Web app penetration test scoping questions

These are the questions a SilentGrid consultant asks when scoping a web application or API penetration test. Answering them first makes the call shorter and the quote more accurate.

"Not sure" is a fine answer. Working out the unknowns is part of what the scoping call is for.

Questions marked **Sizing** help us prepare an accurate quote.

01 Before any test

What prompted the test, and what date does the result need to meet?

Why we ask: the reason sets the depth and report format, and the date fixes the testing window and any retest.

Will testing run against production, staging or a dedicated test copy?

Why we ask: production needs tighter limits on load, data and timing. A test copy allows more thorough techniques.

When can testing run, and are there change freezes or peak periods to avoid?

Why we ask: the schedule is built around them.

Will you want fixes retested?

Why we ask: retesting is agreed at scoping, so it is scheduled and priced from the start.

02 Web applications and services

How many applications are in scope, and what is each built on? Framework, language and hosting. SIZING

Why we ask: the count drives effort, and the stack decides which classes of flaw are most likely.

How many user roles are there, including administrators, and can you provide two accounts per role? SIZING

Why we ask: authorisation testing needs a second user at each level, so one can try to reach the other's data or functions.

Is it multi-tenant, and can we have accounts in two tenants?

Why we ask: one customer reading another's data is the most serious flaw a SaaS platform can have.

Which APIs sit behind it: REST, GraphQL, SOAP or gRPC? How many endpoints, and is there an OpenAPI file or Postman collection? SIZING

Why we ask: documented APIs can be covered completely. GraphQL needs its own test cases.

How do users sign in: passwords, single sign-on through SAML or OIDC, multi-factor, magic links?

Why we ask: each sign-in flow has its own known bypasses, from token handling to skipped multi-factor steps.

Which workflows carry business risk: payments, approvals, refunds, account recovery, exports?

Why we ask: business logic flaws in these workflows are what automated scanners miss.

Does it fetch or process outside content: webhooks, URL previews, file imports, document generation?

Why we ask: server-side request forgery and injection often hide in these features.

Is source code available?

Why we ask: code-assisted testing finds flaws that are invisible from outside, such as unsafe deserialisation or hard-coded secrets.

Discuss your testing priorities

www.silentgrid.com/contact