, 81.6% of organizations report deploying it in some diminished form."*
>
> Christopher M. Steffen, EMA, *The State of AI Friction*, July 2026, n=152, research commissioned by Protegrity

A month is the unit in which an AI project loses its executive sponsor, and two-thirds of the delayed group waited at least that long. Nearly 30% of them waited four months or more. Over the same period, 88% of those surveyed were increasing agentic AI budgets by 10% or more. Money committed and capability delivered are moving in opposite directions, and if the constraint were budget or enthusiasm those two facts could not coexist.

Security review is one source of friction among several. Specialist hiring, legacy integration, legal questions and worries about model accuracy all slow projects too. This paper takes the data-security share because that is what we know, and it is a share rather than the whole.

Why agents break the old model

An AI lead has a pilot that works. It answers the questions it was built to answer, the business sponsor has seen it, and it has been waiting eleven weeks for a signature. Nothing is wrong with the model. What is unresolved is a question about the credentials it runs under, and it is worth understanding why that question is hard, because it is the one holding the project.

Picture a database administrator who has held the same credentials for eight years. She is an illustration, as the situations throughout this paper are, built to be recognizable rather than drawn from any one customer. The two practitioner quotations are the only reported speech. She knows, without thinking about it, which queries she should never run against production during business hours, which tables contain things that would end a career if they left the building, and which joins are technically permitted and practically forbidden. None of that is written down. It lives in her judgment.

Now give an agent her credentials. It has every permission she has and none of the judgment. It will run the join she would never run, not out of malice, but because the join answers the question it was asked and nothing in a model weighs an unwritten rule the way a person does. That is the first thing agents break. Permission was always doing two jobs, granting access and relying on a human to exercise restraint, and an agent only inherits the first one.

The second is speed. A perimeter model assumes the environment holds still long enough to check it, and an agent gathers, joins and reasons in milliseconds.

The third is that an agent chooses its own route as it works, so nobody knows in advance what it will touch, and the audit record has to be reconstructed from what happened rather than checked against what was approved.

Asked which challenges concern them most about deploying autonomous agents, 63% of those surveyed cited data leakage through agent actions, the top answer. Regulatory compliance followed at 49%, prompt injection leading to unsafe execution at 44%, tool manipulation and privilege escalation at 40%, and agent impersonation or spoofing at 32%. Shares are of respondents and the question allowed more than one selection.

FIGURE 2 · WHERE AGENT CONCERNS PHYSICALLY OCCUR

Leakage and injection happen in the prompt path. Escalation and impersonation happen at the tool call. Compliance sits under both.

Set that list against where each problem physically occurs and it separates into three. Data leakage and prompt injection happen in the path between the application and the model. Tool manipulation, escalation and impersonation happen at the moment an agent calls a tool. Compliance is the evidence layer beneath both. Three different places, three different controls, and a review that treats them as one undifferentiated list of worries has no way to resolve any of them.

The gap is already live. Only one in five of those surveyed reported high autonomy, meaning agents that plan and run multi-step work on their own. Add moderate autonomy and 69% are already running agents that write to databases and call external APIs. The agents have their credentials. The governance for what those credentials can do together does not exist yet.

Three things that are new

Inferred leakage

A clinical research organization de-identifies a trial dataset before sharing it with a partner. Names removed, dates shifted, identifiers replaced. The technique was sound when it was chosen, and it passed review.

A language model reads the same dataset differently. It notices that a participant's visit dates, the dosage schedule and the free-text notes describing a rare side effect narrow the field to one person in one region, then confirms the guess against a public conference abstract. Nothing sensitive was in any single field. The sensitive fact was reconstructed from fields that each looked harmless.

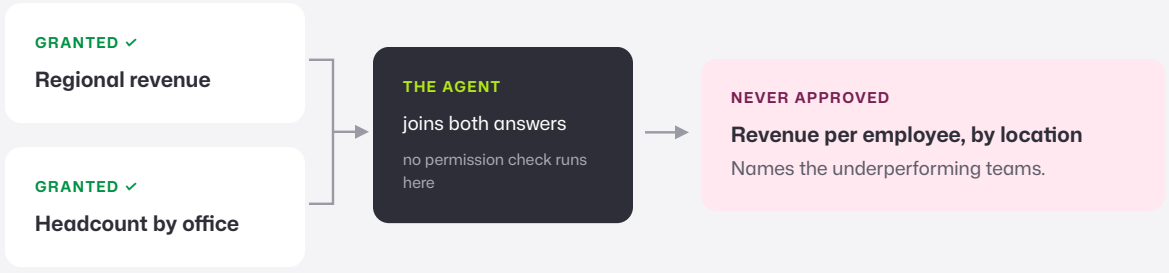
That is inferred leakage, and it is the exposure that should worry a reviewer more than direct access does. A CISO in clinical trials and research services told us re-identification is now a major concern in their environment, precisely because large language models make de-identified content easier to infer back to a person than the techniques producing it had assumed. De-identification is a one-time transformation whose sufficiency was judged against the reconstruction methods of its day, and the methods have moved. Any control whose adequacy was last assessed before 2023 deserves a second look.

