Héctor Ocaña

SQL Injection Playbook

Version History

Version	Update Date	Updated by	Resaon for Update
1.0	08/02/2024	Héctor	Initial Draft

Playbook

How to Use This Playbook

Purpose

To guide our organization in responding to a web application compromise incident. This playbook may also be used for a **website sql injection.**

How to use this Playbook

The steps in this playbook should be followed sequentially where appropriate. With many steps in the **Containment, Eradication, and Recovery** steps, some overlap may occur and is expected.

Playbook

Introduction

An SQL injection attack occurs when a malicious user inserts SQL code into input fields of a web application, exploiting vulnerabilities in the application's database interaction. By injecting SQL commands, attackers can manipulate database queries to retrieve, modify, or delete sensitive data. This type of attack exploits inadequate input validation and sanitation, allowing unauthorized access to databases and potentially compromising the entire system's security. To prevent SQL injection attacks, developers should implement secure coding practices like parameterized queries and input validation.

Playbook

Escalation Path

1. Initial Detection:

- Team Member Involved: Core Incident Response Team
- Action Steps:
  - Identify and verify the SQL injection incident.
  - Isolate affected systems to prevent further damage.
  - Begin initial investigation and data collection.
  - Ensure that web application backups are functioning as expected.

2. Severity Assessment:

- Team Member Involved: Core Incident Response Team (Cybersecurity Specialists)
- Action Steps:
  - Assess the severity and potential impact of the SQLi incident.
  - Determine if it's an isolated incident or part of a broader attack.
  - Evaluate potential data exposure or compromise.

3. Notification to Extended Teams:

- Team Member Involved: Core Incident Response Team (Team Lead)
- Action Steps:
  - Notify Legal, Compliance, and Customer Support teams about the incident.
  - Share initial findings and impact assessment.
  - Initiate collaboration with extended teams.
  - Document third-party web-hosting contacts

Playbook

Escalation Path

4. Investigation Deepening:

- Team Member Involved: Extended Teams (Legal and Compliance)
- Action Steps:
  - Legal team assesses any legal implications or obligations.
  - Compliance team ensures adherence to data protection regulations.
  - Communicate with external stakeholders if required.

5. Executive Briefing:

- Team Member Involved: Core Incident Response Team (Team Lead)
- Action Steps:
  - If severity warrants, brief Executive Leadership on the incident.
  - Provide a high-level overview of the situation, potential impact, and current actions being taken.
  - Discuss resource allocation and strategic decisions.

6. Public Relations Engagement:

- Team Member Involved: Extended Teams (Public Relations)
- Action Steps:
  - Public Relations team prepares for potential public communication.
  - Craft a message for customers or the public, if necessary.
  - Coordinate with Legal to ensure messaging compliance.

