1. **If I give you an IP address, what would you do?**
   Hint: enumeration (ports/services), service/version fingerprinting, banner grab, web apps on non-standard ports, host header / virtualhost discovery, brute subdomains on associated names, check CDN / WAF, pivot plan.
2. **If I give you a domain, what would you do?**
   Hint: passive DNS, DNS enumeration, subdomain discovery (wordlists, cert transparency, brute), zone transfer check, host/endpoint fingerprinting, web asset crawling, identify infra (CDN, cloud provider), e-mail and owner records.
3. **What is your approach to subdomain takeover — manual and automated?**
   - Manual: identify dangling CNAMEs, verify DNS pointing to unclaimed cloud resources, attempt to claim resource (proof-of-concept), escalate impact.
   - Automation: run targeted checks (crt.sh + sublist3r + amass + assetfinder) → detect NXDOMAIN/CNAME to cloud providers → automated verification scripts to detect vulnerable providers.
   - **Include modern S3 / object-storage takeover**: detect misconfigured buckets (public, not created, or unclaimed), try bucket creation where allowed, check object listing, host static site takeover.
4. **On a single web page, what tests would you perform?** *(Avoid mentioning request-smuggling in the answer.)*
   Hint: input sources mapping (URL params, form bodies, cookies, headers, fragment/hash), DOM sinks & sources, reflected/stored XSS, CSRF tokens & behavior, client-side logic, file upload handling, auth state checks, file downloads, parameter pollution, insecure direct object references (IDOR), redirect/open redirect checks, content security policy, CORS, clickjacking.
5. **What is your approach for SQL Injection?**
   Hint: identify injectable input points, blind vs. error-based vs. UNION-based tests, time-based probes, fingerprint DB type/version, exploit path (data exfiltration, auth bypass), use taming techniques (hex/concat) and automated testing (sqlmap) with manual verification.
6. **What is your approach for Server-Side Template Injection (SSTI)?**
   Hint: map template entry points, inject template payloads for common engines (Jinja2, Twig, Velocity, etc.), test for expression evaluation, attempt sandbox escapes, enumerate classpath or access to internal APIs, chain to RCE if possible.
7. **Explain differences — based on file system permissions (read / write / execute):**
   - **Path traversal:** manipulating a path to access files outside intended directory (typically *read* impact — e.g., read /etc/passwd); may achieve write if vulnerable upload/parse exists.

- **Directory traversal:** same concept as path traversal; focus is traversing directory structure (read/write depending on context).
- **SSRF (Server-Side Request Forgery):** server makes network requests on attacker's behalf — can read internal services (remote read), hit metadata endpoints (sensitive tokens), or cause SSRF→RCE chains (write/execute only via discovered vulnerable services).
- **LFI (Local File Inclusion):** includes local files into execution context (usually *read* and code execution if you can include logs/php wrappers — can lead to execute if you can get a file written that contains code).

8. **If you find reflected XSS on a page and the request is `HttpOnly` (cookie flag), how would you increase impact?**
   Hint: pivot from HttpOnly protection: target session fixation, CSRF, login theft via redirect+phishing, use stored XSS paths, abuse other cookies/headers, hunt for other sensitive tokens in responses (bearer tokens, API keys), attempt to combine with clickjacking or social engineering to capture session via a secondary vector.
9. **What are cookie attributes and why they matter?**
   Hint: `HttpOnly`, `Secure`, `SameSite` (Lax/Strict/None), `Domain`, `Path`, `Expires/Max-Age`. Explain how each reduces theft, scope cookies, or allow cross-site use.
10. **What is a request header?**
    Hint: metadata sent with HTTP requests (Host, User-Agent, Referer, Cookie, Authorization, X-Forwarded-For, Content-Type, Accept, etc.) — used for routing, auth, caching, fingerprinting, and often abused for injections or bypasses.
11. **What is `SameSite` for cookies, and what are attack approaches?**
    Hint: explains `SameSite=Strict|Lax|None`, how it limits cross-site cookie sending. Attack approaches: CSRF when SameSite lax/none, exploit link-click behaviors, use top-level navigations, or find other cookies without SameSite to escalate.
12. **What is request smuggling?**
    Hint: class of desync attacks caused by inconsistent request parsing between front-end and back-end (e.g., different CL/TE handling) leading to request splitting, bypasses, or cache poisoning. (You asked to avoid naming it elsewhere — keep definition concise.)
13. **What is your approach to DOM XSS?**
    Hint: identify client-side sinks, map JS sources (location, document.write, innerHTML, eval, setAttribute), test with inert payloads, then real payloads, check CSP and sanitizers, and attempt to locate persistent flows (stored via client storage).
14. **What are symmetric & asymmetric keys?**
    Hint: Symmetric: same key to encrypt/decrypt (fast, needs secure key exchange). Asymmetric: public/private key pairs (RSA, ECC) — public encrypt, private decrypt or sign, used for signatures and key exchange.

15. **What is TLS handshaking?**
    Hint: client hello → server hello → certificate exchange → key exchange (e.g., ECDHE) → verify → derive shared keys → secure channel. Mention cert validation, cipher negotiation, and session resumption basics.
16. **What is your automation approach?**
    Hint: combine discovery tools (amass, assetfinder, crt.sh), scanning (nmap, masscan), active enumeration (gobuster, ffuf), vulnerability scanners (nuclei), custom scripts for logic flaws, and orchestration (bash/python pipelines, GH actions). Emphasize human-in-the-loop for verification.
17. **What is your learning approach?**
    Hint: goal-driven learning — pick a weakness category, read recent write-ups, reproduce PoCs in local labs, build tiny testbeds, write notes, and teach back (blog/write-up) to solidify.
18. **How do you keep yourself up to date daily?**
    Hint: follow security Twitter/X researchers, HackerOne/Bugcrowd reports, RSS/feeds (nuclei-templates updates), changelogs for major frameworks, mailing lists, and curated newsletters. Automate feeds where possible.
19. **How do you structure bug hunts?**
    Hint: prioritize high-value assets, scope mapping, automated discovery, manual review on high-priority targets, escalate findings with PoC + remediation steps, and track time spent vs. expected ROI.
20. **What is your approach to XSS (general/testing & exploitation)?**
    Hint: identify all injection points, test reflected/stored/DOM variants, test bypasses (HTML/JS contexts), check CSP, chains (auth CSRF, cookie theft, token retrieval), demonstrate impact with contextual PoC, and include recommended fixes.