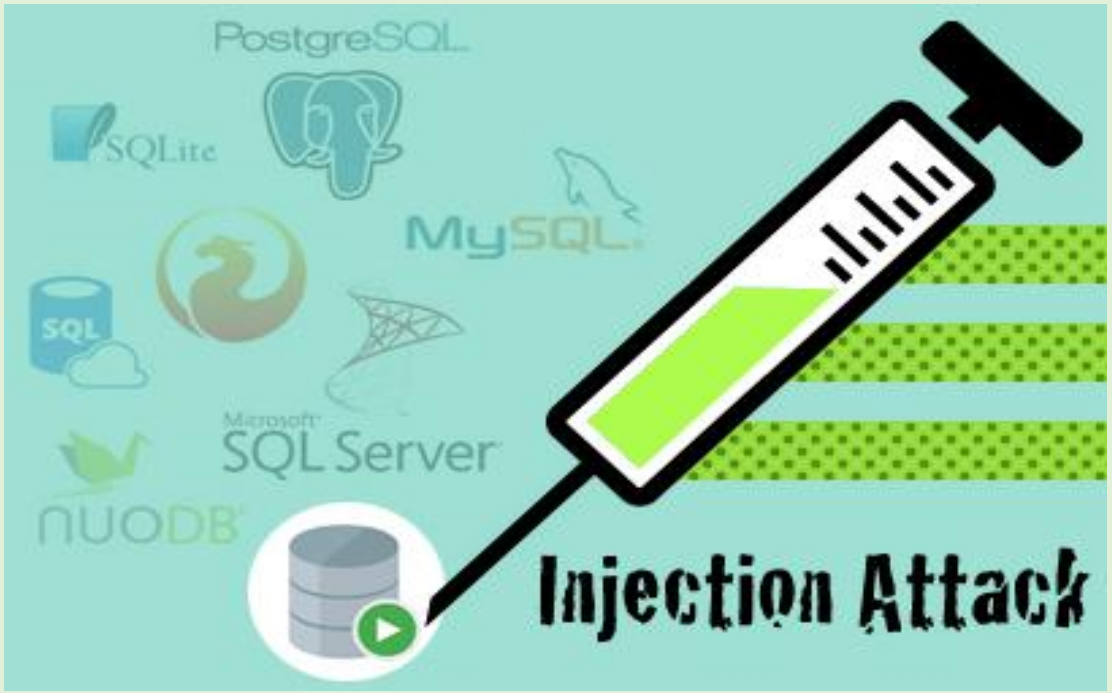
RakshaSutraX Incident Response

SQL Injection Playbook v1.0



Document Control

Title	SQL injection Playbook
Version	1.0
Date Issued	0112/2024
Status	Draft
Document owner	RaptorX7
Creator name	RaptorX7
Creator organisation name	RakshaShutraX
Subject category	Cyber Incident Response Management
Access constraints	

Document Revision History

Version	Date	Author	Summary of changes
1.0	01/012/2024	RaptorX7	Generic Version Created from Public Sector Playbook

Contents

1. Introduction

1.1. Overview

SQL Injection (SQLi) is one of the most prevalent and dangerous types of cyberattacks that target databases by manipulating the SQL queries executed by an application. By injecting malicious SQL code into input fields or URL parameters, attackers can compromise database operations, leading to unauthorized data access, data modification, data deletion, and even complete administrative control of the database. The consequences of successful SQLi attacks can include severe data breaches, financial losses, and reputational damage. Notable examples include the 2014 Sony Pictures attack, which exposed sensitive information, and the 2009 Heartland Payment Systems breach, which compromised millions of credit card details. Understanding SQLi's risk and preventive strategies is crucial for any organization.

1.2. Purpose

The purpose of this playbook is to provide a comprehensive guide for detecting, analyzing, containing, eradicating, and recovering from SQLi incidents. It serves as a step-by-step approach for incident response teams to effectively manage SQLi events and ensure that preventive measures are in place to safeguard against future occurrences.

1.3. SQL Injection Definition

SQL Injection (SQLi) is a security vulnerability that occurs when an attacker inserts or manipulates malicious SQL statements into input fields or application interfaces, compromising database queries. This can result in unauthorized data access, data leakage, database manipulation, and even full system compromise.

