sonet.io

White Paper

Zero Trust Still Leaks Trust

Why Agentic Automation Needs Secure Rendering and Evidence-Grade Controls

Zero Trust has meaningfully improved how enterprises decide who can start a session. MFA, conditional access, and device posture checks are now common. But the most consequential "trust leak" often happens **after** the session begins, when sensitive data renders in a browser or remote UI and teams try to claw back control using endpoint agents, extensions, or reactive monitoring.

That gap matters more in the agentic era. Many AI agents work inside the same web and Windows applications employees use, where permissions were designed for people and roles, not precise, step-by-step control. When automation runs inside these sessions, the governance question is no longer "can the agent log in?" but "what is it allowed to see and send, and can we prove it?"

This paper introduces a practical model for secure rendering and bidirectional session governance: applying policy before content renders and controlling what inputs (uploads, pastes, tool actions) can flow back to systems of record. We pair this model with evidence-grade auditability, integrity-protected audit records, and chain-of-custody that hold up under regulatory and adversarial scrutiny, so CISOs can move early agent deployments from "interesting" to "approvable."

Most initiatives don't fail on capability; they fail on approval. In the agentic era, Zero Trust must evolve from "secure access at the door" to secure behavior inside the session, where rendering and interaction are the real control points. The next model is content-level authorization + bidirectional enforcement (what can be shown, and what can be sent back) paired with evidence-grade audit trails that produce durable, non-repudiable proof of agent actions.

The shift: agents are becoming users

For most organizations, "enterprise AI" started as copilots and isolated proofs of concept. The next wave is different: agents that execute multi-step workflows, operate across multiple applications, and take actions with real operational impact.

A key reality is often overlooked: Agents don't just call APIs. Many agents operate through the same interfaces humans use.

This is especially true in operationally complex enterprises where critical workflows still live in a mix of SaaS, legacy web, and Windows applications that were never designed for automation.

Implication for CISOs: the control plane for agent risk isn't only API gateways and service-to-service auth. It's increasingly the session boundary, the place where UI content renders and where inputs and outputs cross trust zones.

The governance constraint: why approval is the bottleneck

Most enterprises already feel the pattern: automation teams can build something that works, but it stalls before production because the security "yes" requirements aren't met.

Three "approval blockers" show up repeatedly in agentic programs:

1. **Unmanaged execution risk ("agents in the wild")**
   When automation runs on endpoints, in unmanaged browsers, or inside toolchains that security can't reliably constrain, the blast radius is unclear and the compensating controls become a patchwork.
2. **Policy bypass inside sessions**
   Even with strong identity and access at login, agents can still reveal or transmit data the enterprise didn't intend to expose once inside the app.
3. **Evidence gaps**
   When a regulator, auditor, or incident response team asks "what happened?", basic logs are rarely enough. If you can't produce durable evidence with integrity guarantees, you don't have governance.

Where Zero Trust still leaks trust

The authorization gap inside the session

Many teams have done the basics: shrinking VPN sprawl, enforcing MFA, adding conditional access, turning on some session logging. And yet, the lived reality persists: "If you got in, you can probably see it."

Why? Because a lot of "least privilege" is still least privilege at the door, identity, posture, timeouts, role assignment, necessary, but not sufficient.

A session is not a permission model. It's a container for activity.

> **Context note:** This paper is not arguing against Zero Trust. NIST's definition emphasizes per-request decisions and protecting resources rather than network location. But many real implementations still stop at session establishment instead of governing in-session visibility and behavior. See NIST SP 800-207 for the canonical ZTA framing.

Why "detect and react" starts from a losing position

In many stacks, sensitive content renders on the endpoint and only then do controls attempt to prevent copy/paste, screenshots, uploads, or exfiltration through endpoint DLP, enterprise browser extensions, or monitoring.

That approach can help, but it begins from a structural disadvantage: once sensitive content renders on an endpoint, you've already handed over a lot of control.

This becomes sharper in agentic workflows because:

- Agents are fast and consistent, so they can replicate mistakes at scale.
- Agents can take actions without the "social friction" humans experience.
- Many applications were never built for fine-grained visibility controls.

A new model: Secure Rendering + Bidirectional Session Governance

The architectural move is simple to describe and hard to implement well:

Treat rendering as the control point.

