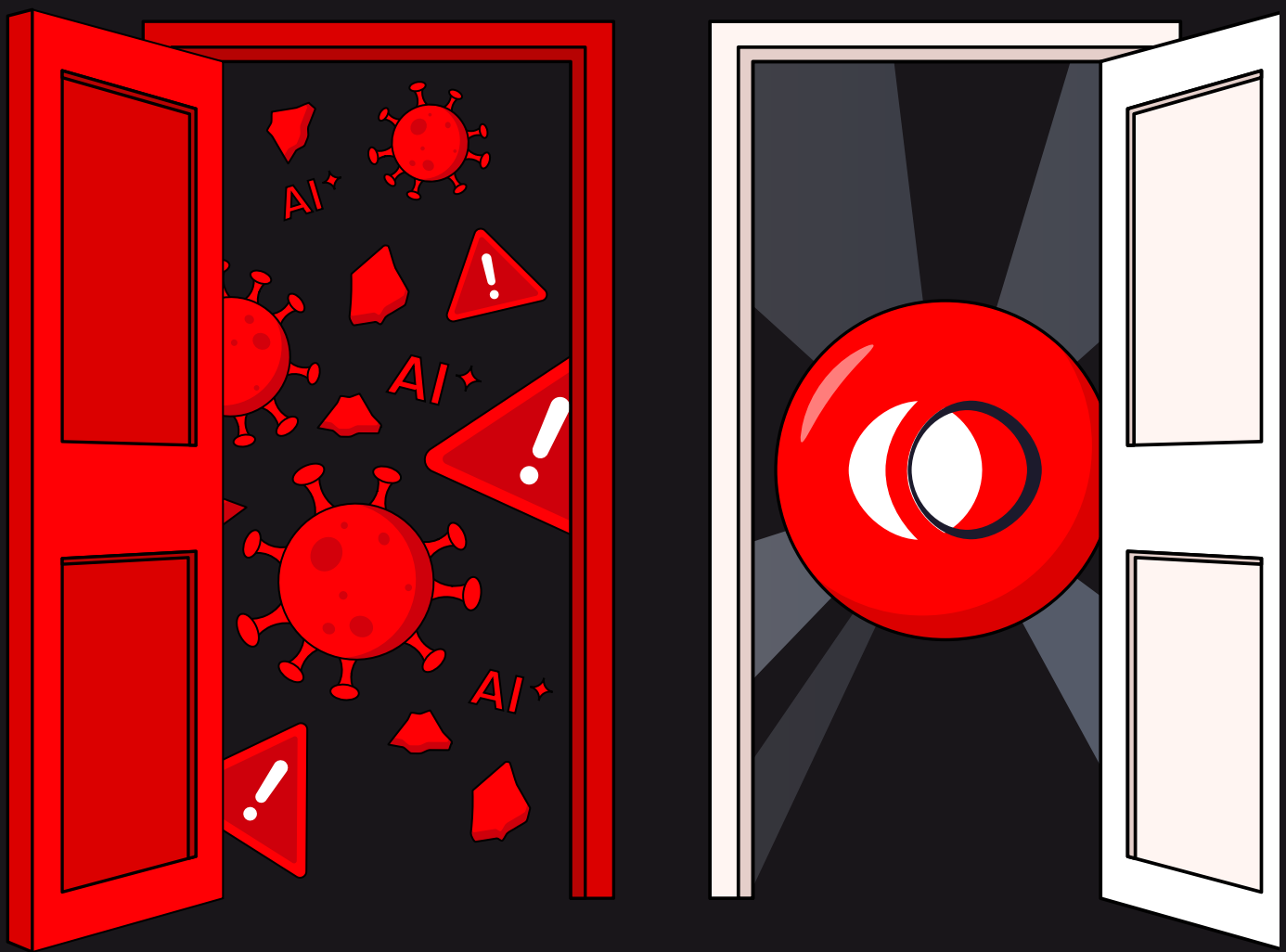
redaccess

# 3 Questions to Ask Before Your SSE POC

A practical guide to evaluating Secure Service Edge with clarity, realism, and real-world expectations



# Executive Summary

Secure Service Edge (SSE) has become the default answer for modern access security. As organizations embrace hybrid work, SaaS-first operations, and third-party access, SSE promises consolidation, simplified policy management, and unified visibility across users and devices.

On paper, it sounds like the perfect solution. But in practice, there's often a wide gap between the promise and what shows up in production.