The chain

Access control answers one question at a time. An analyst is entitled to see regional revenue. The same analyst is entitled to see headcount by office. Neither grant is wrong. Ask both questions of an agent in the same session and it will hand back revenue per employee by location, which is a number that names underperforming teams and was never on anyone's approved list.

Every link in that chain is legitimate. The exposure is in the join, and the join happens inside the agent, where no permission model can see it. Models also carry information forward, so something supplied in one question or in training can surface combined with something unrelated later.

FIGURE 3 · THE EXPOSURE IS IN THE JOIN



Access control checks each request on its own. The exposure is produced by the combination, and the combination happens inside the agent.

Knowledge, not records

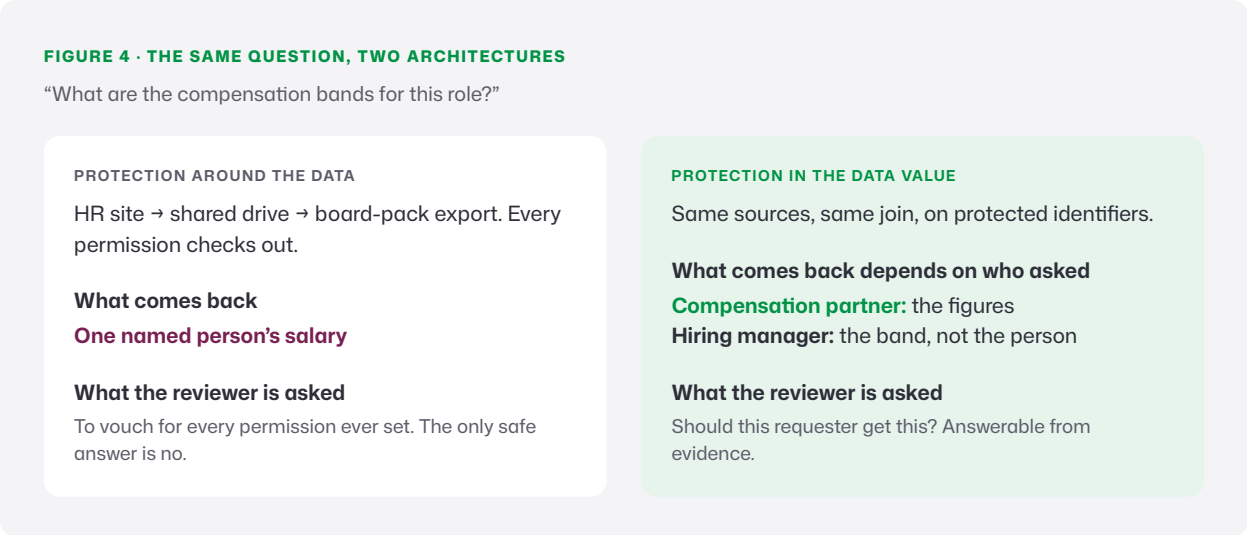
Ask a manufacturer what they would least want a competitor to have and the answer is rarely a table. It is the pricing logic, which lives in a decade of quotes, discount approvals and margin exceptions scattered across three systems. It is which supplier terms flexed under pressure, and which design decisions were made for reasons nobody wrote down.

None of that is a record. It is the relationship between records, which is exactly what an agent is good at assembling. Field-level protection guards individual cells. The exposure emerges when the cells are read together, so protection has to survive the join, and that is the level at which this paper's argument operates.

That is also why a risk assessment stretches from weeks into months. The reviewer is being asked to approve an agent's ability to synthesize across sources, and there is no checkbox for that.

One query, twice

The three ideas above are easier to see in a single request than to argue in the abstract, so here is one, run twice under different assumptions.



Every permission is legitimate in both cases. What changes is what resolves, and for whom.

