Zoho Practice Questions 1

🔹 Java — 30 Code Review Questions

---

1. SQL Injection

```
String sql = "SELECT * FROM users WHERE id=" + request.getParameter("id");
ResultSet rs = stmt.executeQuery(sql);
```

👉 Bug: SQLi.
✅ Fix: Use `PreparedStatement`.

---

2. XSS in JSP

```
<%= request.getParameter("msg") %>
```

👉 Bug: Reflected XSS.
✅ Fix: Escape using `<c:out>` or OWASP Java Encoder.

---

3. Hardcoded credentials

```
String dbUser = "admin";
String dbPass = "password123";
```

👉 Bug: Secrets in code.
✅ Fix: Use environment variables or secret manager.

---

4. Insecure password hashing

```
MessageDigest md = MessageDigest.getInstance("MD5");
```

👉 Bug: Weak hash.
✅ Fix: Use bcrypt/Argon2.

---

5. Broken Auth via Client Param

```
if (request.getParameter("role").equals("admin")) {
   showAdmin();
}
```

👉 Bug: Trusts client input.
✅ Fix: Enforce server-side RBAC.

---

6. Path Traversal

```
File f = new File("/app/data/" + request.getParameter("file"));
```

👉 Bug: `../etc/passwd`.
✅ Fix: Canonicalize & whitelist.

---

7. Insecure Deserialization

```
ObjectInputStream in = new ObjectInputStream(req.getInputStream());
Object o = in.readObject();
```

👉 Bug: RCE via deserialization.
✅ Fix: Use JSON; allowlist types.

---

8. Debug Enabled

```
System.out.println("DEBUG: " + error);
```

👉 Bug: Info disclosure.
✅ Fix: Use logging framework at proper level.

---

9. Session Fixation

```
// login doesn't invalidate old session
```

👉 Bug: Reuse old session.
✅ Fix: `request.changeSessionId()`.

---

10. Sensitive data in URL

```
response.sendRedirect("/reset?token=" + token);
```

👉 Bug: Token in URL leaks.
✅ Fix: Use POST or HttpOnly cookie.

---

11. Insecure Cookie

```
Cookie c = new Cookie("sid", sessionId);
response.addCookie(c);
```

👉 Bug: Missing HttpOnly/Secure.
✅ Fix: Set flags.

---

12. Insufficient Transport Security

```
URL url = new URL("http://api.example.com/data");
```

👉 Bug: HTTP not HTTPS.
✅ Fix: Use HTTPS + cert validation.

---

13. Open Redirect

```
String target = request.getParameter("url");
response.sendRedirect(target);
```

👉 Bug: Open redirect.
✅ Fix: Whitelist domains.

---

14. Weak Random

```
int otp = new Random().nextInt(999999);
```

👉 Bug: Predictable.
✅ Fix: Use `SecureRandom`.

---

15. XML External Entity (XXE)

```
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
dbf.setExpandEntityReferences(true);
```

👉 Bug: XXE.
✅ Fix: Disable DTDs.

---

16. Command Injection

```
Runtime.getRuntime().exec("ping " + ip);
```

👉 Bug: Injection.
✅ Fix: Validate or use ProcessBuilder args.

---

17. Stack Trace Disclosure

```
e.printStackTrace(out);
```

👉 Bug: Leaks internals.
✅ Fix: Log server-side, generic error message.

---

18. Improper Logging of Secrets

```
logger.info("Password entered: " + pwd);
```

👉 Bug: Logs secrets.
✅ Fix: Mask sensitive data.

---

19. Unrestricted File Upload

```
filePart.write("/uploads/" + fileName);
```

👉 Bug: Web shell risk.
✅ Fix: Validate file type, random name, scan.

---

20. Missing CSRF Protection (Spring MVC)

```
<form action="/transfer" method="POST"> … </form>
```

👉 Bug: No CSRF token.
✅ Fix: Enable Spring Security CSRF.

---

21. Insecure Session Timeout

```
session.setMaxInactiveInterval(86400);
```

👉 Bug: Too long.
✅ Fix: Use short expiry (15–30 min).

---

22. Clickjacking

No `X-Frame-Options`.
👉 Bug: UI redress attack.
✅ Fix: Add `DENY` / `SAMEORIGIN`.

---

23. Insufficient Input Validation

```
int age = Integer.parseInt(request.getParameter("age"));
```

👉 Bug: May crash / injection.
✅ Fix: Validate numeric range.

---

24. Improper Resource Exposure

```
response.getWriter().write(new FileReader("/etc/passwd").read());
```

👉 Bug: Sensitive data exposure.
✅ Fix: Restrict access.

---

25. Race Condition

```
if (balance >= amount) {
   balance -= amount;
}
```

👉 Bug: Race on concurrent requests.
✅ Fix: Use synchronization/transactions.