Most SSE proof-of-concepts are built around vendor-curated demos, sanitized workflows, and idealized user behavior. They rarely reflect how real users actually work, on unmanaged devices, inside browsers, across dozens of SaaS apps, and increasingly with GenAI tools and AI browsers. The result is a false sense of confidence: the POC looks successful, only for blind spots to surface after rollout, when it's far more expensive to fix.

This guide outlines three critical questions security leaders should ask before starting an SSE POC. Each question highlights where SSE commonly breaks down, helping you design a POC that exposes real risk instead of masking it.

# Introduction: Is SSE Enough?

**GenAI adoption, browser-centric workflows, SaaS sprawl, BYOD, and third-party access have expanded the attack surface beyond what legacy network-centric tools were designed to handle. SSE emerged as a unified architecture, combining SWG, CASB, ZTNA, and sometimes DLP, delivered as a cloud service.**

**Conceptually, it makes sense. But consolidation alone doesn't guarantee coverage.**

## Challenge #1: Network "Rip and Replace"

Deploying SSE often means rerouting traffic through new cloud proxies, redesigning network paths, reworking firewall and VPN configurations, and shifting enforcement to entirely new control points. This touches core infrastructure and requires tight coordination between networking, security, identity, and IT teams, just to reach baseline protection.

## Challenge #2: Limited AI, Browser and Session Visibility

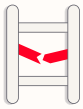
Most SSE platforms operate primarily at the network layer. They see connections, URLs, and traffic flows, but not what actually happens inside the browser or SaaS session. That's where modern risk lives: AI browsers, GenAI prompts, copy/paste of sensitive data, OAuth abuse, in-app exports, bulk downloads, and post-login script execution. Security teams think they're protected, but critical actions remain invisible and uncontrolled.

## Challenge #3: The Operational Tax

SSE is expensive, and it's not just the license. SSE carries a heavy "human tax." Managing proxy infrastructure, policy engineering, and constant exception handling requires a dedicated team just to maintain the status quo. This creates significant operational drag: security professionals end up as glorified policy operators instead of risk managers, while IT teams are sidelined by endless troubleshooting for performance and access issues.

## Challenge #4: Architectural Fragility

The underlying architecture of SSE is often too brittle for modern enterprise speeds. The interplay of agents, certificates, PAC files, and routing changes introduces risk at every layer. Because a single misconfiguration can trigger a localized outage or lock users out of critical apps, teams become hesitant to innovate. Security becomes a "black box" that people are afraid to touch, the exact opposite of a dynamic control layer.



## Challenge #5: The Deployment Gap

SSE projects are notorious for "timeline creep." Phased migrations, regional edge cases, and internal politics often stretch rollouts into multi-year marathons. This creates a dangerous lag: by the time the deployment is "finished," the organization's SaaS footprint and the global threat landscape have already evolved. You end up securing yesterday's environment with tomorrow's budget.

# 3 Questions to Ask Before Your SSE POC

**The SSE market is crowded with "all-in-one" promises and lists of features. But a successful POC should validate whether the solution actually reduces your most important risks.**

**Here are three questions to ask before and during POC engagements.**

## Question 1: Can SSE Solve My High-Priority Security Challenges?

**Most failed security projects start with vague goals like "improve security" or "modernize access." Before starting a POC, be explicit about the outcome you care about.**

**Is your main goal to:**

**1. Secure SaaS access at the session and action level -** Controlling risky in-app actions (export, download, copy/paste, print, etc.) and preventing mass data exfiltration from tools like Salesforce, Notion, Google Drive, etc.

**2. Protect the browser itself -** Dealing with broken, outdated, or risky browsers, defending against malicious or rogue extensions, and "Seeing inside the browser" instead of only at the network edge

**3. Govern GenAI usage -** Preventing sensitive data from being pasted into GenAI apps and enforcing DLP policies across GenAI tools.

**4. DLP -** Protect corporate data access browsers, SaaS applications, GenIA tools, browser extensions and messaging platforms, while ensuring consistent enforcement for employees and third-parties alike.

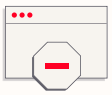
**5. Reduce risk from unmanaged or external devices -** Securing BYOD, contractors and external collaborators without deploying full agents.

# Where SSE Often Struggles

**SSE is a powerful framework for unifying network security, but because it primarily operates at the network layer it often lacks the "eyes and ears" inside the user's browser or device to manage granular actions.**

## Granular SaaS Action Control

- **Encapsulated Traffic:** Many in-app actions (like clicking a "print" button or "copying" text) don't trigger a new network request that an SSE proxy can intercept.
- **API Limitations:** SSE often relies on out-of-band CASB (API-based), which is "near real-time" but cannot block an action before it happens; it can only alert you after the data is already gone.
- **Visibility Gap:** Modern SaaS apps use complex JavaScript. SSE struggles to distinguish between a "view" action and a "download" action when both look like similar encrypted data streams.



