APPS BY VIKAS

Meet Vikas



GenzLudo

Ludo with 3D characters and up to 8 players. A first-of-its-kind no-teaming-up feature keeps play fair. Try it - you'll love the difference.

Download on Google Play



Kachu

Too much scrolling straining your neck? Add your addictive apps and complete daily neck exercises to unlock them.

Download on Google Play

01 / AUTHENTICATION, TOKENS & IDENTITIES

Advanced Website & App Security Checklist

20 critical exploit checks for AI-coded apps, with ready-to-use Claude prompts

By Vikas Yadav | November 2026

Use this list to audit your website, API, or app. Mark a check only after it has been verified in code and tested. No checklist or AI prompt can guarantee that a website will never be hacked.

- [ ] **1. Prevent 52-Character Username Truncation**

If a database column caps usernames at 50 characters without backend validation, registering "admin" padded with spaces gets trimmed to "admin", letting attackers impersonate accounts.

CLAUDE FIX: Add strict input validation before database queries to reject usernames longer than 30 characters instead of letting SQL truncate them, and trim whitespace.

- [ ] **2. Block Unicode Homograph Username Cloning**

Attackers swap Latin letters like "a" with lookalike Cyrillic "а" to create identical-looking usernames and scam users. Normalize strings and restrict allowed characters.

CLAUDE FIX: Normalize all usernames using Unicode NFKC normalization and enforce an ASCII-only alphanumeric regex (/^[a-zA-Z0-9_]+$/) during registration.

- [ ] **3. Stop Forged SAML Logins (XML Signature Wrapping)**

Apps sometimes verify a valid signature on one part of a SAML XML payload while reading login attributes from an unsigned section injected elsewhere.

CLAUDE FIX: Audit our SAML authentication handler so user attributes are extracted strictly from the exact signed assertion element verified by the library.

01 / AUTHENTICATION, TOKENS & IDENTITIES

Authentication, Tokens & Identities

- [ ] **4. Prevent Microsoft OAuth Email Spoofing**

Trusting mutable email claims in Microsoft OAuth/OIDC tokens lets organization admins alter emails to hijack accounts. Always link accounts using immutable tenant and object IDs.

CLAUDE FIX: Update Microsoft OAuth/OIDC login logic to identify and link users using the immutable combination of 'tid' (Tenant ID) and 'oid' (Object ID) instead of 'email'.

- [ ] **5. Block 2FA Brute-Force via GraphQL Aliases**

Attackers send 500 aliased verify2FA mutations inside a single HTTP POST request to bypass HTTP-level rate limiters and guess 6-digit codes.

CLAUDE FIX: Add GraphQL query alias and complexity limits, and enforce a strict per- user failed-attempt lockout directly inside the verify2FA resolver.

- [ ] **6. Prevent 2FA TOTP Code Replay**

A 6-digit authenticator code stays valid for 30 seconds. If you do not mark a used code as consumed, an attacker can replay the same code within that window.

CLAUDE FIX: Store each successfully used TOTP code per user in Redis/cache with a 60-second TTL, and reject any login attempt that reuses a recently consumed code.

02 / CRYPTOGRAPHY, SESSIONS & RANDOMNESS

Cryptography, Sessions & Randomness

- [ ] **7. Fix JWT Public Key Algorithm Confusion**

If your server trusts the "alg" header inside a JWT, an attacker can switch RS256 to HS256 and forge valid tokens signed using your public RSA key.

CLAUDE FIX: Pass an explicit allowed algorithms whitelist (e.g., algorithms: ['RS256']) to jwt.verify() everywhere so tokens cannot be downgraded to HS256.

- [ ] **8. Replace Predictable Math.random() Tokens**

Math.random() is not cryptographically secure. Attackers can predict password reset links, invite codes, or session IDs generated with it.

CLAUDE FIX: Replace every usage of Math.random() used for tokens, OTPs, session IDs, or invite codes with crypto.randomBytes() or crypto.randomUUID().

- [ ] **9. Eliminate Token Timing Attacks**

Comparing API keys or webhook secrets with standard "===" returns early on the first mismatched character, leaking secret characters via response timing.

CLAUDE FIX: Replace standard string equality checks (===) on API keys, webhook signatures, and secret tokens with constant-time crypto.timingSafeEqual().

