Question: You are a security operations system analyst and helpful cybersecurity assistant that generates detailed incident response playbooks based on NIST guidelines. please generate a playbook for a SQL Injection attacks effectively?

Answer:

Title: SQL Injection Attack Playbook

1. Preparation:

Objective: Establish contacts, define procedures, and gather information to save time during an incident.

- \*\*Security Controls\*\*:

- Implement input validation and sanitization for web applications.

- Use prepared statements or parameterized queries for database interactions.

- Apply Web Application Firewall (WAF) rules to detect and block injection patterns.

- \*\*Asset Inventory\*\*:

- Maintain an up-to-date list of all databases and web applications, including endpoints handling user inputs.

- \*\*Access Controls\*\*:

- Ensure least privilege access for database users.

- Regularly review and rotate database credentials.

- \*\*Monitoring Tools\*\*:

- Configure SIEM tools to monitor anomalous database logs and unauthorized changes to data.

- Set up alerts for suspicious URL patterns or payloads detected in HTTP traffic.

- \*\*Incident Drills\*\*:

- Simulate SQL injection scenarios to test detection and response procedures.

2. Detect:

Objective: Detect the incident, determine its scope, and involve appropriate parties.

- \*\*Identify Threat Indicators\*\*:

- \*\*Alerts\*\*:

- Detect common SQL injection payloads (e.g., `' OR '1'='1` or `UNION SELECT`) via WAF.

- Monitor unusual database queries, such as `DROP TABLE` commands, mass exfiltration of sensitive data, or queries from unauthorized IPs via SIEM.

- \*\*Logs\*\*:

- Review web server logs for unusual query strings in GET/POST requests.

- Analyze database logs for queries executed outside normal application patterns or login attempts using non-application accounts.

- \*\*Identify Risk Factors\*\*:

- \*\*Common Risks\*\*:

- Data theft or unauthorized modifications.

- Database corruption or deletion.

- Privilege escalation in the database.

- \*\*Company-Specific Risks\*\*:

- Compromise of sensitive customer data (e.g., PII).

- Breach of compliance standards (e.g., GDPR, HIPAA).

- \*\*Data Collection\*\*:

- Examine malicious inputs submitted via web forms, URLs, or APIs.

- Review WAF logs for blocked or allowed suspicious requests.

- Correlate suspicious input with database queries executed during the same timeframe.

- \*\*Categorize\*\*:

- \*\*Injection Types\*\*:

- Error-Based SQL Injection (visible database errors).

- Union-Based SQL Injection (data exfiltration via `UNION SELECT`).

- Blind SQL Injection (true/false payloads without direct error feedback).

- \*\*Is it an Advanced Attack?\*\*:

- Indicators include:

- Use of out-of-band (OOB) channels to exfiltrate data (e.g., DNS).

- Automated tools like SQLmap detected in logs.

- Cross-database queries targeting linked servers.

- \*\*Triage\*\*:

- \*\*Assess Impact\*\*:

- Determine the severity of data accessed or modified.

- Check whether the attacker achieved code execution or database control.

- \*\*Prioritization\*\*:

- High priority if sensitive data was accessed or database availability is at risk.

- \*\*False Positive Assessment\*\*:

- Cross-check suspicious activity with developers and confirm anomalies in database or web server logs.

- Document and close the case if it is a false positive.

3. Analyze:

Objective: Verify the attack, evaluate its scope, and correlate findings with threat intelligence.

- \*\*Verify\*\*:

- Reproduce malicious payloads in a secure test environment.

- Check application responses and database logs for confirmation.

- \*\*Identify IOCs\*\*:

- Exploited input fields (e.g., login forms, search boxes).

- Suspicious query patterns in database logs.

- \*\*Extract IOCs\*\*:

- Document malicious payloads, IP addresses, and timestamps.

- \*\*Submit to Partners\*\*:

- Share findings with WAF vendors or database administrators.

- \*\*Scan Enterprise\*\*:

- Perform web application vulnerability scans to identify other SQL injection risks.

4. Contain/Eradicate:

Objective: Mitigate the attack’s effects and remove the threat.

- \*\*Contain the Threat\*\*:

- Block offending IPs at the firewall or WAF.

- Temporarily disable compromised application endpoints.

- \*\*Eradicate the Root Cause\*\*:

- Patch the vulnerable web application.

- Implement stricter input validation and sanitization.

- Update WAF rules to block similar payloads.

- \*\*Validate\*\*:

- Test the application to confirm SQL injection is no longer exploitable.

5. Recover:

Objective: Restore systems and applications to normal operations.

- \*\*Restore Operations\*\*:

- Restore corrupted or deleted data from backups.

- Reactivate application endpoints after thorough testing.

6. Lessons Learned:

Objective: Document the incident’s details, discuss lessons learned, and adjust plans and defenses.

- \*\*Post-Incident Review\*\*:

- Review the attack vector and determine how it was missed during development or testing.

- \*\*Strengthen Security\*\*:

- Adopt secure coding practices.

- Update the playbook and train developers and analysts to handle similar incidents.