## Protecting the Browser Surface

- **Blind to Local Exploits:** SSE sits in the cloud; it cannot see if a user has installed a malicious extension that is scraping data directly from the screen.
- **Device Health:** While SSE can check if a device is "known," it cannot easily verify if the browser itself is outdated or if a rogue process is injecting code into the browser's memory.
- **The "Last Mile" Gap:** Once the SSE proxy decrypts and inspects traffic, it sends it to the browser. The security ends there, leaving data vulnerable to screen capture or local browser cache theft.

## Governing GenAI Usage

- **The "Paste" Problem:** Copying and pasting sensitive code into GenAI apps or AI browsers is a local browser event. Unless the SSE proxy is performing heavy-handed SSL inspection on every single character stroke, it often misses these snippets.
- **Prompt Lack of Context:** SSE filters struggle to understand the intent of a GenAI prompt, making it difficult to differentiate between a harmless query and a prompt designed to leak intellectual property.

## Consistent DLP Enforcement

- **Fragmented Visibility:** SSE provides great DLP for traffic it "sees," but it misses data movement within local browser extensions or messaging apps that use proprietary encryption (like WhatsApp desktop or Slack's local cache).
- **Third-Party Blind Spots:** Enforcing DLP on contractors often requires "steering" their traffic through the SSE cloud, which is difficult to do on a device you don't own without a heavy agent.

## Risk from Unmanaged/ External Devices (BYOD)

- **Agent Fatigue:** SSE usually requires an agent (PAC files or connectors) to steer traffic. Contractors and partners are often unwilling or unable to install corporate agents on their personal hardware.
- **Vulnerability to Malware:** Without an agent on the device, SSE cannot guarantee that a BYOD laptop isn't already infected with a keylogger that captures credentials before they ever reach the "secure" SSE tunnel.

## Question 2:
## How Frictionless Is Deployment?

**The best security solution isn't the one with the most features. It's the one that actually gets deployed and adopted.**

**Key questions to test during your POC:**

- **Endpoint impact**

  Does it require an agent on every device?
  What happens with BYOD, contractors, or unmanaged endpoints?

- **Network impact**

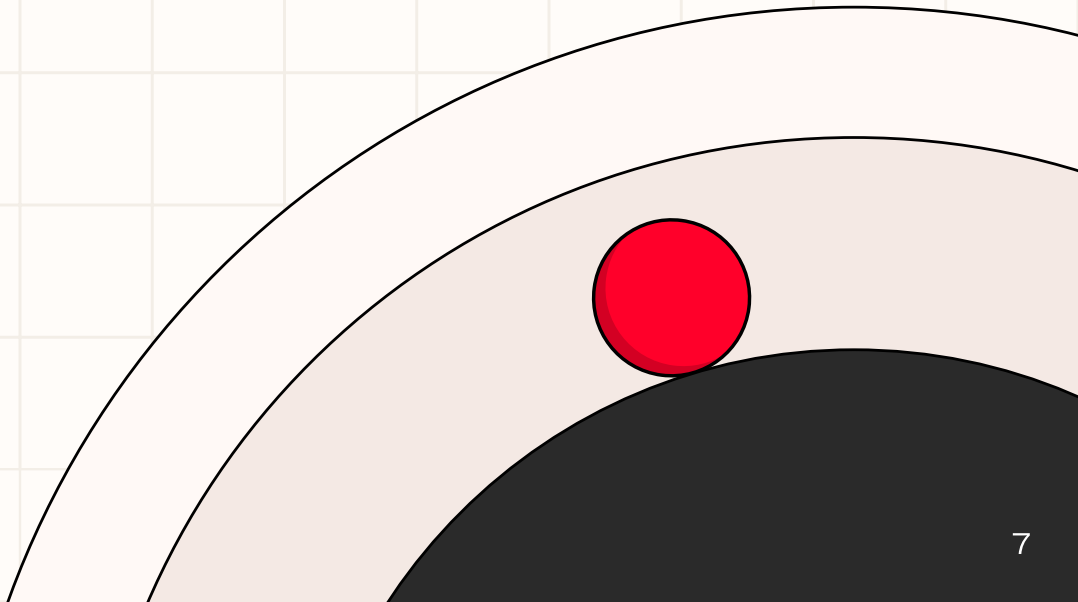
  Does it require traffic steering, tunneling, or proxying?
  Does it force coexistence with, or replacement of, existing VPNs and firewalls?