- [ ] **10. Fix Cookie Padding Oracle Vulnerabilities**

Encrypting cookies with AES-CBC without an integrity tag leaks padding error differences, allowing attackers to decrypt and forge session cookies byte-by-byte.

CLAUDE FIX: Migrate custom AES-CBC cookie or payload encryption to authenticated encryption using AES-256-GCM with unique IVs and authentication tags.

03 / NETWORK, PROXIES & DATABASE ACCESS

Network, Proxies & Database Access

11. Mitigate HTTP/2 Rapid Reset DDoS

Opening and immediately cancelling HTTP/2 streams in rapid loops overwhelms backend workers unless stream concurrency and RST_STREAM rates are capped.

CLAUDE FIX: Configure strict HTTP/2 stream concurrency limits and RST_STREAM rate- limiting in our server and reverse-proxy configuration.

12. Block HTTP Request Smuggling

Disagreements between a front-end proxy and backend server over Content-Length and Transfer-Encoding headers let attackers smuggle hidden admin requests.

CLAUDE FIX: Configure our HTTP server and reverse proxy to reject requests containing both Content-Length and Transfer-Encoding headers.

13. Prevent Web Cache Poisoning via Headers

Reflecting unkeyed headers like X-Forwarded-Host inside HTML responses lets attackers poison CDN caches and serve malicious scripts to all visitors.

CLAUDE FIX: Remove all reflection of unkeyed headers like X-Forwarded-Host in responses, and generate redirect or asset URLs strictly from environment variables.

14. Stop Rate-Limit Bypass via Spoofed IP Headers

Reading client IPs directly from untrusted X-Forwarded-For headers lets attackers send fake IPs on every request to bypass rate limits.

CLAUDE FIX: Configure trusted proxy hops accurately or read client IPs strictly from verified edge headers like CF-Connecting-IP.

15. Enforce Row-Level Security (RLS) on SQL Views

In PostgreSQL and Supabase, standard Views execute with the view creator's permissions by default, bypassing Row-Level Security policies on underlying tables.

CLAUDE FIX: Audit all PostgreSQL/Supabase VIEW definitions and recreate them with 'WITH (security_invoker = true)' so they enforce the querying user's RLS policies.

04 / CODE EXECUTION, PARSERS & WEBSOCKETS

Code Execution, Parsers & Websockets

16. Prevent Server Freeze via Regex (ReDoS)

Regular expressions with nested quantifiers suffer catastrophic backtracking on crafted strings, locking the server CPU at 100%.

CLAUDE FIX: Refactor nested quantifiers into linear-time regex patterns and enforce maximum input string lengths before running regex checks.

17. Block Prototype Pollution to RCE

Recursively merging untrusted JSON containing "__proto__" or "constructor.prototype" corrupts global JavaScript objects and can lead to remote code execution.

CLAUDE FIX: Harden all deep-merge and object-assignment utilities by explicitly blocking '__proto__', 'constructor', and 'prototype' keys.

18. Prevent Server-Side Template Injection (SSTI)

Concatenating user input directly into template strings (EJS, Pug, Jinja) instead of passing it as data context allows attackers to execute arbitrary server code.

CLAUDE FIX: Ensure user input is never concatenated into template strings directly, and is always passed safely through the template data context object.

19. Disable XML External Entities (XXE)

Parsing uploaded XML files with DTD and external entity resolution enabled allows attackers to read local server files like /etc/passwd.

CLAUDE FIX: Configure all XML parsers in the application to disable DTD processing and external entity resolution completely.

20. Stop Cross-Site WebSocket Hijacking (CSWSH)

If WebSocket handshakes rely solely on browser cookies without verifying the Origin header, malicious sites can open authenticated WebSocket connections as the victim.

CLAUDE FIX: Validate the Origin header against allowed domains during WebSocket upgrades and require a short-lived one-time authentication ticket.

05 / COPY & USE

Final Instruction for Claude

Paste this after the 20 checks when asking Claude or Cursor to audit your codebase:

> Do not just tell me my website or app is secure. Check every one of these 20 security points carefully across my codebase and show me the exact vulnerabilities you found. Fix the problems cleanly without breaking existing features or changing how my app works. Test every fix and verify the security checks again. If something depends on cloud infrastructure and cannot be verified in code alone, clearly tell me. Do not guess and do not claim my app is 100% secure.

More things worth exploring

Find more free guides and developer checklists at vikasbuilds.com/resources