Instead of asking only "can the user/agent access the app?", the policy questions become:

- Can they see this portion of a page?
- Can they download this document type?
- Can they copy data out?
- Can they paste data in?
- Can this interaction be allowed back into the application?

This is what makes Zero Trust real inside the session, not just at login.

Downstream control: what the app is allowed to show

"Downstream control" is applying policy before content is delivered for rendering.

The key idea is not "hide it later." It's: restricted content is excluded so it never renders on the endpoint.

What good looks like (downstream control)

- Field- and element-level redaction for sensitive UI components (not just whole-app allow/deny).
- Role + context based decisions (identity, task, data classification, workflow state).
- Deterministic enforcement at the rendering boundary rather than best-effort endpoint behavior.

Upstream control: what the user/agent is allowed to send back

"Upstream control" addresses an often-missed reality: the risk is not only data leaving; it's also unsafe or noncompliant data entering systems of record.

Bidirectional enforcement turns the session into a governed interface rather than a blanket trust zone.

Why this matters more for agentic automation

When agents operate visually and interact with applications like a person, governance must follow them into that interface layer.

For CISOs, the agentic leap raises the bar.

- **Visibility must be reconstructable.** Not "we had some logs," but "we can replay what happened and defend integrity."
- **Controls must be proactive**. Not only post-event proof, but prevention at the control point.
- **Containment must be enforceable.** Centralize policy and telemetry, avoid agents operating unbounded on endpoints.

The patent-backed techniques, explained in plain terms

The ideas in this paper can sound abstract until you translate them into one concrete question.

Where, exactly, do you enforce policy?

Most controls enforce at session establishment, then rely on downstream endpoint behavior and monitoring once content is already visible. The two patents below describe enforcement at a different boundary, the point where interfaces are rendered and where user or agent actions flow back to the system.

Patent 1: Bidirectional web application session control

U.S. Patent 12,088,598, Secure access via a remote browser is easiest to understand as a "policy-controlled web session." A protective layer evaluates permissions and then generates modified media instructions that shape the session in two ways.

First, it can exclude unauthorized portions of a web page so restricted content never renders locally. Second, it can apply the same session policy to what the user submits back to the application, filtering or constraining user-provided data and actions under that policy.

What this enables in practice:

- Fine-grained policy over what web content is rendered (at the level of page elements, not only whole apps).
- Policy-governed handling of user inputs and submissions back to the web app within the same session.
- Reduced reliance on after-the-fact controls once sensitive content has already been displayed.

Patent 2: Bidirectional control for remote client applications (desktop apps and remote systems)

U.S. Patent 12,341,782, Secure access via a remote client expands the model beyond web pages to client applications running on remote computers (desktop applications for example). It describes enabling end users to access and interact with those remote client applications through a protective layer, and preventing unauthorized content exchange back to remote systems by determining permissions for content and actions and generating modified media instructions to enforce those permissions.

What this enables in practice:

- Governance over what an agent can send back (uploads, pastes, submissions), and, where applicable, what it can see, when interacting with remote client applications.
- A more complete session policy model for workflows that live in desktop apps and other remote systems where APIs are limited.

Taken together, these patents describe policy-driven, bidirectional session governance across web browsers, desktop applications, and remote systems: controlling what content is rendered to the user and what content and actions are exchanged with the underlying systems.

Together, these methods operationalize a simple principle.

If governance is defined by what can be seen and what can be done, enforcement must live where rendering and interaction occur.

Controls that matter for CISOs

This section is intentionally practical: if you're deciding whether to approve agentic automation, these are the controls that determine whether a program is governable.

Policy enforcement at the rendering boundary

Objective: enforce policy at the point where interfaces are rendered and where actions are submitted, not only at session establishment.

Minimum control set

- **Pre-render enforcement:** prevent unauthorized content from rendering at d), not only allow or deny the whole application.
- **Interaction controls:** govern downloads, printing, clipboard, file transfer, and other interactions consistently.
- **Context-aware policy:** decisions based on identity, role, workflow state, and data class.
- **Bidirectional governance:** controls that apply both to what is shown and what can be sent back (uploads, pastes, tool actions).

