



Web application penetration testing: Vendor evaluation checklist

Choosing a web application penetration testing company is not just a features comparison. What separates useful testing from an expensive checkbox exercise comes down to a handful of factors: whether findings are validated, whether testing covers your actual business logic, how deep API testing goes, whether it fits your release cycle, and how usable the reporting actually is.

This checklist is a companion to our guide comparing the top web application penetration testing companies in 2026. Use it to evaluate any vendor you are considering, whether or not they are on that list.

Work through it with your security and engineering teams before signing with a vendor. Tick items when the vendor has demonstrated it, not when they have simply said yes.



Manual exploit validation vs enumeration

- ☐ **Ask whether findings are validated as exploitable, not just flagged**

Confirm the vendor demonstrates that a vulnerability can actually be exploited in your environment rather than listing it as a theoretical possibility. This is what separates a useful report from a list your team has to re verify manually.

- ☐ **Ask how false positive rates are controlled**

Confirm the vendor can speak to how they keep false positives low. A high false positive rate means your team spends more time re checking findings than fixing real issues.

- ☐ **Ask for an example of a finding that was proven exploitable, not just listed**

A vendor that can walk through a real validated finding, including how they confirmed it was exploitable, demonstrates a testing process built around proof rather than enumeration.

Business logic coverage

- ☐ **Confirm testing covers your specific workflows, not just generic vulnerability classes**

Ask whether the vendor's methodology can model purchase flows, approval chains, role transitions, and permission boundaries specific to your application, not just checks that apply to any web application.

- ☐ **Ask for examples of business logic findings from similar applications**

A vendor that can show past business logic findings, not just generic OWASP Top 10 results, has a testing process built to catch workflow specific flaws.

- ☐ **Confirm the vendor can test role transitions and permission boundaries**

Ask how testing accounts for the difference between what a role should be able to do and what the application actually allows, since this is where high impact logic flaws tend to live.

API testing depth

- ☐ **Confirm REST and GraphQL endpoints are both covered**

Modern applications enforce authorization in APIs, not just in the UI. Confirm the vendor's methodology covers both protocols rather than treating one as an afterthought.

- ☐ **Ask how the vendor tests authorization at the API layer, not just the UI**

Confirm testing checks whether one user can reach another user's data or tenant through the API directly, since access control gaps at this layer are often invisible from the interface.

- ☐ **Confirm testing goes beyond endpoint discovery**

Ask whether the engagement includes token based authentication flow testing and object level and cross tenant boundary checks, not just a list of endpoints the scanner found.

CI/CD integration

- ☐ **Confirm testing can run inside your release cycle, not just as a periodic engagement**

If security testing only happens once a quarter, it is already behind your release cycle. Confirm the vendor can integrate with your pipeline so findings surface before code reaches production.

- ☐ **Ask how findings are delivered back into your existing workflow**

Confirm results can reach your team through the tools you already use, so testing does not become a separate process your engineers have to check manually.

- ☐ **Confirm testing frequency matches your actual release cadence**

If your team ships weekly, ask whether testing can realistically keep pace, or whether it is still structured around quarterly or annual engagements.

Reporting quality

- ☐ **Request a sample report before committing**

Confirm the report includes what was found, how to reproduce it, what the business impact is, and how to fix it in the context of your specific stack, not generic remediation advice.

- ☐ **Confirm reports are built for engineers, not just security teams**

A report that requires back and forth between security and development to interpret slows remediation. Confirm findings are written so the engineer who owns the fix can act on them directly.

- ☐ **Confirm each finding includes reproduction steps and stack specific guidance**

Ask whether remediation guidance is generic advice or written for your specific technology stack, since generic guidance is what creates back and forth between security and development.

Remediation support and retesting

- ☐ **Confirm retesting is included, not sold as a separate engagement**

Identifying a vulnerability is step one. Confirm the vendor retests after remediation to verify a fix actually closed the finding, rather than leaving that verification to your team.

- ☐ **Ask what happens when a fix does not fully resolve a finding**

Confirm there is a defined process for findings that reopen or were only partially fixed, so a recurring issue does not require starting a new engagement from scratch.

- ☐ **Confirm there is a clear process for closing out findings**

Ask how the vendor marks a finding as resolved, and what evidence they require before considering it closed, so remediation status is not left to assumption.

Match the choice to your priority

The right vendor depends on what your team needs most right now: continuous automated coverage that fits your release cycle, deep manual testing for complex business logic, or fast turnaround through a managed testing service. Use this checklist to compare vendors against what actually matters for your application, not just their feature list.