1.4. Scope

This playbook is intended for use by all members of the incident response team, including database administrators, IT security analysts, system administrators, and application developers. It should be used as a standard reference for preventing and responding to SQLi incidents. The playbook applies to all systems handling organizational data and should be integrated with the broader Cyber Incident Response Plan (CIRP).

1.5. Review Cycle

The playbook will be reviewed annually by the Cyber Incident Response Team (CIRT) lead and after any major security incidents. Reviews will also be conducted whenever new vulnerabilities, database technologies, or third-party services are integrated into the infrastructure.

2. Preparation Phase

Phase Objectives

The preparation phase is crucial for setting up a resilient defense system to handle SQLi threats. The objective is to build strong preventive and detection mechanisms, educate the workforce, and simulate attack scenarios to ensure readiness.

Key activities:

- Proactive deployment of advanced database monitoring and defense tools.
- Ongoing education for employees on identifying and reporting suspicious activities.
- Periodic security assessments and attack simulations to test defenses.

Technical Steps:

- **Tool Deployment**: Ensure the installation and configuration of powerful database monitoring tools (e.g., SQL Server Management Studio, Oracle Enterprise Manager) and specialized software like SolarWinds Database Performance Analyzer. These tools should continuously track database performance and flag anomalies that could indicate SQLi attempts.
- **IDS Configuration**: Optimize Intrusion Detection Systems (IDS) such as Snort or Suricata to identify SQLi-specific attack signatures and patterns. Implement custom rule sets and update regularly to cover new attack vectors.
- **SIEM Integration**: Enhance threat intelligence by incorporating feeds into SIEM solutions like Splunk, LogRhythm, or IBM QRadar. This integration should facilitate real-time event correlation and alerting for potential SQLi attempts.
- **Employee Training**: Initiate interactive workshops, hands-on phishing simulations, and courses on the importance of data sanitization practices and SQLi reporting. Regularly update training material to include new attack techniques and defensive measures.
- **Patch Management**: Institute a schedule for routine application of patches and updates to database management systems and related software to close vulnerabilities that could be exploited by SQLi attackers.

Advanced Preparation Measures

- **Network Segmentation**: Divide the network into secure zones and restrict communication between zones to contain any potential SQLi attack.

- **Database Backup Strategy**: Implement automated, encrypted database backups with offsite storage to ensure quick recovery in the event of a compromise.
- **Vulnerability Scanning**: Utilize vulnerability scanning tools (e.g., Nessus, OpenVAS) to identify weak points in the infrastructure and remediate them promptly.
- **Access Controls**: Strengthen user access management by enforcing multi-factor authentication (MFA) for administrative accounts and limiting user privileges to the minimum necessary for their tasks.
- **Enhanced Logging**: Establish comprehensive logging of all database activities with detailed audit trails for future analysis. Implement centralized log storage and ensure logs are tamper-proof.

Example Preparation Action Plan

1. **Deploy Monitoring Tools**:
    - Identify critical databases and install monitoring software.
    - Configure alerts for known SQLi attack patterns (e.g., 'DROP TABLE', 'UNION SELECT').
2. **Establish IDS and SIEM System**:
    - Implement IDS with custom rules tailored for SQLi detection.
    - Integrate with SIEM for automated data correlation.
3. **Run Employee Workshops**:
    - Schedule monthly training sessions on recognizing SQLi attempts and secure coding practices.
4. **Apply Security Patches**:
    - Verify patch management procedures and ensure application vulnerabilities are updated monthly.

By taking these proactive steps, organizations can establish a formidable defense against SQLi attacks, minimize risk, and prepare to respond swiftly should an incident occur.

3. Detection

3. Detection Phase

3.1. Detection Overview

Detection is the cornerstone of an effective incident response plan, providing real-time visibility into activities that may indicate an SQLi attack. A thorough detection strategy combines log analysis, behavioral analysis, and the deployment of specialized tools to monitor for anomalies.

3.2. Key Detection Techniques

3.2.1. Log Analysis

Detailed Log Parsing: Analyze server logs, application logs, and database logs for suspicious SQL queries, such as those containing unusual or unexpected SQL keywords.

Anomaly Detection: Look for deviations from normal user patterns, including high-frequency query execution or requests with known SQLi payloads.

Error Message Analysis: Monitor for detailed error messages that could indicate the presence of SQL injection attempts (e.g., SQL syntax errors, connection failures).

3.2.2. Behavioral Analysis

UEBA Systems: Deploy systems that analyze user and entity behaviors to detect deviations from established baselines. Unusual patterns like a sudden spike in database access or activity from non-typical locations could indicate an SQLi attempt.

