Account Recovery Hardening Checklist

Companion resource for “Stopping Account Takeover at the Recovery Layer”

Core thesis

Account recovery is privileged access. If recovery is weaker than login, attackers route around your strongest controls. The purpose of this checklist is to help teams review recovery as a complete trust reissuance workflow, not as a single password reset form.

How to use this checklist

- Use it during design review, security review, or post-incident hardening.

- Walk the chain in order: request, delivery, verification, change, shutdown, notification, monitoring.

- Mark each item as Done, Risk Accepted, or Needs Work in your own tracking system.

- Prioritize shutdown and token lifecycle first if you are time-constrained.

Fast risk triage: five questions

Question	What to look for
Can a caller infer whether an account exists?	Content, timing, API shape, email side effects, or throttling may reveal it.
Can a recovery artifact be reused?	If yes, replay risk exists even if the artifact has not expired.
Can a reset link leak through normal plumbing?	URLs, referrers, logs, analytics, and link scanners are common leak paths.
Can a weaker channel reissue stronger trust?	MFA reset, support recovery, and email change are common bypass paths.
Can old authenticated state survive recovery?	If yes, the account may look recovered while attacker-held access remains alive.

1. Request: prevent account discovery

[ ] Use the same external message for known and unknown identifiers.
[ ] Avoid early-return behavior that creates obvious timing differences.
[ ] Return the same public API status and response shape for both cases.
[ ] Do not expose USER_NOT_FOUND, emailQueued, userId, maskedEmail, or similar account-state fields to the client.
[ ] Rate-limit by source signals as well as identifier, without making throttling confirm account existence.
[ ] Log attempts internally without logging secrets or creating client-visible differences.

2. Delivery: treat the recovery channel as a carrier of temporary authority

[ ] Send recovery artifacts only to verified recovery channels.
[ ] Do not reveal whether an email, SMS, or push message was actually delivered.
[ ] Avoid trusting user-controlled Host headers or unvalidated domains when constructing reset links.
[ ] Keep reset pages free from unnecessary third-party scripts that could receive referrer data.
[ ] Treat the full reset URL as sensitive because it can act as a bearer artifact.
[ ] Consider whether the channel is strong enough for the action being authorized.

3. Verification: constrain the recovery artifact lifecycle

[ ] Use high-entropy random tokens generated with a cryptographically secure random source.
[ ] Store only a derived or hashed form of live recovery tokens where practical.
[ ] Bind each artifact to one user and one action.
[ ] Set a short, risk-appropriate TTL and reject expired artifacts.
[ ] Consume the artifact immediately after successful use.
[ ] Reject replay attempts and supersede older artifacts when a newer recovery flow starts.
[ ] Account for bounded clock skew where short-lived codes are used, without creating large replay windows.

4. Change: treat credential and factor updates as trust transitions

[ ] Require reauthentication or step-up checks before sensitive account changes where appropriate.
[ ] Do not let password reset silently bypass enabled MFA.
[ ] Treat MFA reset, recovery email change, and recovery phone change as security-critical workflows.
[ ] Consider cooling-off periods before newly changed recovery channels gain full recovery authority.
[ ] Notify old and new channels when recovery destinations change.
[ ] Prevent a newly changed email or phone from immediately becoming the only trusted recovery route without assurance checks.

5. Shutdown: recover control, not just a password field

[ ] Invalidate the used recovery artifact.
[ ] Revoke existing browser sessions after high-risk recovery events unless a specific exception is justified.
[ ] Revoke refresh tokens and long-lived mobile sessions where applicable.
[ ] Expire remembered-device trust and trusted-browser flags where risk is high.
[ ] Require fresh authentication before sensitive follow-up actions after recovery.
[ ] Keep historical session records for audit, but mark revoked sessions as no longer valid.

6. Notification: make recovery visible to the real user

[ ] Send a clear notification when password, MFA, recovery email, or recovery phone changes.
[ ] Notify existing trusted channels for high-risk changes, not only the new channel.
[ ] Include useful context such as time, rough location, and device type where appropriate.
[ ] Provide a clear “this was not me” path.
[ ] Do not include raw tokens, reset URLs, or session secrets in notifications.

7. Monitoring: make recovery an operational surface

[ ] Track spikes in reset requests across accounts and from the same source.
[ ] Track repeated attempts against one account or one recovery channel.
[ ] Alert on reset followed by MFA reset, email change, password change, or high-value action.
[ ] Correlate recovery events with session continuity, geography shifts, and device changes.
[ ] Log structured events and outcomes, not bearer artifacts.
[ ] Create an incident-response path for suspected recovery abuse.

Production shutdown sequence

Use this sequence after a sensitive recovery event. The exact implementation depends on risk and product constraints, but the principle is fixed: recovery should recontain old trust, not only update a credential.

1. **Consume the recovery artifact:** Mark the reset token or code as used so it cannot be replayed.
2. **Change the credential or factor:** Apply the intended password, MFA, or recovery-channel change.
3. **Revoke old authenticated state:** Invalidate old browser sessions, refresh tokens, mobile sessions, and remembered-device trust where appropriate.
4. **Force fresh proof for sensitive actions:** Require reauthentication before account, payment, admin, or recovery-setting changes.
5. **Notify trusted channels:** Tell the user what changed and give them a “this was not me” path.
6. **Emit audit and monitoring events:** Keep evidence of what was changed, what was revoked, and what should be investigated.

Demo takeaway

If the password changed but the attacker session survived, the account was not meaningfully recovered. The product updated a field, but trust was not recontained.

What to avoid

- Do not treat “generic message” as the whole enumeration fix.

- Do not treat token randomness as enough without lifecycle constraints.

- Do not log raw reset links, raw tokens, or session secrets.

- Do not let email change, MFA reset, or support recovery become weaker alternative login paths.

- Do not call recovery complete while old attacker-held sessions remain valid.