Go back to the forgotten spreadsheet. Someone asks the assistant a reasonable question about compensation bands for a role they're hiring. The assistant reaches the HR site, then a shared drive, then an export somebody made for a board pack two years ago. No control is broken anywhere. Every permission checks out, because every one of them was granted by a real person for a real reason at some point. What comes back is an individual's salary, assembled from three places that each looked harmless.

This is what the risk assessment is actually about, and it's why the assessment takes three months instead of three days. The reviewer isn't deciding whether this person should see compensation data. They're deciding whether a system that can reach every shared drive in the company and reason across

them should be switched on at all. An access model has no answer to that, because every individual permission in the chain is defensible and the combination is the problem. So the safe answer is no, or yes to something much smaller.

Now run the same question where protection is applied to the values themselves. The join still happens, because a protected employee identifier is deterministic and matches across the three sources, so the assistant can still tell these records describe one person. The salary figures aggregate too. The query runs through a governed path that resolves values against policy as it executes, rather than handing them over first, so a real compensation band comes back and what resolves inside it depends on who asked. A compensation partner is entitled to the values and sees the figures. The hiring manager is entitled to the band and sees the band, without the individuals inside it. Neither of them had to know which of the three sources was the sensitive one, and neither did the person who set up the query.

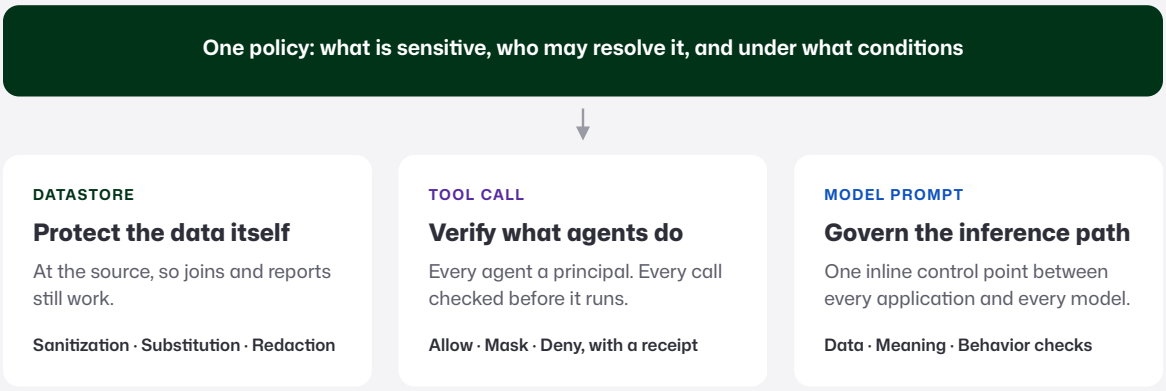
Every step the assistant took is logged in a form somebody can read back to an auditor. That matters because the forgotten spreadsheet is still sitting there. Protection at the value level doesn't fix the permission somebody set wrong in 2024. It stops that mistake from becoming an answer to a question.

The reviewer's question changes shape. Instead of asking whether this system can be trusted with every shared drive in the company, they ask whether this action should go ahead given what this requester is entitled to see. That is narrower, it is answerable from evidence, and it gets answered once instead of once per project.

That change of question is the argument of this paper. The review is currently being asked something no control model can answer, and a different architecture lets it be asked something answerable instead. Everything that follows is what has to be true for that to happen.

What removes it

FIGURE 5 · DEFINED ONCE, ENFORCED IN THREE PLACES



Sensitivity is defined once. The same definition governs the database, the agent's action and the model prompt.

The friction tax comes off when protection stops guarding the place data sits and lives in the data value itself. Sensitivity is defined once, in policy. That definition is enforced in the three places an AI workflow touches sensitive data, the datastore, the agent's tool call and the model prompt. And every enforcement decision leaves a record a reviewer can read back. That is the whole answer. The rest of this paper is how each part works and what it changes for the reader.

Four things have to be true for it to hold, and only one is organizational.

1. Every AI workflow gets the full data and knowledge it needs to finish its task.
2. One centrally managed policy governs security and governance, with enforcement verified at every step.
3. Sensitive values are protected at every step, including inside the model's reasoning, so there is no point at which they sit in the clear.
4. One team owns production AI, staffed from data, security and governance.

The third is what makes the first two possible, because a workflow can only be given everything it needs once nothing in it is exposed. The third is also what shrinks the fourth. Most of what those functions argue over is who carries the risk of an exposure that, under this model, does not occur.