---

26. Excessive Data Exposure

```
return gson.toJson(user);
```

👉 Bug: Leaks all fields.
✅ Fix: Return only necessary fields (DTO).

---

27. Missing Rate Limiting
Login endpoint no throttling.
👉 Bug: Brute force.
✅ Fix: Rate limit & lockout.

---

28. Weak JWT Secret

```
String secret = "12345";
```

👉 Bug: Guessable.
✅ Fix: 256-bit random key.

---

29. Directory Listing
Default Tomcat allows browsing `/webapps`.
👉 Bug: Info exposure.
✅ Fix: Disable directory listing.

---

30. Dependency Vulnerability
Using Log4j 2.14 (Log4Shell).
👉 Bug: RCE.
✅ Fix: Update to patched version.

🔹 Python — 30 Code Review Questions

---

1. SQL Injection

```
cur.execute("SELECT * FROM users WHERE id = %s" % uid)
```

👉 Bug: SQLi.
✅ Fix: Use parameterized query.

---

2. Command Injection

```
os.system("ping " + ip)
```

👉 Bug: Injection.
✅ Fix: `subprocess.run([“ping”, ip])`.

---

3. Pickle Deserialization

```
obj = pickle.loads(data)
```

👉 Bug: RCE.
✅ Fix: Use JSON.

---

4. Debug Mode

```
app.run(debug=True)
```

👉 Bug: Dangerous in prod.
✅ Fix: Debug = False.

---

5. Hardcoded Secrets

```
API_KEY = "abcd1234"
```

👉 Bug: Secret in code.
✅ Fix: Env variables.

---

6. Weak Password Hashing

```
hashlib.sha1(pwd.encode()).hexdigest()
```

👉 Bug: Weak.
✅ Fix: bcrypt/argon2.

---

7. Path Traversal

```
open("/data/" + filename)
```

👉 Bug: `../`.
✅ Fix: Sanitize + whitelist.

---

8. XSS in Flask

```
return f"Hello {request.args['name']}"
```

👉 Bug: XSS.
✅ Fix: Escape output.

---

9. No CSRF Token
Flask form POST no CSRF.
👉 Bug: CSRF.
✅ Fix: Flask-WTF CSRFProtect.

---

10. Insecure Cookie

```
resp.set_cookie("sid", sid)
```

👉 Bug: No HttpOnly/Secure.
✅ Fix: Add flags.

---

11. Missing Rate Limiting
Login route no throttling.
👉 Bug: Brute force.
✅ Fix: Flask-Limiter.

---

12. Info Disclosure in Error

```
raise e
```

👉 Bug: Stacktrace leak.
✅ Fix: Custom error page.

---

13. Directory Listing
Static files served directly.
👉 Bug: Info exposure.
✅ Fix: Disable autoindex.

---

14. Weak Random

```
otp = random.randint(100000, 999999)
```

👉 Bug: Predictable.
✅ Fix: `secrets.randbelow`.

15. SSRF

```
requests.get(request.args["url"])
```

👉 Bug: SSRF.
✅ Fix: Whitelist domains.

---

16. Log Injection

```
logger.info(request.args["msg"])
```

👉 Bug: Attacker logs.
✅ Fix: Sanitize input.

---

17. YAML Deserialization

```
yaml.load(data)
```

👉 Bug: RCE.
✅ Fix: `yaml.safe_load`.

---

18. JSON Injection

```
return jsonify({"user": request.args["name"]})
```

👉 Bug: If unescaped → injection.
✅ Fix: Validate, sanitize.

---

19. Excessive Data Exposure

```
return jsonify(user.__dict__)
```

👉 Bug: Sends passwords.
✅ Fix: Filter fields.

---

20. Improper Input Validation

```
age = int(request.args["age"])
```

👉 Bug: Crash / injection.
✅ Fix: Validate.

---

21. No Session Expiry

Flask default permanent.
👉 Bug: Long-lived cookie.
✅ Fix: Short expiry.

---

22. Open Redirect

```
return redirect(request.args["next"])
```

👉 Bug: Open redirect.
✅ Fix: Whitelist.

---

23. CORS Misconfig

```
CORS(app, resources={r"*": {"origins": "*"}})
```

👉 Bug: Wildcard.
✅ Fix: Restrict origins.

---

24. Missing Content Security Policy

👉 Bug: XSS risk.
✅ Fix: Add CSP headers.

---

25. File Upload

```
f.save("/uploads/" + f.filename)
```

👉 Bug: RCE.
✅ Fix: Validate type.

---

26. Logging Secrets

```
logger.info("Password: " + pwd)
```

👉 Bug: Secret log.
✅ Fix: Mask.

---

27. JWT None Algorithm

```
jwt.decode(token, verify=False)
```

👉 Bug: Forged tokens.
✅ Fix: Always verify + strong secret.

