SQL Injection – Controls & Checklist

Legend
■ **Attack**: Attacker path and risky components
■ **Prevent**: Preventive controls (WAF, prepared statements, IAM, SG/NSG)
■ **Detect/Respond**: Logging, SIEM, SOAR, incident actions

Controls Checklist

Control Area	Insecure Baseline	Hardened Target
Input handling	■ String concatenation (dynamic SQL)	■ Prepared statements / ORM
WAF rules	■ None / default only	■ SQLi managed rules, custom rules, bot mgmt
Secrets mgmt	■ Credentials in env/code	■ Secrets Manager + KMS, rotation policy
IAM to DB	■ Broad static user/pass	■ IAM auth / least privilege role
Network	■ DB open / shared SG	■ Private subnet, SG allow-list from App only
Logging	■ App-only/no audit	■ WAF + App + DB audit → central SIEM
Response	■ Manual reaction	■ SOAR playbooks (block IP, rotate keys, rollback)

Interview Insight
If asked in an interview how to mitigate SQL Injection, walk through your architecture diagram:

- Start with **WAF**: blocks known payloads at the edge.
- Move to **App Tier**: explain use of prepared statements/ORM and input validation.
- Highlight **IAM + Secrets**: least privilege DB role, rotated credentials.
- Point out **Network isolation**: DB only accessible from App subnet/SG.
- Close with **Logging & Response**: WAF + App + DB logs feed SIEM, which triggers SOAR to block attacker IP and rotate secrets automatically.

End with: “Defense-in-depth ensures that even if one control fails, others reduce risk and provide visibility.”