All four line up with what EMA recommends, which is to audit where security review sits in the AI delivery pipeline, treat compliance operating cost as a board-level metric, match autonomy ambition to governance capacity before scaling agent budgets, and evaluate data protection methods that travel with the data.

This architecture predates the survey, and the survey independently arrives at it. Asked how they want data-security controls deployed, those surveyed split almost evenly. Half chose a security-team-owned model, a central platform (29%) or appliance clusters (21%). Another 39% wanted protection embedded at the developer level, as an SDK or container (23%) or as modules inside developer workflows (16%). The rest would outsource it.

EMA reads that as one requirement rather than two camps, because the halves answer different questions. The first is about policy, meaning who sets and audits the rules. The second is about enforcement, meaning where protection runs. In their words, "these preferences resolve to a single architectural requirement, central policy management with distributed enforcement. Organizations want one place to set and audit policy, and they want that policy enforced at the point of data use, inside the pipeline, inside the workflow, inside the agent, rather than applied as a gate before deployment begins."

That is the model this paper describes, stated by the analyst before we said it. Asked about a solution that delivers AI-ready data without roadblocks by embedding protection inside the workflow itself, 92% of those surveyed called it extremely or very valuable, 41% extremely and 51% very. At that level the demand is close to universal among the people asked.

Enterprise security was built to keep people away from sensitive data. Guard the store, check the role, allow or deny. Enforcement had to be re-applied everywhere data was accessed, and the only available tool was blocking. Analytics stalled on that for years, and AI makes it worse, because data now flows into pipelines, prompts and model context continuously and blocking that flow defeats the point of the project.

Protecting the value itself, and letting that protection travel, is the inversion of that model. It is enforced in three places.

Protecting the data itself

This pillar settles one question. Can a model reason over data without anyone being exposed to it? The method that makes the answer yes is semantic encryption, which means protecting the value inside the data while leaving its meaning and shape intact, so a protected date is still a valid date and a protected account number still matches itself across four systems, which is what lets joins hold and lets a governed query order and aggregate without exposing the values. Three techniques apply, chosen by what the data is for.

Remove the identity permanently (sanitization). When the original values are never needed again, the data is irreversibly de-identified or replaced with a synthetic set of the same statistical shape and no real values. Strongest privacy posture, right for training data and shared research sets.

Replace the value but keep it usable (substitution). When the data has to stay both usable and recoverable, vaultless tokenization and format-preserving encryption swap each sensitive value for an equivalent of the same type, format and length. An analyst can join on it and a model can reason over it. The real value comes back only for a principal whose policy allows it.

Control what is shown at the moment of access (redaction). When the stored value should not change but not everyone should see it, masking returns an obscured version and row filtering returns only what that principal is entitled to, so the same query from two people returns two results.

FIGURE 6 · WHAT STILL WORKS ON PROTECTED DATA

THE VALUE	PROTECTED FORM	ON THE PROTECTED VALUE	THROUGH THE GOVERNED QUERY
Account identifier 1047-2231-8890	`7F2A-91QK-3310` same token every time	**Joins and matches**	nothing further needed
Date 2024-03-17	`2031-11-04` still a valid date	**Parses and matches**	**Sort, range, bucket**
Salary 85,000	`31,742` same type and length	**Stored and moved**	**Sort, sum, average**
Free-text note “rash after dose 3”	“●●●● `after dose 3”` sensitive spans protected	**Read and reasoned over**	**Resolves for entitled readers**

Values illustrative.

Some operations run on the protected value itself. The rest run through a governed query that resolves against policy as it executes.

What that buys, in the order a reviewer tends to ask about it.

- **Protected data stays usable.** Type, format, length and the relationships between tables survive, so protected data can still be queried, joined, analyzed and reasoned over.
- **Define once, enforce everywhere.** Sensitivity is defined a single time and enforced wherever the data lives or moves.
- **Existing tools keep working.** Because protected values keep their type and format, queries, joins and reports behave as they did. The less a method changes the shape of the data, the less has to change around it.
- **A breach loses its payload.** When protection is applied at the source, what an attacker reaches is a format-preserving token with no exploitable value.