User Interaction Monitoring: Track user interactions across the application to identify suspicious behavior, such as attempts to inject SQL commands through input fields or URL parameters.

3.2.3. Web Application Firewall (WAF)

Advanced WAF Configuration: Ensure the WAF is configured to inspect HTTP/HTTPS traffic for signs of SQL injection, such as suspicious payloads, non-standard SQL commands, and known attack patterns.

Signature-Based Detection: Update WAF signatures regularly to detect known SQLi payloads and attack vectors, ensuring the firewall can block these at the earliest point of entry.

Rate-Limiting and IP Blacklisting: Implement rate-limiting to prevent excessive database queries from a single IP, and blacklist IPs that show suspicious or malicious activity.

3.2.4. Real-Time Alerts

Alert Thresholds: Set up alert thresholds for abnormal activities, such as:

An unexpected increase in database connection attempts.

Queries with SQL commands that differ from typical usage patterns.

Unauthenticated database access attempts.

Integration with SIEM: Ensure real-time alerts are integrated with your SIEM tool for correlation and immediate analysis. Alerts should trigger automated responses or escalate to a human analyst for investigation.

3.3. Technical Indicators to Watch For

3.3.1. SQL Keywords

Indicators: Unusual or unauthorized use of keywords like SELECT, INSERT, UPDATE, DELETE, UNION, DROP, and EXEC.

3.3.2. Abnormal User-Agent Strings

Indicators: Requests containing suspicious user-agent strings, such as those indicative of automated scanning tools.

3.3.3. Unusual IP Addresses

Indicators: Monitoring for access attempts from IPs that have not previously interacted with the system or from known malicious IPs.

3.3.4. Error Codes

Indicators: Database error messages such as SQL syntax error, table not found, or column not found, which could point to probing attempts.

3.3.5. Verbose Error Messages

Indicators: Detailed error responses that may reveal underlying database structure or expose vulnerable endpoints.

3.4. Additional Indicators

3.4.1. SQLi Payload Patterns

Indicators: Repeated attempts to submit data with payloads designed to trigger errors or return data structures, such as ' OR '1'='1.

3.4.2. Database Access Frequency

Indicators: A sudden increase in access frequency, especially from user accounts or IPs that do not normally initiate such traffic.

3.4.3. Session Hijacking Attempts

Indicators: Multiple failed login attempts followed by a sudden successful attempt, which may indicate an attacker exploiting a SQLi vulnerability to gain access.

4. Analysis

Initial Investigation

1. Data Collection: Gather logs from the web server, application server, and database. Ensure logs contain detailed timestamps, user IDs, query strings, and HTTP response codes.
2. Correlate Events: Use SIEM tools to correlate data from different sources to identify patterns that may indicate an SQLi attack.

Root Cause Analysis

- Trace Attack Path: Identify the entry points and how the SQLi was injected. Was it through user input, URL parameters, or HTTP headers?
- Vulnerability Identification: Determine whether input fields were not sanitized or queries were dynamically constructed without parameterization.
- Database Forensics: Review database logs for anomalies such as unauthorized data modifications or suspicious admin-level commands.

Pattern Recognition

- Historical Comparison: Compare with past incidents to identify similarities or new tactics. Utilize threat intelligence databases such as OTX or Mitre ATT&CK framework for context.
- Attack Vectors Analysis: Identify techniques used (e.g., Union-based SQLi, Blind SQLi, Error-based SQLi, Time-based SQLi).

5. Remediation – Contain, Eradicate, and Recover

5.1. Containment

Goal: Limit the spread and impact of the incident by isolating affected systems and restricting malicious access.

Steps:

