SME Digital Advisor

Plain-English advice on digital risk

OWASP · 10 CARDS

OWASP Top 10 Flashcards

The OWASP Top 10 web application security risks, translated for owners. 2021 edition.

OWASP A01 · WEB APPLICATION RISK

Can the wrong person on your website see or change the wrong data?

The most common web-application flaw. About whether the website properly enforces who can do what once someone's logged in. A flaw means customer A can sometimes view customer B's records by changing a URL, or that "admin" pages exist that any logged-in user can reach.

Questions to ask your developer:
1. *"Show me the access-control matrix for our customer portal — who can see what, who can do what. If you can't produce one, that's the answer."*

2. *"When the portal was built, did you specifically test 'logged in as one customer, try to view another customer's data'?"*

3. Decision: For anything that handles money or personal data, commission an external penetration test before launch and annually after.

OWASP A02 · WEB APPLICATION RISK

Is sensitive data scrambled properly when it's stored or sent?

Two things: data sent over the internet, and data stored on disk. Classic failures: HTTPS not enforced everywhere, passwords stored as plain text, sensitive data sat unencrypted in a database.

Questions to ask your developer:
1. Verify yourself: Test your site at SSL Labs. Aim for an A grade. If it's lower, ask why.

2. *"Show me how passwords are stored. If you can decrypt them to compare on login, the design is wrong — they should be one-way hashed with bcrypt or argon2."*

3. Decision: What sensitive data are we keeping that we don't actually need? Agree a retention period and a deletion schedule.

OWASP A03 · WEB APPLICATION RISK

Can someone trick your website into running their own commands?

The classic web flaw and still common: when user-typed input gets passed directly to a database or system command, an attacker types something like `'; DROP TABLE customers;--` and the database obediently does it.

Questions to ask your developer:
1. *"Are parameterised queries (or the framework's ORM) used everywhere — or are we still building SQL strings by hand anywhere?"*

2. *"Has the site been scanned with OWASP ZAP or a similar tool? When? Can you run one now and share the report?"*

3. Decision: For business-critical apps, commission a penetration test at launch (£3k–£10k) and annually after.

OWASP A04 · WEB APPLICATION RISK

Was the website designed to handle attack, or just good behaviour?

Some flaws can't be patched — they're built into the design. A password-reset flow with no rate limiting. A checkout that trusts the price the browser sends.

Questions to ask your developer:
1. *"What threat modelling did we do before building this? What does an attacker try?"* A blank look here means none was done.

2. *"What rate limits are in place on logins, password resets, account creation, contact forms?"*

3. *"Does the system trust prices, user IDs, or role flags the browser sends — or are these always re-checked server-side?"*

OWASP A05 · WEB APPLICATION RISK

Are your servers and tools properly locked down?

The most common source of breaches in the wild. Default passwords. Admin interfaces facing the public internet. Verbose error messages. Cloud storage buckets accidentally public.

Questions to ask your developer:
1. Verify yourself: Run Security Headers on your site. Aim for at least a B grade. The fixes are header changes that cost nothing.

2. Verify yourself: Try the obvious admin paths (`/admin`, `/wp-admin`, `/phpmyadmin`) from outside the office Wi-Fi. If they load a login page, ask why they're internet-facing.

3. *"Are we using Microsoft Defender for Cloud / AWS Trusted Advisor / Google SCC to flag misconfigurations? Show me last month's report."*

OWASP A06 · WEB APPLICATION RISK

Is your website built on software with known holes?

Most websites are built from many bits — CMS, plugins, libraries. If anything's out of date, the website inherits the holes. Most hacked WordPress sites are running old plugins, not zero-days.

Questions to ask your developer:
1. *"Can you produce an SBOM — a list of every third-party library we use and its current version?"* If they can't produce one, that's a sign nobody's tracking.

2. *"When did we last apply security updates to our CMS, plugins, and libraries? What's our patching cadence?"*

3. Decision: For WordPress sites, move to a managed WP host (WP Engine, Kinsta, SiteGround) that handles patching.

OWASP A07 · WEB APPLICATION RISK

Is your login system actually secure?

Bad login systems are why so many account takeovers happen. The classic mistakes: no rate limiting on failed logins, predictable password resets, sessions that never expire, MFA missing.

Questions to ask your developer:
1. *"Are we using a recognised authentication provider (Auth0, Microsoft Entra ID, AWS Cognito, Okta, Clerk) — or did we build the login system from scratch?"* Self-built auth is a red flag.

2. *"Is MFA required for any customer account that handles money or personal data? If not, can we make it the default for new accounts?"*

3. *"Walk me through our password rules — length, complexity, lockout, reset token expiry. How are passwords hashed?"*

OWASP A08 · WEB APPLICATION RISK

Can someone tamper with your software or data updates without you noticing?

How modern software updates itself. Auto-updates from untrusted sources or unverified library pulls can be poisoned — the SolarWinds incident was the famous example.

Questions to ask your developer:
1. *"Are third-party libraries pinned to specific versions, or do we auto-pull the latest? Pinned is safer."*

2. *"Who has the right to push code to production? Is there two-person review on production deploys?"*

3. *"Where do our software updates come from, and how do we verify they haven't been tampered with in transit?"*

OWASP A09 · WEB APPLICATION RISK

Would you know if your website was being attacked?

Time-to-detect for SME breaches is measured in weeks or months, mostly because nobody's watching. If your website doesn't log meaningful events, and nobody's reading the logs, an attacker has the run of the place until something visibly breaks.

Questions to ask your developer:

1. *"What security-relevant events do we log — authentication, admin actions, errors, bulk data exports? Where do those logs go?"*

2. *"Who reads them? On what cadence? When was the last alert anyone acted on?"*

3. *"Are logs stored off-server, so an attacker who gets in can't wipe them?"*

OWASP A10 · WEB APPLICATION RISK

Can someone use your website as a proxy to reach private systems?

Some websites let users provide a URL. If the website blindly fetches that URL, an attacker can provide an internal URL and pull private data. In cloud environments this often leads to full takeover via metadata-service abuse.

Questions to ask your developer:

1. *"Does our site fetch URLs that users provide (link previews, webhook configuration, 'import from URL')? If so, do we block requests to internal IP ranges?"*

2. *"On our cloud provider, is IMDSv2 (AWS) or its equivalent enabled to protect the metadata service?"*

3. *"Can we allowlist the external destinations our server is allowed to call?"*