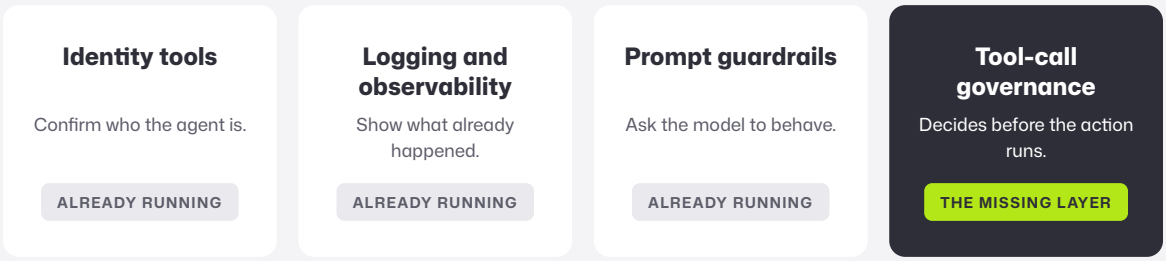
There is a real cost here, and it belongs in the open. Applying protection at the source means classification has to be right, and classification on a large, old estate is work. The organizations that get the most out of this treat discovery as a standing practice rather than a project.

Verifying what agents do

Sixty-nine percent of those surveyed are already running agents that write to databases and call external APIs. Go back to the administrator's credentials. The agent holding them has no identity of its own and no role of its own. It acts on behalf of a user, so any decision about it has to reflect both the agent and the person behind it. And it cannot be handled with a plain allow or deny, because an agent exists to do something useful and blocking it defeats the purpose of having one.

The fix is to govern the tool-call boundary, the exact moment an agent tries to act. Every agent is a registered principal with a verifiable identity. Every tool call is checked against policy before it executes. Field-level protection is applied to whatever comes back, so sensitive values stay protected regardless of how the model phrased the request. Every decision leaves a cryptographic receipt, which turns "what was this agent stopped from doing" into a query rather than a forensic exercise.

FIGURE 7 · ONLY ONE OF THESE CAN STOP AN ACTION



Categories, not vendors. Most teams already run the first three and still cannot answer what an agent was prevented from doing.

What that buys, once an agent is a principal rather than an anonymous caller.

- **Agents run at full capability.** When a team cannot prove agents are safe, they cut permissions and autonomy as a hedge and the project ships smaller than it was designed. When every tool call is identity-checked, policy-enforced and logged, the hedge is unnecessary.
- **No raw credentials sitting in agents.** The control plane holds every connector credential. Agents call a governed tool and never hold a database password, an API token or a connection string.
- **Enforcement a model cannot talk its way around.** Policy is compiled into the query and protection applied to results server-side, so the guarantee does not depend on the model behaving.
- **One endpoint for any framework.** Any MCP-compatible agent connects to a single endpoint, and every tool reachable through it already has policy enforcement and logging built in. Teams do not write a security wrapper per integration.
- **Audit evidence that is ready rather than reconstructed.** Every decision is a structured, tamper-evident record, exportable for SOC 2, EU AI Act, ISO 27001 and NIST AI RMF.

Governing the inference path

An account manager pastes a customer's full support history into a prompt to draft a renewal email. The history contains a card number and a home address. The model does its job well, and now the card number is in the prompt, in the log, and in front of whoever reviews the log.

Prompts and responses became a place sensitive data is exposed, and controls built for data at rest never reached it. The common response, a scanner bolted onto each application, gives inconsistent coverage that no one can audit as a whole.

The alternative is one inline control point between the application and the model, returning a single decision per call, allow, block, log or protect, with an audit record for every request. Per-check timeouts and circuit breakers keep a slow check from taking down the pipeline, streaming and standard responses are both handled, and third-party guardrail engines can run alongside our own, so there is one control point rather than a safety stack per application.

Four kinds of check run there.

- **Data checks** find and protect sensitive values in the inference path. On each request the scanner validates the requesting principal, identifies sensitive data, and rebuilds the content with protected values in place. The account manager's card number becomes a token before the model ever sees it, and only someone with the right entitlement can turn it back.
- **Meaning checks** work on the content of the request, stopping unsafe or off-topic queries and catching leakage in real time.
- **Behavior checks** constrain what the model and the workflow are permitted to do, which is the defense against prompt injection leading to unsafe execution. Meaning checks judge content. Behavior checks constrain action.
- **Governed natural-language analytics** turns a plain-language question into a structured query against approved sources, applies protection to the results by policy, and resolves real values only for those entitled to them. The model reasons over metadata and relationships rather than raw values, so it gets query context without the data.

What that buys at the inference boundary.

- **Sensitive data is protected before it reaches the model**, in flight, rather than left to prompt discipline or a per-application scanner. One inline gateway covers traffic across every application and provider.
- **Clearance decides exposure.** Authorized users see real values. Everyone else sees protected ones, including for data they submitted themselves.
- **Natural-language analytics under the same policy.** A plain-language question resolves to what the asker is entitled to see.
- **Audit-ready by default.** Every request produces a record of what was checked, what each check decided, and why.