Checklist (policy enforcement)

- ☐ **Pre-render control exists**: Can you prevent unauthorized content from rendering, not only watermark or detect after the fact?
- ☐ **Granularity is meaningful:** Can policy scope be narrower than an app, for example a field, element, record, section, or document region?
- ☐ **Decision inputs are explicit:** Are decisions based on identity, role, workflow state, and data class, plus session risk signals where relevant?
- ☐ **Bidirectional policy:** Are both "show" and "do" governed, including copy, paste, download, print, upload, and high-risk UI actions?
- ☐ **Deterministic enforcement:** Is enforcement consistent across endpoints, without relying on local agents or extensions to behave correctly?
- ☐ **Policy change control:** Is there an audit trail for policy edits (who changed what, when, why), with rollback?
- ☐ **Break-glass behavior:** Is there a controlled, auditable exception path for emergencies, with strict scope and time limits?
- ☐ **Session boundaries are clear:** Can you scope policy per workflow or per session, rather than creating permanent broad access?
- ☐ **Evidence is produced:** For each enforcement decision, can you log the policy evaluated, inputs, and outcome for later review?

Isolation and "no agents in the wild"

Objective: keep agent execution within a controlled boundary so you can constrain blast radius, revoke access quickly, and avoid unmanaged "agents in the wild."

Minimum control set

- **Centralized runtime:** agent execution is governed in a controlled environment rather than distributed across unmanaged endpoints.
- **Scoped network access:** connectivity constrained per workflow and per application, with reduced exposure to untrusted destinations.
- **Credential discipline:** minimize long-lived secrets, broker or scope credentials where possible, and prevent credential reuse outside policy.
- **Revocation controls:** immediate session termination and access revocation across humans and agents.

Checklist (containment)

- ☐ **Centralized, governed runtime:** Does the agent run in a controlled environment rather than on employee laptops or unmanaged VMs?
- ☐ **No unapproved runtimes:** Are there technical controls preventing "agent in the wild" paths from executing the same workflows?
- ☐ **Workflow-scoped network access:** Can you allow only required applications and destinations, and restrict general internet access where appropriate?
- ☐ **Segmentation by risk:** Can you isolate high-risk workflows into tighter network and policy boundaries with stronger evidence retention?
- ☐ **Credential discipline:** Are credentials scoped to the minimum necessary, rotated, and protected from extraction or reuse outside policy?
- ☐ **Least privilege by default:** Is access granted per workflow and time-bounded, not persistent broad entitlements?
- ☐ **Fast revocation:** Can you revoke agent access instantly and terminate active sessions, with proof revocation occurred?
- ☐ **Asset inventory:** Can you enumerate all agents, their owners, permissions, and the systems they can access?
- ☐ **Change management:** Are agent configuration and permission changes controlled, reviewed, and logged?
- ☐ **Egress controls:** Can you prevent data from being sent to unauthorized external destinations, including "helper" services?

Evidence-grade auditability (integrity-protected audit records + chain-of-custody)

Objective: produce audit records that withstand adversarial scrutiny and support incident response, compliance review, and internal governance.

Minimum control set

- **Complete event capture:** who did what, to which system, when, and under which policy.
- Trustworthy time: time synchronization and reliable timestamps across logs and recordings.
- **Integrity protections:** integrity-protected audit records for high-stakes events.
- **Chain-of-custody:** preserve evidence from capture through export so it can be defended.
- **Controlled access:** separation of duties and access controls for evidence storage, retrieval, and export.

Checklist (evidence-grade auditability)

- ☐ **Evidence package export:** Can you export a package that reconstructs activity end-to-end (events, policy decisions, and relevant session context)?
- ☐ **Who/what/when/where/why:** Do records include actor identity (human/agent), target system, action, result, policy evaluated, and reason codes?
- ☐ **Trustworthy time:** Are timestamps reliable and time-synchronized across components so sequences of actions are defensible?
- ☐ **Integrity protections:** Are records protected from silent modification or deletion (cryptographic sealing, immutability/WORM where warranted, retention locks)?
- ☐ **Chain-of-custody:** Is there a record of evidence handling (access, export, sharing), and can you prove integrity?
- ☐ **Admin activity logging:** Are administrative actions on logging, retention, and evidence systems recorded and reviewable?
- ☐ **Separation of duties:** Can you separate operators from evidence administrators and auditors, with least-privilege evidence access?
- ☐ **Retention controls:** Can you configure retention by workflow risk and enforce holds when required?
- ☐ **Privacy controls:** Can you minimize raw capture by default where possible and restrict sensitive evidence access?
- ☐ **IR readiness:** Can incident response use the evidence quickly to answer what happened, what data was exposed, and what actions occurred?

