Phishing-Resistant Web Authentication: Passkeys, QR Login, and the Patterns That Actually Work

Web Authentication / Last updated 2026-06-11 / https://www.scrambleid.com/learn/phishing-resistant-web-authentication

In one sentence: Phishing-resistant web authentication means an attacker cannot authenticate by relaying prompts or stealing OTPs, because proofs are **origin-bound** (WebAuthn) and/or **session-bound** (QR(DID) with binding).

TL;DR (canonical)

- "MFA" is not automatically phishing-resistant; OTP and push approvals can be relayed.
- WebAuthn/passkeys are phishing-resistant because the authenticator validates the relying party (origin/RP ID) and signs a fresh challenge.
- QR(DID) is phishing-resistant only when the QR is signed (QID), short-lived, and the success is bound to the initiating browser session.
- Production deployments commonly offer both: WebAuthn for same-device, QR(DID) for cross-device and shared terminals.

What does phishing-resistant authentication mean?

Phishing-resistant authentication is any method where an attacker who controls a fake login page cannot complete the ceremony on behalf of the real user. CISA's phishing-resistant MFA guidance names FIDO/WebAuthn and PKI-based MFA as the methods that qualify; the property they share is origin binding, where the credential is cryptographically tied to the legitimate domain. ScrambleID's position is that session binding with a separate device proof delivers the same resistance for cross-device flows.

> A method is phishing-resistant if a user cannot be tricked into producing a reusable proof that an attacker can replay from a different site or session.

The common failure pattern:

- attacker proxies the real login page (AiTM)

- user completes OTP/push
- attacker receives tokens/cookies and logs in

What are the two proven phishing-resistant primitives?

1) WebAuthn (passkeys)

Why it works:

- the authenticator checks the RP ID / origin context
- the user signs a fresh challenge
- there is no password/OTP to steal

Operationally, WebAuthn is strongest when you require **User Verification (UV)** for high-risk actions.

2) QR(DID) with session binding

QR(DID) is strongest when:

- the QR is a **signed QID** (tamper-evident)
- the DID TTL is short and single-use
- approval is delivered only to the initiating browser session
- the user types a short code (explicit intent) after scanning

This is how QR(DID) avoids "approve blind" anti-patterns.

> **Cryptographic detail:** how session binding actually works (the structure of the signed confirmation payload, the atomic verification check, per-channel binding mechanics, and the threat model coverage) is specified in Session binding cryptography in the architecture reference.

When should you use WebAuthn vs QR(DID)?

Environment	Prefer	Why
Laptop/phone same device	WebAuthn	best UX and strongest origin binding
Shared kiosk / clean room	QR(DID)	avoids local credentials; works with locked down browsers
Cross-device login	QR(DID)	phone approves; browser receives session-bound success
Mixed workforce devices	Both	reduces support burden and edge-case lockouts

What are the threat model pitfalls?

Pitfall: "QR screenshot relay" (quishing)

Attack: attacker shows a screenshot of a QR and convinces user to scan it.

Design fixes:

- short TTLs (seconds)
- require typed code shown on the screen
- show relying party/app name in the app confirmation UI

Pitfall: success delivered to the wrong browser session

Fix:

- bind DID/QID to the initiating websocket/session id
- only deliver success to that specific session

Pitfall: weak fallback becomes the real attack path

Fix:

- disable OTP and email links for high-risk apps/actions
- use step-up chains instead of "OTP as the escape hatch"

What should your deployment checklist include?

- Choose default: WebAuthn first (same-device), QR(DID) as cross-device fallback.
- Define max retries, lockouts, and clear "expired" UX.
- Decide policy: which apps/actions require UV or step-up.
- Instrument events: presented, confirmed, mismatch, timeout.
- Integrate with your IdP via SAML/OIDC.

Key Takeaway

Phishing-resistant web authentication requires origin binding (WebAuthn) or session binding (signed QR with DID). SMS OTP, email OTP, and push notifications without context are NOT phishing-resistant, they can be relayed through adversary-in-the-middle (AiTM) attacks. True phishing resistance means the authentication ceremony cannot be proxied through a fake site.

FAQ

What is phishing-resistant authentication?

Phishing-resistant authentication is a verification method that cannot be bypassed through fake websites, social engineering, or real-time credential relay attacks. It requires cryptographic binding between the authentication ceremony and either the origin (website URL) or the specific session. Methods that qualify include WebAuthn/passkeys (origin-bound) and signed QR codes with typed confirmation codes (session-bound).

What authentication methods are NOT phishing-resistant?

SMS OTP, email OTP, voice callback, time-based OTP (TOTP), and push notifications without additional context are NOT phishing-resistant. These can all be relayed through adversary-in-the-middle (AiTM) attacks where an attacker proxies the victim's credentials in real-time through a fake site.

What is an AiTM (adversary-in-the-middle) attack?

AiTM attacks use a reverse-proxy phishing site that sits between the victim and the legitimate site. When the victim enters credentials or approves an MFA prompt, the attacker captures and replays them instantly. AiTM toolkits like Evilginx make this attack accessible to low-skill attackers.

Are passkeys enough?

Often yes for same-device workforce journeys. QR(DID) is valuable for cross-device scenarios, shared terminals, and constrained browsers where passkeys may not work.

Is QR login safe?

Only if the QR is cryptographically signed, short-lived, and session-bound with a typed confirmation code. A QR that simply encodes a URL that can be forwarded is not phishing-resistant.

What about session theft after login?

Phishing-resistant login reduces takeover, but you should still secure session tokens (short TTLs, device binding where possible, monitoring for session anomalies).

References (public)

- CISA phishing-resistant MFA fact sheet: https://www.cisa.gov/sites/default/files/publications/fact-sheet-implementing-phishing-resistant-mfa-508c.pdf
- WebAuthn (W3C): https://www.w3.org/TR/webauthn/

Related reading

- SSO Integration Quickstart
- Dynamic Identifiers (DID/QID)
- XFactor Step-Up