- **Rollout model**

  Can it be enabled incrementally (by app, user group, or risk)?
  Or does it require a big-bang cutover?

- **Time to protection**

  How long until meaningful enforcement, not just visibility?
  How quickly can risky behavior actually be blocked?

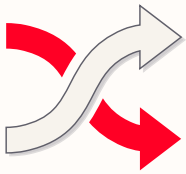


# Where SSE Often Struggles

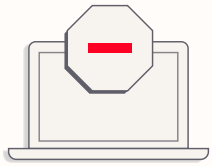
SSE diagrams look simple. Deployments rarely are. What's sold as cloud-native often still requires network rerouting, identity rewiring, parallel policy stacks and traffic hairpinning through POPs.

Common friction points:

Mandatory agents



Tunnels and traffic redirection



Device posture checks that block legitimate users

Complicated exception handling

**All of this slows rollout and creates internal resistance.**

## Question 3:
## What are the Real Costs and Required Resources?

**Sticker price isn't the only security cost of a tool. The true cost includes people, time, and complexity. Your POC should validate not just technical fit, but economic and operational sustainability.**

**Evaluate:**

- **Direct costs**

  Licensing models, per-user or per-device pricing, and premium features like DLP.

- **Infrastructure costs**

  Traffic routing, gateways, bandwidth, latency, and integrations with IdP, SIEM, CASB, MDM, and EDR.

- **Human costs**

  Staffing requirements for deployment, tuning, exception handling, and alert response. Dependence on professional services or managed services.

- **Time costs**

  Deployment timelines, time to protection, admin training, user friction, and ongoing support overhead.

- **Opportunity costs**

  Which projects get delayed or deprioritized because this absorbs capacity?

## Where SSE Often Struggles

SSE platforms are often priced attractively upfront but require identity rewiring, network re-architecture, parallel policy management and long professional services engagements. These inflate the real cost significantly.
Even after go-live, policies need tuning, exceptions pile up, alerts grow and edge cases consume time. The operational cost often exceeds the license cost within a year.

# What is Agentless Session Security?

Agentless session security means protecting every user's web and SaaS activity at the session level (e.g., the actual browsing session in the browser or web app) without installing any software agents, browser extensions, or special browsers on user devices.

Traditional security tools often require endpoint agents, extensions, or network reroutes (like VPNs), which can be hard to deploy, slow performance, or create friction for users. Session-level security works by silently routing traffic for inspection and policy enforcement through its cloud service, so users stay secure without changing how they work and IT doesn't have to manage lots of client-side software.

Here's what that means in practice:

## Session-level protection:

Instead of just monitoring packets or installing local software on devices, the solution watches and secures **each browsing and web app session** in real time. It can apply DLP, block phishing or malware, govern GenAI prompts, and enforce zero-trust policies while the session is active.

## Agentless by design:

There's no agent to install on endpoints or browser, no browser plugin, and no special browser required. A simple configuration enables the security controls across all browsers and web apps.

## Seamless user experience:

Because it doesn't disrupt workflows or force users to install anything, people can use any browser or SaaS app normally while still being protected.

## Deep policy enforcement:

Security teams get granular control and visibility (e.g., who accessed what app, what data was shared, risk indicators, shadow SaaS detection) without heavy infrastructure changes.

# SSE vs. Agentless Session Security: A Practical Comparison

| | SSE | Agentless Session Security |
|---|---|---|
| **Deployment model** | Agents, tunnels, PAC files, traffic routing | Browser-native, no agents, no rerouting |
| **Managed devices** | Strong | Strong |
| **Unmanaged devices** | Limited | Full session-level control |
| **Browser protection** | URL-level | Action-level (copy, export, download) and browser-level (patching, extensions, etc.) |
| **SaaS session control** | Connection-focused | Behavior-focused |
| **GenAI and AI Browser protection** | Rule-based, limited | Behavioral, intent-aware |
| **Insider risk** | Partial | Behavioral detection |
| **Visibility** | App-level events | High-fidelity user actions |
| **Latency** | Medium | Low |
| **Operational complexity** | High | Low |
| **Best fit** | VPN replacement, managed endpoints | SaaS, GenAI, AI browsers, BYOD, contractors, insider risk |