Use cases for governed agents in real enterprise environments

Agentic automation is already showing up in production-like deployments across industries. The examples below are representative of what teams are doing today.

Use case A: Cross-system user onboarding (HR + IT workflows)

Example pattern: Automating user onboarding across systems like Jira and BambooHR.

Why it matters: onboarding workflows touch identity, access requests, ticketing, and HR records, which makes them high-leverage and high-audit.

Governance requirements to validate

- Session watermarking for agent-operated sessions to deter misuse.
- Upstream control on what can be submitted into HR and IT systems (uploads, pasted data, and approval actions).
- Evidence package that reconstructs who approved what, when, and under which policy.

Use case B: Legacy line-of-business navigation (pharmacy and healthcare workflows)

Example pattern: Auto-filling prescription refills by navigating legacy pharmacy management systems.

Why it matters: legacy UIs often lack fine-grained authorization and are difficult to integrate via APIs, which pushes automation into the session layer.

Governance requirements to validate

- Session watermarking for agent-operated sessions, paired with strict workflow scoping so agents access only the screens and steps required for the task.
- Strict upstream controls on entries, selections, and submissions to avoid incorrect or unauthorized changes.
- Integrity-protected audit records suitable for regulated review.

Use case C: Clinical and equipment operations (medical device setup + tracking)

Example pattern: Automating IV pump setup and streamlining equipment tracking workflows that integrate with systems like SAP.

Why it matters: device and clinical operations combine safety, compliance, and operational uptime. Mistakes scale quickly when an agent repeats a flawed step.

Governance requirements to validate

- Clear workflow boundaries and step-level guardrails, not broad application access.
- Strong change control for agent configuration and permissions.
- High-fidelity evidence for high-risk actions (setup, configuration, inventory movements).

Use case D: Enterprise-wide SAP process automation (operational workflows)

Example pattern: Running centralized, cloud-hosted agents to execute enterprise SAP workflows at scale (for example: order management, procurement, finance ops), with controlled access paths into SAP. **Why it matters:** ERP automation changes core business records. A small policy mistake can have material financial impact.	**Governance requirements to validate** • Bidirectional governance on both what can be viewed and what can be committed back to the ERP. • Separation of duties and approvals for sensitive actions. • Audit-ready evidence packages that allow reconstruction of changes, policy decisions, and operator oversight.

Use case E: Personal agents for administrative and data tasks

Example pattern: Deploying personal agents to employees to automate repetitive administrative and data tasks. **Why it matters:** Personal agents increase the number of sessions and actions dramatically. Governance becomes an evidence and policy scalability problem.	**Governance requirements to validate** • Policy templates that scale across many agents while still supporting role and context differences. • Containment that prevents personal agents from becoming unmanaged automation on endpoints. • Evidence controls that support investigations without over-collecting sensitive data.

Evaluation framework

Evaluation checklist

Policy & enforcement

- [ ] Can policy be enforced before content renders (element/field/section)?
- [ ] Can you control inputs back to the app (uploads, pastes, tool actions)?
- [ ] Do policies apply consistently to humans and agents (no “agent exceptions”)?

Containment

- [ ] Is agent execution centralized/contained (“no agents in the wild”)?
- [ ] Can you scope network access per workflow/app and restrict egress?
- [ ] Can you revoke sessions instantly and prove revocation occurred?

Evidence & audit

- [ ] Do you produce evidence-grade logs designed for adversarial scrutiny?
- [ ] Are logs protected with integrity controls (sealing, immutability/WORM where needed)?
- [ ] Is chain-of-custody supported across capture → inference → action → export?
- [ ] Are timestamps trustworthy and time-synchronized (AU-8 intent)? [R4]

Operational reality

- [ ] Can teams start small with a focused use case and scale intentionally?
- [ ] Does the model avoid creating a new fragile integration maintenance burden (tokens/permissions sprawl)?
- [ ] Are privacy constraints addressed (minimize raw capture retention, control access to logs)?

