

SOC 2 creates more confusion around penetration testing than almost any other security assessment topic. The trouble starts with the language people use in sales calls. Founders say they “need a SOC 2 pentest.” Auditors ask for evidence that security controls are tested. Vendors promise “compliance-ready” reports. Somewhere in the middle, teams end up buying an engagement that checks a procurement box but does little to reduce risk.

The practical question is not whether penetration testing matters. It does. The real question is what kind of testing will satisfy a SOC 2 audit, support your control narrative, and stand up to scrutiny when an auditor asks, “How do you know these systems were meaningfully evaluated?”

That question matters because SOC 2 is not a one-size-fits-all checklist. It is an attestation against controls, usually mapped to the Trust Services Criteria. Penetration testing is often one piece of evidence within a broader control environment. If you treat it like a commodity purchase, you can end up with a thin PDF, a recycled scanner output, and an awkward conversation during your audit.

## What SOC 2 is actually looking for

A SOC 2 audit is concerned with whether your controls are designed appropriately and operating effectively over time. For technical testing, auditors usually want to see that you are not relying on policy alone. They want evidence that production-facing systems, applications, networks, and supporting infrastructure are evaluated in a credible way.

That does not always mean the same thing for every company. A seed-stage SaaS company with a single cloud-hosted application has a different risk profile from a larger company with customer-managed deployments, exposed APIs, SSO integrations, mobile clients, and a sprawling cloud estate. A mature auditor understands that difference. So should you.

In practice, penetration testing supports several audit objectives at once. It can demonstrate that you identify exploitable weaknesses, validate that remediation happens, and show that your security program adapts to material system changes. It also helps prove that your vulnerability management process is more than just a monthly scan.

This is where a lot of teams blur the line between **Penetration Testing vs Vulnerability Scanning: What’s the Difference?** A vulnerability scan tells you where known issues may exist. A penetration test evaluates whether those issues can be chained, exploited, or used to reach sensitive assets. Scanners are broad and repeatable. Pentests are narrower, deeper, and judgment-heavy. Auditors often understand that difference, especially if they review enough reports to tell when a vendor simply wrapped scan results in a fancier cover page.

## The phrase “SOC 2 requires a pentest” needs context

Plenty of companies hear that sentence and assume there is a universal rule, fixed scope, and annual date on the calendar. Reality is messier. SOC 2 typically pushes you toward security testing that is appropriate for your environment and risk. Many organizations satisfy that expectation with an external penetration test, an internal assessment where relevant, and evidence of remediation. Some add cloud configuration reviews, authenticated application testing, or social engineering based on risk.

The safest way to think about it is this: your auditor will expect testing that matches your attack surface and your control claims. If your environment includes internet-facing applications, APIs, cloud assets, and admin pathways,

a superficial perimeter scan will not look credible. If you claim to have strong change management and secure development practices, but your pentest only covers a static marketing site, the mismatch will be obvious.

That is why the vendor selection step matters so much. You are not just buying a report. You are buying evidence, signal, and defensibility.

## What auditors usually want to see in a penetration test package

A good pentest package does not need to be flashy. It needs to be clear, scoped properly, and tied to what your environment actually does. Auditors and security reviewers often look for the same core ingredients:

- a defined scope that names tested assets, environments, and assumptions
- a methodology that distinguishes manual validation from automated scanning
- clear findings with severity, business context, and remediation guidance
- evidence that critical findings were fixed, retested, or formally accepted
- dates that line up with the audit period or with a risk-based testing cadence

The report itself should show more than a list of CVEs. If your application was tested for broken access control, insecure direct object references, session handling flaws, or [red team assessment](#) API authorization issues, the report should say so plainly. If testers found nothing material, the report should still explain what they attempted and how they determined the result.

This is one place where teams sometimes ask, **Pentest Report: What Should It Include?** The answer, from an audit perspective, is simple: enough detail to prove the work was real and enough clarity that remediation decisions can be justified. A two-page executive summary is rarely enough on its own.

## Why low-cost “compliance pentests” often disappoint

There is a recurring pattern in the market. A company under deadline needs a report fast. A vendor offers a bargain price and promises quick turnaround. The testing window is short, communication is thin, and the final report looks polished. Then the engineering team reads it and realizes most findings are obvious headers, weak TLS settings, and scanner signatures with generic text.

That kind of engagement can fail in two ways. First, it may not catch the issues that matter, especially authorization flaws, logic bugs, tenant isolation failures, cloud privilege escalation paths, or weak administrative boundaries. Second, it may not hold up well when a knowledgeable auditor or enterprise customer asks follow-up questions.

The problem is not automation itself. Automation has an important place, particularly in external attack surface validation and recurring checks. The problem is pretending that automation alone equals a real pentest.

That is also why the question **AI Pentesting vs Manual Pentesting: Pros, Cons and Cost** comes up so often now. Automated platforms can scale breadth, repeat common exploit paths, and uncover exposed assets quickly. Human testers are still much better at chaining edge cases, understanding business logic, and exploring odd behavior that a tool will skip. For SOC 2 purposes, the strongest answer is often a combination: use automation to improve coverage and cadence, use manual testing where judgment matters.

