Node.js Security Best Practices Summary

1. Quick Table: Attack → Protection

Attack / Problem	Protection / Mitigation
Compromised DB	Store passwords & reset tokens hashed (bcrypt, crypto).
Brute-force	bcrypt slows down checking, add rate limiting, max login attempts.
XSS	Sanitize input (xss-clean), Helmet headers, store JWT in httpOnly cookie.
CSRF	Use csurf package, SameSite cookies, or send JWT in headers.
DoS / Flooding	Rate limiting, limit body size, avoid evil regex.
NoSQL Injection	Mongoose schema validation, express-mongo-sanitize.
Token Theft	Store JWT in secure httpOnly cookie, not localStorage.
Token Revocation	Invalidate token on password change, use blacklist/Redis.
Parameter Pollution	hpp middleware, whitelist safe params.
Secrets Exposure	Never commit .env, use secret managers.
Reset Tokens	Random crypto tokens, store hashed, set expiry.
Refresh Tokens	Use short-lived access tokens + long-lived refresh tokens.
2FA	Second factor: SMS/Authenticator app.

2. Explanation of Each Point

Compromised DB
Passwords must always be hashed (bcrypt). Reset tokens must be hashed and stored with expiry.

Brute-force
bcrypt slows login attempts. Add express-rate-limit and limit failed attempts per user.

XSS
Sanitize input, use helmet headers, and never store JWT in localStorage. Use httpOnly cookie.

CSRF
Cross-Site Request Forgery can be prevented with csurf, SameSite cookies.

DoS / Flooding
Prevent massive requests with rate limiting, body size limit, and avoid vulnerable regex.

NoSQL Injection
Mongoose schemas + express-mongo-sanitize protect against injection queries.

Token Theft
httpOnly, secure cookies prevent JS from accessing JWT.

Token Revocation
Invalidate JWT if password changed. Use blacklist for malicious tokens.

Parameter Pollution
hpp middleware prevents duplicate query params causing unexpected logic.

Secrets Exposure
Keep secrets in .env, never commit to GitHub.

Reset Tokens
Generate crypto.randomBytes, store hashed token in DB with expiry, send plain token by email.

Refresh Tokens
Use refresh tokens to keep users logged in without long-lived access tokens.

2FA
Require SMS/Authenticator code for sensitive actions.