---

28. Path Injection

```
shutil.copy(src, dest)
```

👉 Bug: If dest = `../../etc/passwd`.
✅ Fix: Sanitize path.

---

29. Memory DoS

```
json.loads(huge_input)
```

👉 Bug: DoS.
✅ Fix: Limit input size.

---

30. Dependency Vulnerability

Using Django 2.0 (EOL).
👉 Bug: CVEs.
✅ Fix: Update to LTS.

🔹 C / C++ — 30 Code Review Questions

1. Buffer Overflow

```
char buf[16];
gets(buf);
```

👉 Bug: Overflow.
✅ Fix: `fgets`.

---

2. Format String

```
printf(user_input);
```

👉 Bug: Injection.
✅ Fix: `printf("%s", user_input)`.

---

3. Command Injection

```
system("ls " + user_input);
```

👉 Bug: Injection.
✅ Fix: Whitelist args.

---

4. Integer Overflow

```
int size = length * count;
malloc(size);
```

👉 Bug: Overflow.
✅ Fix: Check multiplication.

---

5. Use After Free

```
free(p);
*p = 10;
```

👉 Bug: UAF.
✅ Fix: Null pointer after free.

---

6. Double Free

```
free(p);
free(p);
```

👉 Bug: Corruption.
✅ Fix: Nullify after free.

---

7. Race Condition

```
if(access("file", W_OK) == 0) { open("file", O_WRONLY); }
```

👉 Bug: TOCTOU.
✅ Fix: Open securely with flags.

---

8. Path Traversal

```
open("/data/" + filename, O_RDONLY);
```

👉 Bug: `../`.
✅ Fix: Canonicalize.

---

9. Hardcoded Credentials

```
char *pwd = "root123";
```

👉 Bug: In code.
✅ Fix: Secure storage.

---

10. Weak Random

```
srand(time(NULL));
rand();
```

👉 Bug: Predictable.
✅ Fix: Use `/dev/urandom`.

---

11. Missing Bounds Check

```
strcpy(buf, user_input);
```

👉 Bug: Overflow.
✅ Fix: `strncpy`.

---

12. Memory Leak

```
malloc(100); // no free
```

👉 Bug: Leak.
✅ Fix: Free memory.

---

13. Insecure Temp File

```
fopen("/tmp/file.txt", "w");
```

👉 Bug: Symlink attack.
✅ Fix: `mkstemp`.

---

14. Logging Secrets

```
printf("Password: %s", pwd);
```

👉 Bug: Exposes.
✅ Fix: Mask.

---

15. Error Handling

```
if(open("file", O_RDONLY)) { … }
```

👉 Bug: Ignoring errno.
✅ Fix: Check properly.

---

16. Outdated SSL

```
SSLv2_client_method();
```

👉 Bug: Weak.
✅ Fix: TLS 1.2+.

---

17. Improper Pointer Arithmetic

```
char arr[10];
arr[20] = 'a';
```

👉 Bug: OOB write.
✅ Fix: Validate index.

---

18. Insecure Deserialization (C++)

```
ifstream f("data.ser");
f >> obj;
```

👉 Bug: No validation.
✅ Fix: Validate schema.

---

19. Race Condition (fork)

Shared resource no lock.
👉 Bug: Race.
✅ Fix: Mutex.

---

20. Integer Truncation

```
short s = bigInt;
```

👉 Bug: Data loss.
✅ Fix: Check before cast.

---

21. Use of Deprecated Function

```
gets(buf);
```

👉 Bug: Unsafe.
✅ Fix: Remove.

---

22. No ASLR / DEP

👉 Bug: Build insecure.
✅ Fix: Compiler flags.

---

23. Buffer Under-read

```
char *p = buf - 1;
```

👉 Bug: Underflow.
✅ Fix: Pointer bounds.

---

24. Improper File Permissions

```
open("file", O_CREAT, 0777);
```

👉 Bug: Too permissive.
✅ Fix: 0600.

---

25. Format String in Syslog

```
syslog(LOG_ERR, user_input);
```

👉 Bug: Injection.
✅ Fix: `%s`.

---

26. Stack Exhaustion

```
char big[1000000];
```

👉 Bug: Crash.
✅ Fix: Heap allocation.

---

27. Unchecked Return Value

```
read(fd, buf, 10);
```

👉 Bug: Ignore errors.
✅ Fix: Check return.

---

28. Insecure Cryptography

```
MD5(data, len, out);
```

👉 Bug: Weak.
✅ Fix: SHA-256+ or Argon2.

---

29. NULL Dereference

```
char *p = NULL; *p = 'a';
```

👉 Bug: Crash.
✅ Fix: Null check.

---

30. Dependency Vulnerability

Old OpenSSL version.

👉 Bug: Heartbleed.

✅ Fix: Update library.