The five sources of friction, and which part answers each

EMA ranked the five largest contributors to the friction tax by impact. Here they are in EMA's order, with what changes for each when protection travels with the data.

Source of friction, ranked by EMA	What changes	What the reader gets
Security review. "AI systems cannot move forward until they clear a process that was designed for a different era of software deployment."	Data is protected before it reaches a model, and that protection stays with the field for the life of the project, from ingestion to result.	The risk assessment is about an action, not about trusting an agent with a system, so it has something it can actually evaluate.
Compliance and audit. "Governance and audit processes designed for static systems are being applied to dynamic AI deployments, creating manual overhead that slows approvals and increases operational cost at scale."	Because protection travels with the data, each step leaves a record of which policy applied, which values resolved, and for whom.	You hand your auditors evidence instead of an assurance. One policy across every datastore and pipeline, with an approval trail nobody has to reconstruct.
Sensitive data access. "AI systems require high-value enterprise data to deliver meaningful outputs. When that data is difficult to secure and approve for use, organizations substitute lower-quality inputs and get lower-quality AI."	Analysts and AI teams work with the data the question actually needs, because protecting it no longer means withholding it.	Mobility and safety stop being a trade-off. Protected data behaves like unprotected data in a query, so access stops being the thing that gets negotiated.
Autonomy. "The gap between agentic AI ambition and the governance infrastructure required to authorize and monitor autonomous action keeps agents narrower than designed."	Two things change. Sensitive values aren't present in the clear for an agent to reach, and each request is judged against what that agent has already asked for.	Agents run at the capability they were designed for instead of the capability that felt safe.
Trust and risk. "Data leakage, prompt injection, privilege escalation, and agent impersonation are active deployment constraints that force security teams to restrict what agents can do before they go live."	Protection is applied before the project starts and enforced through to the end, so a leak has nothing readable to carry.	Nobody has to throttle the project back as a hedge.

Source for all five, and for the ranking, is EMA, *The State of AI Friction*, July 2026, 150+ respondents, commissioned by Protegrity.

Which part of the architecture answers which source of friction

Each of EMA's five sources is answered by a specific part of the architecture rather than by the architecture in general. Three enforcement pillars sit on one shared governance foundation, and the mapping is close enough to one to one that it is worth setting out directly.

FIGURE 8 · ONE ANSWER PER SOURCE OF FRICTION

	RANKED BY EMA	WHAT ANSWERS IT
1	Security review	Discovery and Policy, Data Protection
2	Compliance and audit	Discovery and Policy
3	Sensitive data access	Data Protection
4	Autonomy	Agent Trust
5	Trust and risk	Governed Inference, Agent Trust

Discovery and Policy is the foundation. It answers the top two sources, roughly two-fifths of the friction.

EMA ranked the five. Each is answered by a specific part of the architecture, and the foundation carries the top two.

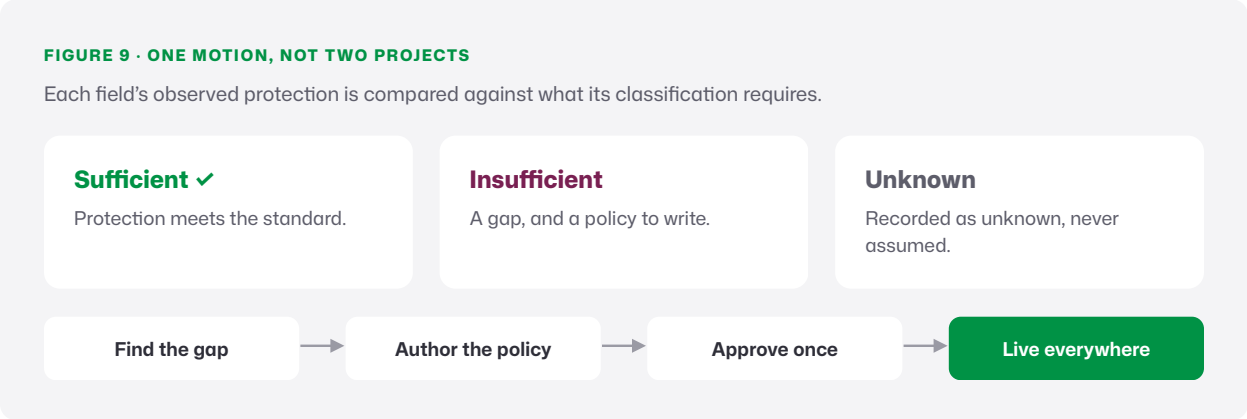
Two things follow. The foundation carries EMA's first and second ranked sources, since Discovery and Policy answers compliance and audit outright and is half the answer to security review. Knowing what you have and being able to prove it turns out to be two-fifths of what slows an AI project, which is a different weight than "preparation" implies. And the three enforcement points are not alternatives to one another. They share one policy definition, which is the property that makes the mapping hold, and where these enforcement points are separate products the mapping breaks at the seams.