Decision scorecard	
Score each 1–5 (1 = weak/absent, 3 = partial, 5 = strong and provable)	
① ② ③ ④ ⑤	**Downstream control strength** (pre-render redaction, granularity, determinism)
① ② ③ ④ ⑤	**Upstream control strength** (input governance, tool/action gating, exfil resistance)
① ② ③ ④ ⑤	**Containment** (centralized runtime, network segmentation, "agents not in the wild")
① ② ③ ④ ⑤	**Evidence-grade audit** (integrity + time correctness + chain-of-custody + exportable evidence packages)
① ② ③ ④ ⑤	**Operational friction** (deploy and maintain)
① ② ③ ④ ⑤	**Policy UX** (testability, change control, reviewability)
Red flags	
① ② ③ ④ ⑤	"We log everything" but no integrity guarantees.
① ② ③ ④ ⑤	Controls depend on endpoint behavior after sensitive data has already rendered.
① ② ③ ④ ⑤	Agent runtime is distributed across unmanaged machines ("agents in the wild").
① ② ③ ④ ⑤	Policies are app-level allow/deny only, no content-level or interaction-level enforcement.

Where this approach is NOT a fit

This model is powerful, but it's not universal. It may be a poor fit when:

- Your risk is primarily API-level and workflows never touch UI sessions.
- You require offline operation or local execution on endpoints by design.
- Your applications already enforce truly granular authorization internally.
- You can't justify evidence-grade retention for the category of work.

What to require for approvable deployment

Approving agents for production requires requirements that are testable, enforceable, and evidence-producing. Use the criteria below to evaluate whether a given approach can be governed at scale without expanding trust inside the session.

Require controls at the rendering and interaction boundary

- **Pre-render enforcement:** The platform must be able to prevent unauthorized content from rendering at meaningful granularity (field/element/section), not only block whole applications.
- **Bidirectional governance:** Controls must apply to what can be shown and what can be sent back (uploads, pastes, tool actions).
- **Consistency:** The same policies must apply to human sessions and agent sessions, with no "agent exception path."

Require containment (no unbounded agents)

- **Centralized execution:** Agent runtime should be centralized and governed rather than distributed across unmanaged endpoints.
- **Scoped connectivity:** Ability to constrain network access per workflow/app and reduce exposure to untrusted destinations.
- **Fast revocation:** Immediate session termination and access revocation with evidence that revocation occurred.

Require evidence you can take to audit

- **Evidence package export:** Define what must be exportable: event logs, integrity proofs (sealing/immutability as appropriate), timestamp posture, access history, and formats your GRC/IR teams can consume.
- **Integrity guarantees:** Demonstrate that audit records cannot be silently altered or deleted, and that administrative access is controlled and reviewable.
- Privacy-aware retention: Minimize raw capture by default where possible, restrict evidence vault access, and align retention to policy.

Require a pilot that proves governability

- Pilot success criteria must include: policy enforcement (downstream and upstream), containment, and evidence integrity under realistic conditions.
- Run a tabletop exercise with Security, GRC, and Incident Response using the exported evidence package to confirm you can reconstruct actions and defend integrity claims.

Conclusion

Zero Trust significantly improved how we decide who can begin a session, but in the agentic era, the biggest failures often happen after login. When sensitive content renders broadly, and when agents can act inside sessions faster than humans can supervise, the enterprise needs controls that operate at the true control point: rendering and interaction.

The practical model is:

- **Secure Rendering** (prevent overexposure before content appears)
- **Bidirectional Session Governance** (control what can be sent back)
- **Evidence-Grade Auditability** (integrity-protected audit records and chain-of-custody proof)

This is how CISOs turn "interesting pilots" into "approvable production."

Schedule a Demo

Contact Us :
sales@sonet.io

sonet.io

Secure Workspaces for Humans and AI

sales@sonet.io

Find us online at **sonet.io**

3031 Tisch Way, 110 Plaza West
San Jose, CA 95128