1. **Isolate Affected Systems**:
   - **Network Segmentation**: Use network segmentation tools or firewall rules to isolate affected systems. For example, implement VLANs (Virtual LANs) or SDN (Software-Defined Networking) to compartmentalize traffic.
   - **Host Isolation**: Use tools like `tcpdump` or `Wireshark` to monitor real-time traffic to and from affected systems. Identify suspicious outbound connections and cut them off.
   - **Network Isolation Strategy**: Physically disconnect systems from the network if needed or block their access using network access control lists (ACLs) to contain the attack within that segment.
2. **Restrict Access**:
   - **Account Lockdowns**: Use centralized access management systems like Active Directory (AD) or Identity and Access Management (IAM) tools (e.g., Okta) to lock down user accounts associated with suspicious activity.
   - **IP Blocklist**: Add malicious IP addresses to firewall or proxy blocklists, and leverage threat intelligence feeds to automate updates for known malicious IPs.
   - **Temporary Credential Suspension**: Implement temporary account suspensions through automated scripts that check for unusual login patterns or locations, using SIEM (Security Information and Event Management) solutions like Splunk or QRadar.
3. **Implement WAF Rules**:
   - **Update Web Application Firewall (WAF)**: Ensure the WAF is configured to block known malicious patterns and signatures by applying updated rule sets from providers like AWS WAF, Cloudflare, or ModSecurity.
   - **Custom Rules**: Create custom WAF rules for unique attack vectors specific to the organization. This may involve writing regex patterns to detect anomalous SQL queries or unusual HTTP methods (e.g., `OPTIONS`).

5.2. Eradication

Goal: Remove the root cause of the incident to prevent recurrence.

Steps:

1. **Patch Vulnerabilities**:
    - **Immediate Patching**: Apply critical patches to vulnerable systems. Use patch management tools like Microsoft WSUS (Windows Server Update Services) or automated patch management platforms such as Ivanti or ManageEngine Patch Manager Plus.
    - **Review CVEs**: Utilize the CVE (Common Vulnerabilities and Exposures) database and tools like OpenVAS or Nessus to scan for known vulnerabilities in the environment and patch them.
    - **Hotfix Deployment**: In cases of urgent zero-day vulnerabilities, deploy hotfixes directly and monitor the systems for any unexpected behavior.
2. **Code Review and Refactoring**:
    - **Static Application Security Testing (SAST)**: Use tools like SonarQube, Checkmarx, or Fortify to perform a detailed code scan for security flaws.
    - **Review Vulnerable Code Patterns**: Identify and replace insecure code patterns like dynamic SQL with parameterized queries and prepared statements. For example, refactor code from:
    - "SELECT * FROM users WHERE username = '" + username + "' AND password = '" + password + """;

      To:

      "SELECT * FROM users WHERE username = ? AND password = ?";

1.
    - **Manual Code Review**: Conduct peer reviews, focusing on authentication/authorization logic, user input validation, and API security.
2. **Database Configuration Hardening**:
    - **Permission Audits**: Audit database user permissions regularly using tools like dbForge or manual SQL scripts to ensure that only required users have access to critical data.
    - **Disable Unused Features**: Turn off unused database services or functions (e.g., disable xp_cmdshell in SQL Server).
    - **Apply Security Best Practices**: Configure databases using principles like the least privilege, enforce strong authentication methods, and utilize encrypted connections (e.g., SSL/TLS for database traffic).

      5.3. Recovery

Goal: Restore systems to normal operation while ensuring they are secure and free of compromise.

Steps:

1. **System Integrity Checks**:
   - **Integrity Verification Tools**: Use tools like Tripwire or AIDE (Advanced Intrusion Detection Environment) to monitor and verify the integrity of critical files and configurations. Set up cron jobs or scheduled tasks to automatically run these checks.
   - **File Hash Comparison**: Regularly compare hashes of key files against a baseline hash stored in a secure location to detect unauthorized changes.
2. **Restore from Backup**:
   - **Backup Validation**: Ensure backups have been taken from systems before the attack and are free from malware. Test backups using sandbox environments to verify integrity.
   - **Automated Restoration**: Use backup and recovery tools like Veeam, Veritas NetBackup, or native solutions (e.g., Windows Server Backup) for automated, rapid restoration.
   - **Disaster Recovery Plans**: Execute pre-defined disaster recovery procedures that include restoring systems and testing functionalities before they are brought back online.
3. **Continuous Monitoring**:
   - **Re-enable Monitoring**: Restart continuous monitoring tools like SIEMs, IDS/IPS systems, and endpoint detection and response (EDR) solutions.
   - **Threat Hunting**: Initiate active threat hunting using tools like ELK Stack (Elasticsearch, Logstash, Kibana) and open-source threat intelligence platforms such as MISP (Malware Information Sharing Platform).
   - **Regular Scans**: Schedule vulnerability scans post-recovery to ensure no hidden threats remain.

3. 6. Post Incident

6.1. Lessons Learned

Goal: Improve future responses and strengthen defenses.