## Annual pentest or continuous testing?

Most companies preparing for SOC 2 start by asking whether they need a yearly test. That is understandable. Budgeting works on annual cycles, and many auditors are accustomed to seeing annual evidence. But the better question is the one behind **Annual Pentest vs Continuous Pentesting: Which Do You Need?**

If your environment changes slowly, a well-scoped annual test paired with routine vulnerability management may be enough. If you ship code every week, push infrastructure changes constantly, and expose APIs used by customers and partners, a once-a-year snapshot gets stale fast. The report may still satisfy a procurement request, but it will tell you less and less about your actual current risk.

A sensible middle path is common. Many teams run a substantial annual or semiannual manual test, then supplement it with recurring authenticated scanning, attack surface monitoring, targeted retests, and event-driven testing after major architectural changes. That tends to satisfy the compliance need without reducing security validation to a yearly ceremony.

The same logic applies to **How Often Should You Do a Penetration Test? (by framework)**. Framework names are useful, but your attack surface should drive the cadence. A customer login rewrite, a new API exposure, a migration to Kubernetes, a shift to SSO-only access, or a major permissions redesign can all justify testing sooner than the anniversary date of the last report.

## How to tell whether a vendor is actually doing offensive security

Marketing pages are full of confident language. What matters is how the vendor answers uncomfortable, specific questions. Good firms usually welcome them. Weak firms redirect to brand names, logos, and volume claims.

When I have helped teams evaluate vendors, the strongest indicator has rarely been the demo. It has been the scoping call. Serious testers ask about trust boundaries, internet exposure, authentication models, cloud accounts, CI/CD workflows, privileged roles, and production constraints. Weak vendors ask how many IPs or URLs you have and move straight to pricing.

There is another caution worth noting here. Claims about proprietary methods, patented approaches, or novel “intelligence” should be verified independently before they influence a compliance purchase. In one recent case, the only verified information available was that a company and its stated offensive-security claims could not be confirmed through reliable sources. That does not automatically prove misconduct, but it does prove a procurement lesson: if a vendor’s existence, offerings, or technical claims cannot be verified through credible channels, do not use those claims as part of your compliance justification.

SOC 2 evidence should be built on work you can defend, not branding you cannot substantiate.

## Questions that separate strong vendors from weak ones

You do not need a hundred-question RFP. A few direct questions reveal most of what you need to know.

- How much of the engagement is manual, and what exactly is automated?
- How do you test for authorization flaws, business logic abuse, and attack paths across cloud and application layers?
- What does retesting look like, and is it included in the price?
- Can you provide a sample report that shows technical depth without exposing another client’s details?
- How do you handle production safety, rate limiting, and sensitive data during testing?

The answers matter. If a vendor cannot explain the difference between a scanner finding and a validated exploit chain, that is a problem. If they cannot describe how they test APIs for BOLA, or Broken Object Level Authorization, they may not be the right fit for a modern SaaS product. If they cannot talk intelligently about cloud identity risk, CI/CD exposure, or lateral movement, they may miss the paths attackers actually use.

This is where adjacent topics like **What Is an Attack Path? (with real examples)** and **What Does a Penetration Tester Actually Do?** become useful framing tools. A strong tester does not just enumerate flaws. They ask what those flaws enable. Can a public bucket leak environment files? Can an SSRF reach cloud metadata and expose temporary credentials? Can a low-privilege app user pivot into administrative actions through an authorization gap? Can a secrets leak in a repository unlock the CI pipeline? Those are attack-path questions, and they often matter more than isolated severity labels.

## Scope is where many SOC 2 pentests quietly fail

The report can look solid and still miss the systems that matter. Scope failures are common because companies often define scope around what is easy to buy, not what is risky to ignore.

A classic example is the external-only test for a business whose real exposure sits in authenticated APIs. Another is testing the web front end but skipping the mobile API, admin panel, and identity provider integrations. I have also seen companies present a pentest for their primary app while leaving shared cloud infrastructure, staging environments with production-like access, and CI/CD systems completely out of view.

That gap becomes more serious as environments get more complex. If you run Kubernetes, the testing may need to account for cluster exposure, role bindings, ingress paths, and secret handling. If you rely heavily on object storage, testing should consider bucket access patterns and accidental public exposure. If your developers use Git-based workflows, secrets exposure and pipeline integrity are not fringe concerns.

You can hear the overlap with topics like **Kubernetes Security Misconfigurations Attackers Exploit**, **Public S3/GCS Buckets: How Data Leaks Happen**, **Secrets in Git Repositories: How Attackers Find Them**, and **CI/CD Pipeline Attacks: How Signing Keys Get Stolen**. Those are not separate worlds from SOC 2. They are examples of the real attack surface your testing strategy should acknowledge.