The rest of this paper describes how each of those is delivered.

The foundation, knowing your data and setting policy

An auditor asks a simple question. Is the customer date-of-birth column in the claims warehouse protected right now, to the standard its classification requires? In most organizations the answer takes three weeks and arrives as a slide deck. One person screenshots a masking rule, another confirms encryption at rest, a third finds a service account nobody has reviewed since 2023, and the answer is a guess in a suit.

Discovery replaces the guess with a record. It does not assume a native control is sufficient just because it exists. It records what is actually in place and compares that against what the data's classification requires, so "protected" becomes a status with evidence behind it. Every gap it finds is a policy waiting to be written, and writing it is the same motion rather than a second project with its own toolchain and its own audit trail.



"Protected" stops being a claim and becomes a status with evidence behind it.

Knowing your data's true condition

Discovery has to survive an auditor's follow-up question, which means recording what is actually in place rather than what the architecture diagram says should be. Six things make the record hold up.

- **Continuous inventory** across datastores, lakehouses, streaming platforms and schema registries, covering not just content but the protections, actors, transformations and policies already in place.
- **Classification with its working shown.** ML models by default, with transparent fallback to NLP, regex or dictionary methods where coverage is incomplete. Every finding carries its classifier type and confidence score.
- **Actual condition recorded as fact.** Whether a value is clear, masked, tokenized or encrypted is recorded as observed, never assumed, and compared against what its classification requires, so each item is flagged sufficient, insufficient or unknown.

- **Relationships held explicitly.** Every data object, classification, protection state, actor, access path and transformation is a node with stated relationships, column to view to export to user, so a conflict is an explained attribute rather than a line in a static report.
- **Human approval at scale.** High-confidence findings are proposed automatically, and nothing reaches a downstream system without a person approving it.
- **Remediation you run.** For any datastore it scans, discovery generates the command that closes the gap and presents it for a human to review and execute. It does not execute changes itself.

Setting policy once

Authoring is where most governance tooling loses the people who are supposed to use it, usually by asking a compliance lead to write code. The design goal here is that the person who owns the policy can author it without an engineer.

- **Two guided authoring paths**, one for encryption, masking and attribute-based filtering of data at rest and in queries, one for protecting classifications inside AI workflows. One page each.
- **Enforcement points derived automatically**, so the author never learns an enforcement-point taxonomy or writes a line of policy code.
- **Review catches two kinds of conflict**, a method that does not match best practice, and a new policy contradicting an approved one for the same principal and classification. Each is acknowledged individually, and the written rationale required before approval becomes a permanent part of the record.
- **One searchable system of record** showing every policy's source, classifications, principals, methods, frameworks and outstanding exceptions.

Where to start, and in what order

FIGURE 10 · SIX STEPS, IN ORDER

	THE STEP	WHAT IT PRODUCES FOR THE NEXT ONE
01	**Find out what you actually have**	A list of gaps, with evidence
02	**Write the policy once**	One definition of what is sensitive
03	**Protect the data at the source**	Joins and applications keep working
04	**Extend the policy to agents**	A receipt for every decision
05	**Extend it to the inference path**	No values left in the clear
06	**Hand the reviewer the record**	**A question the review can answer**

Run the project currently stuck in review through steps 1 to 3 first.

Each step produces the thing the next one needs. Start with one workflow, not the estate.

The architecture above is not adopted all at once, and it should not be. The order matters because each step produces the thing the next one needs.