Steps:

1. **Incident Review**:

- **Incident Debrief**: Schedule a post-incident meeting involving all response team members. Use a structured approach to walk through the timeline, decision-making process, and effectiveness of containment and remediation efforts.
- **Root Cause Analysis (RCA)**: Conduct a thorough RCA using techniques like the "5 Whys" or a Fishbone diagram to identify the primary causes and contributing factors.

2. **Update Documentation**:
   - **Revision of the Playbook**: Incorporate the new insights into the incident response playbook. Ensure the updated procedures are distributed across relevant teams.
   - **Checklist Enhancements**: Create a checklist for quicker identification and response to similar threats in the future.
3. **Additional Training**:
   - **Refresher Courses**: Schedule training sessions for team members to review updated incident response protocols and security measures.
   - **Simulated Drills**: Organize tabletop exercises or red team/blue team simulations to practice the revised procedures and refine response capabilities.

6.Post Incident

Documentation:

- **Timeline of Events**:
  - Detail the sequence of events from detection through resolution. Include timestamps for each significant action and milestone (e.g., initial alert, escalation, containment measures, and final resolution).
  - Include relevant diagrams or visual timelines to simplify understanding.
- **Response Actions**:
  - List the steps taken during the response, including initial detection, investigation methods, communication protocols, and mitigation measures. This should also highlight any deviations from the incident response plan and their rationale.
- **Impact Analysis**:
  - **Operational Impact**: Specify the business units, services, or functions affected. Describe disruptions and their duration.
  - **Financial Impact**: Quantify the economic implications, such as lost revenue, response costs, and any potential fines or legal settlements.
  - **Data Breaches and Exposures**: Include details about compromised data, if any, including its type (e.g., PII, financial records) and how it was handled.
- **Root Cause Findings**:
  - Conduct a thorough examination to uncover the underlying cause of the incident, focusing on technical, procedural, and human elements.
  - Identify any gaps in the current security posture, such as unpatched vulnerabilities, flaws in access controls, or inadequate monitoring.
- **Lessons Learned**:
  - Summarize the takeaways, emphasizing what was done well and where improvements are needed.
  - Identify effective response tactics and highlight any necessary training or updates to security policies and procedures.

Utilizing Tools:

- **Documentation Platforms**: Use **Microsoft Word** or collaborative tools such as **Confluence** for detailed and easily accessible reports. Incorporate features like version control and collaborative editing for team input.

Enhanced Report Details:

- Use visual aids such as **charts**, **graphs**, **tables**, and **infographics** to illustrate critical data and trends.
- Ensure a professional layout that allows readers to quickly identify sections of interest.
- **6.2. Impact Assessment**
- **Operational Impact**: Discuss the services or functions affected, the extent of the downtime, and the direct consequences for stakeholders.
- **Financial Losses**:
    - Include calculations for loss of revenue, operational costs for response and recovery, and potential legal fees.
    - Address any third-party impacts or customer compensation, if applicable.
- **Data Breach Details**:
    - Specify the nature and scope of any data exposed or compromised.
    - Assess risks related to potential identity theft, privacy violations, and regulatory repercussions.
    - Note steps taken to mitigate data exposure and future protection plans.
- **6.3. Sharing Findings**

Internal Communication:

- **Distributing Reports**:
    - Send out comprehensive post-incident reports via **email**, **intranet** platforms, or **collaboration tools**.
    - Ensure that all involved stakeholders receive a summary version or key highlights, with full reports available for deeper analysis.
- **Documentation Access**:
    - Store reports in a secure, accessible location (e.g., on a **company intranet**, document management system, or **knowledge base**).

Cross-Team Awareness:

- **Presentation Sessions**:
    - Organize presentations and workshops to communicate findings across departments.

- Tailor sessions for different teams, such as IT, management, customer service, and compliance, focusing on relevant aspects of the incident and prevention.

- **Webinars and Meetings**:
  - Conduct webinars for remote teams to facilitate broad awareness and engagement.
  - Offer Q&A sessions to address questions and clarify points from the report.
- **Feedback and Recommendations**:
  - Use these sessions to collect feedback and incorporate suggestions from team members to improve future incident response plans.
  - Discuss new training needs or potential changes to security measures based on findings.

Promoting a Culture of Security:

- Emphasize the importance of collaborative learning to foster an organization's culture of proactive cybersecurity.
- Reinforce the importance of lessons learned in preventing future incidents and integrating them into existing incident response and risk management strategies.

7.Annex A: Flow Diagram