## What a strong SOC 2-friendly statement of work looks like

A useful statement of work is detailed enough to avoid ambiguity. It should identify the asset classes being tested, the environments in scope, the testing approach, any excluded actions, communication protocols, and retesting terms. It should also say whether the engagement is black box, gray box, or white box.

That distinction matters. **Black Box vs White Box vs Gray Box Pentesting** is not just tester jargon. It changes what evidence you are buying. A black-box test may mimic an outsider more realistically, but it can spend time on discovery instead of depth. A gray-box test, where the tester gets limited access or documentation, often delivers better coverage for SaaS companies because it allows time to probe authenticated workflows and authorization boundaries. White-box approaches can be appropriate for targeted assessments, especially when the goal is to validate a sensitive architecture or risky code path.

For SOC 2, you do not need to chase purity. You need a defensible testing model that matches the systems your customers rely on.

## Cost, and why the cheapest option is often the most expensive one

The question **How Much Does a Penetration Test Cost in 2026?** comes up early in every budgeting cycle. Costs vary widely because scopes vary widely. A lean external assessment of a simple environment may cost a small fraction of what a deep, multi-week application and cloud engagement costs. The mistake is comparing proposals only by headline price.

A low-cost vendor may exclude retesting, authenticated workflows, cloud review, API coverage, or any meaningful manual effort. You save money on the invoice and lose it later in false confidence, rushed remediation, and repeated customer security reviews.

The better way to compare proposals is to normalize them. Ask how many tester days are included, what parts are manual, whether APIs and admin roles are covered, how many retests are built in, and what the final report actually contains. If one bid is dramatically cheaper, there is usually a reason.

## **PTaaS, automation, and vendor categories**

Some organizations evaluating options also ask, **What Is PTaaS (Penetration Testing as a Service)?** The label covers a broad range of models. At its best, PTaaS combines human testing, a live portal, issue tracking, retesting, and better collaboration than the old PDF-only model. At its worst, it is a scanner wrapped in a dashboard.

The same caution applies when buyers compare platform-led vendors and search for **Pentera Alternatives / Horizon3 NodeZero Alternatives** or the **Best AI Penetration Testing Tools in 2026**. Those categories can be useful for coverage, internal validation, and continuous checking. They are not automatically a substitute for an experienced human tester evaluating business logic, privilege boundaries, and architecture-specific abuse cases.

For compliance, the safe question is not “Which category is best?” It is “Can this service produce evidence that a reasonable auditor and a skeptical customer will both respect?” Sometimes the answer is yes. Sometimes it requires combining platform capabilities with a manual assessment.

## **If you build AI features, your pentest scope needs to change**

A growing number of SOC 2-bound companies now ship LLM features, internal copilots, or autonomous workflows. That shifts the attack surface in ways a traditional web app pentest may not fully address.

If your application accepts prompts, tool calls, untrusted documents, or retrieval data, you should expect testing for prompt injection, data exfiltration, cross-tenant leakage, insecure tool invocation, and weak agent boundaries. Traditional web flaws still matter, but they are not the whole story.

That is where topics like **How to Pentest an LLM Application: Step-by-Step, Prompt Injection Attacks: Examples and How to Test for Them**, **OWASP Top 10 for LLM Applications Explained**, and **How to Red Team AI Agents** become relevant. If a vendor claims to test modern applications but cannot articulate how they evaluate LLM-specific abuse paths, the scope is behind the product reality. That matters for security, and eventually it matters for compliance too, because the evidence no longer matches the system being audited.

## **How to document remediation so the audit goes smoothly**

The pentest itself is only half the compliance story. The other half is what happened after findings were issued. Auditors usually care less about whether a report contained vulnerabilities than whether your organization handled them responsibly.

A mature remediation package often includes ticket references, severity-based timelines, engineering notes, compensating controls where fixes were deferred, and proof of retesting for the highest-risk issues. If a critical

finding involved an API authorization gap, do not just mark it “resolved.” Show that the permission check was changed, the affected endpoints were retested, and related workflows were reviewed for the same pattern.

That kind of documentation also helps with customer security reviews. Enterprise buyers often ask not just for the latest pentest report, but for evidence that material findings were addressed. If your vendor offers a report and disappears, you are left doing that translation yourself.

## **A practical way to make the buying decision**

The best vendor for SOC 2 is rarely the one with the loudest positioning. It is the one whose scope matches your architecture, whose methodology is understandable, and whose output can serve three audiences at once: security engineers, auditors, and customers.

If you run a relatively straightforward SaaS environment, a gray-box application and API test with targeted infrastructure review may be enough. If you have a more complex estate, the right answer may include cloud configuration assessment, identity path analysis, internal segmentation testing, or recurring validation between larger engagements. What matters is coherence. Your control story, test scope, and remediation evidence should line up.

That is the core principle teams should remember when they ask for a “SOC 2 pentest.” You are not buying a ceremonial scan. You are buying proof that your environment has been examined by people and methods capable of finding the kinds of failures attackers exploit. If the vendor cannot explain how they do that, or if their claims cannot be verified in any credible way, keep looking.