1. **Find out what you actually have.** Inventory the datastores an AI workflow will touch, classify what is in them, and record the protection state of each field as observed rather than assumed. “Protected” becomes a status with evidence behind it, and the gaps become a list.
2. **Write the policy once.** Turn each gap into a rule in one place, check it against every rule already active, and approve it with a written rationale. There is now a single definition of what is sensitive and who may resolve it, and every later step enforces that definition.
3. **Protect the data at the source.** Apply format-preserving protection to the classified fields where they live. Joins still hold, applications still run, and anything that needs order resolves under the policy from step two.
4. **Extend the same policy to agents.** Register each agent as a principal, route its tool calls through a point that checks them against the same policy, and apply protection to what comes back. Agents run at the capability they were designed for, and every decision has a receipt.
5. **Extend it to the inference path.** Put one control point between applications and models so prompts and responses are protected in flight. The last place sensitive values could sit in the clear is closed.
6. **Hand the reviewer the record.** It is a by-product of the five steps above, not a sixth project. The review is now answering a question it can answer.

Start with one workflow, not the estate.

FIGURE 11 · WHAT EACH FUNCTION THAT HAS TO SIGN GETS

SECURITY

Evidence of which policy applied, to which values, for whom.

DATA AND AI

The data the question needs, and projects that keep their original goals.

COMPLIANCE AND AUDIT

Policy set once, enforced wherever data moves, provable to auditors.

THE BUSINESS

AI that lands on time and does what was promised, with less exposure to a costly leak.

The CISO holds final approval at 47% of organizations surveyed, 83% with the CIO and CTO.

Removing the friction tax resolves a different problem for each function that has to sign.

What would have to be true for this to be wrong

An argument nobody has tried to break is an argument that has not been tested, so here is where this one is vulnerable.

If classification is wrong, everything downstream is wrong. Protection that travels with the data depends on knowing which data is sensitive. On a large, old estate that is genuinely hard, and a misclassified column is protected data nobody knows is unprotected. Continuous discovery reduces the problem and does not eliminate it. An organization that cannot commit to discovery as an ongoing practice will get less from this than this paper implies.

The guarantee follows the path, not the data alone. Joins and matching work on the protected value directly, because the same input always protects to the same token. Ordering and aggregation work too, through a governed query path that resolves against policy as the query executes, and sensitive values inside free text are found and protected inline rather than left readable. All of that depends on the work going through those paths. A workload that reaches the data around them, on a platform not covered or through a connection nobody registered, gets the storage-level protection and not the rest. The architecture is only as good as its coverage, and coverage is a deployment question rather than a property of the method.

A review culture is not only a tooling problem. We argue the review is stuck for want of evidence. Some reviews are slow because the organization is risk-averse, under-resourced, or has been burned before. Better evidence helps, and it does not automatically produce a faster decision. If your security team's constraint is headcount rather than information, this removes a reason for delay without removing the delay.

The EMA evidence is commissioned. Protegrity paid for the study. EMA conducted it independently, the sample is real at 150+ respondents, and the sponsorship is disclosed here and on the source's own cover. A reader is entitled to weigh it accordingly, and we would rather say so than have them discover it.

The alternative might be good enough. For an organization running one narrow AI use case on a single governed platform, native controls may well be sufficient. The argument in this paper is about what happens at the third and fourth use case, when policy has to hold across systems that do not share a control plane.

FIGURE 12 · CISO, PAYROLL AND HR SERVICES

"That spreadsheet somebody in HR forgot to set the permissions on is now an answer to a question."

The exposure was already there. What the assistant changed is that something is now looking for it.

Conclusion

Budgets are rising, appetite is not the constraint, and the projects are stalling anyway. Eighty-eight percent of those surveyed are raising agentic budgets while 82% ship diminished AI, and nothing in that gap closes on its own.

The pilot already works. What stops it is a review being asked to approve something it has no tools to assess. Not whether an agent should be trusted with sensitive data, but whether a workflow in which sensitive data was never exposed needs that approval at all.

The workflow can answer that question because it hands the reviewer evidence instead of assurance. Which policy applied, to which values, for whom, at every step. A reviewer who can read that back does not have to take anyone's word for it, and a decision made on evidence is a decision that holds the second time it is asked.

That is a question a security team can answer once, and answer yes.

What follows that yes belongs to the team that has been waiting for it. The pilot goes to production at the capability it was designed for, on the data the question actually needs, and the next project starts from an approval that already exists rather than from the beginning of the queue.

Until it is asked, the tax keeps accruing, and it accrues on the projects with the most to lose. Every quarter this stays unresolved is another quarter of agents running below the capability they were funded for, another review measured in months rather than days, and another budget increase spent on

capability that ships diminished. And unlike most delays, this one does not reverse cleanly. A model trained on data it should not have seen cannot be untrained, and an inference that has already left the building cannot be recalled.

NEXT STEP

To map the friction in your own AI pipeline, talk to us.

See how organizations like yours are moving AI into production on the data the question actually needs – protegrity.com


protegrity