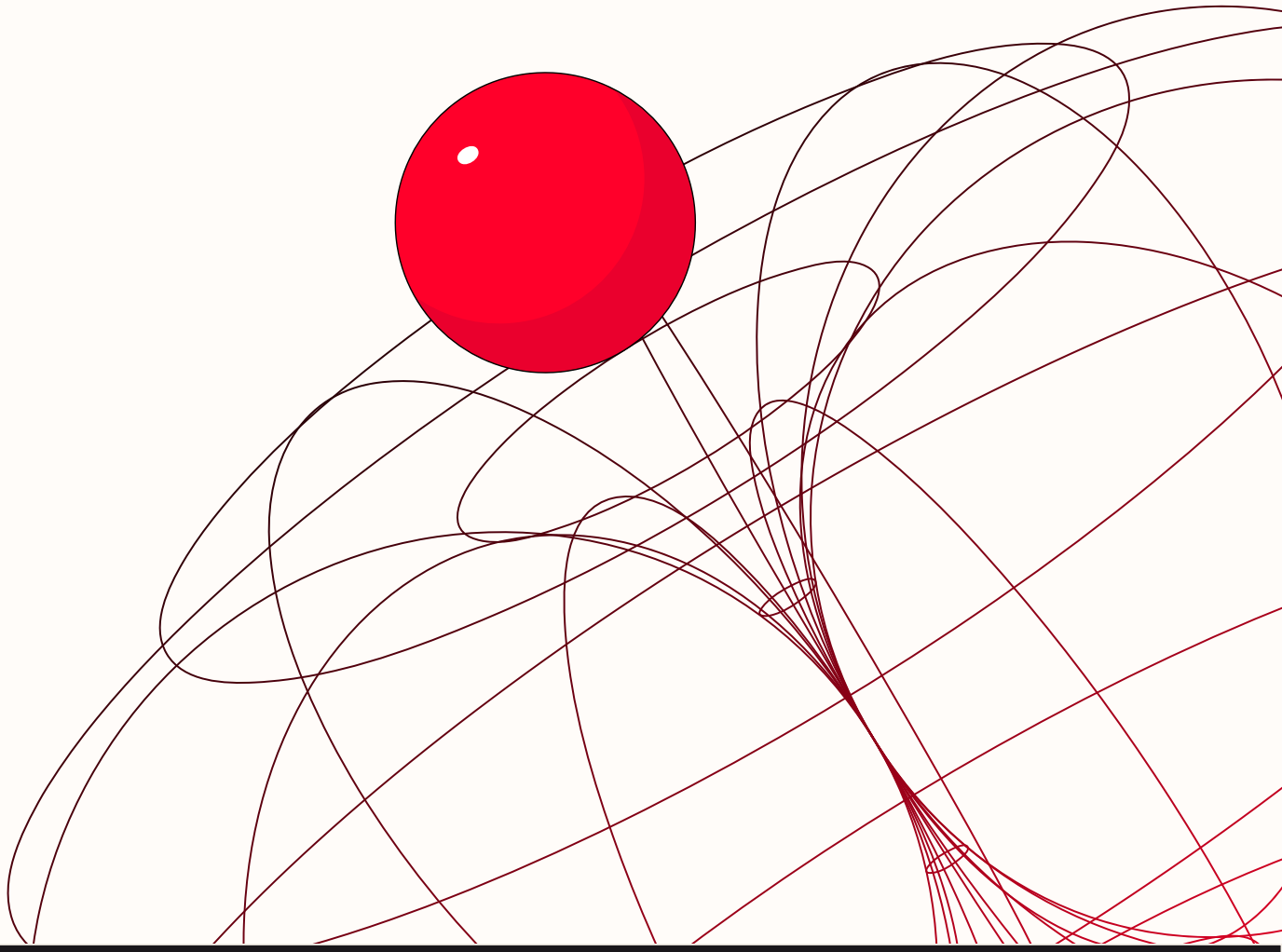
# About Red Access

Red Access is a cloud-native, agentless security platform that protects all web and SaaS activity, without requiring browser extensions or endpoint agents. It's deployed in minutes via a simple configuration through tools like Intune, MDM, or GPO, silently routing browser and app traffic (including Browsers, Embedded browsers, Webview2, and other HTTP/S traffic) through its secure cloud for real-time inspection.

Unlike traditional proxies or heavyweight SSE solutions that introduce friction, latency, or require full redirection, Red Access uses selective, context-aware traffic routing, letting organizations "turn on" security exactly where they need it, without disrupting what doesn't require control. Even when enabled, it runs silently, with no performance hit to users.

Red Access enforces deep security policies like DLP, phishing and malware protection, SaaS access control, and GenAI usage governance across any browser, device, or location. It integrates seamlessly with firewalls, identity providers, and SIEMs, giving security teams full visibility and control without disrupting users or overhauling infrastructure.

redaccess



# The Market is Moving to the New SSE

DISCOVER RED ACCESS