Playbook

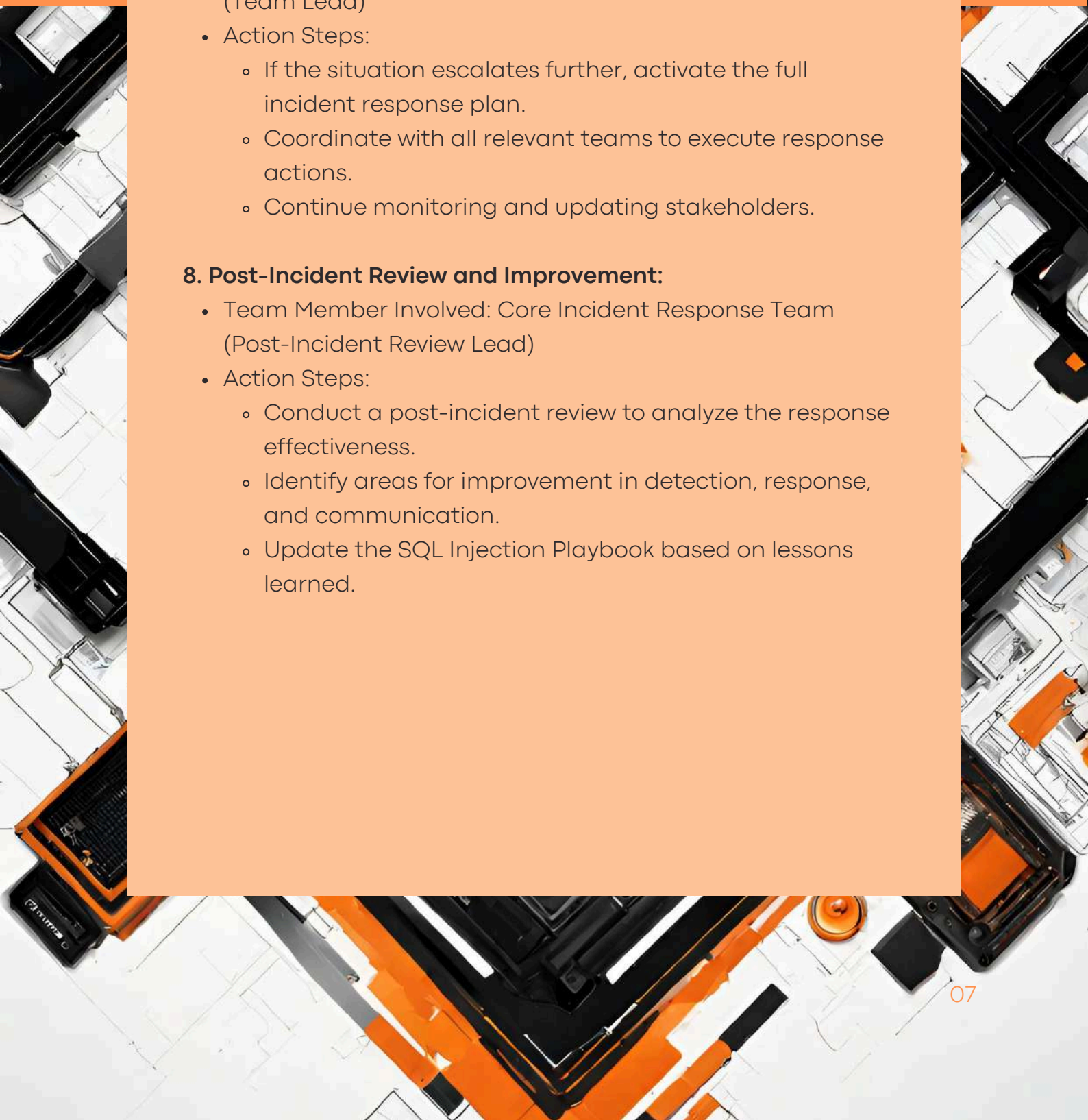
Escalation Path

7. Full Incident Response Activation:

- Team Member Involved: Core Incident Response Team (Team Lead)
- Action Steps:
  - If the situation escalates further, activate the full incident response plan.
  - Coordinate with all relevant teams to execute response actions.
  - Continue monitoring and updating stakeholders.

8. Post-Incident Review and Improvement:

- Team Member Involved: Core Incident Response Team (Post-Incident Review Lead)
- Action Steps:
  - Conduct a post-incident review to analyze the response effectiveness.
  - Identify areas for improvement in detection, response, and communication.
  - Update the SQL Injection Playbook based on lessons learned.



Playbook

Planning and Prevention

Performing regular security analyses throughout the software development process is essential to identify potential SQL injection vulnerabilities. Leveraging static code analysis tools and vulnerability scanners enables developers to detect and resolve security concerns prior to deploying the application in a production environment.

01 Dynamic Application Security Testing (DAST) Analysis:

Conduct DAST analysis in test and production environments to **identify potential SQL injection vulnerabilities** in the web application. Utilize automated security scanning tools to simulate attacks and detect weaknesses in the application. Some tools are listed later.

02 WAF (Web Application Firewall):

Implement a WAF to **monitor and filter** web traffic for attack patterns associated with SQL injections. Configure custom rules on the WAF to block requests attempting to exploit SQL injection vulnerabilities.

Planning and Prevention

03 Prepared Statments

Prepared Statements are a security feature provided by many database systems and programming languages. They are designed to separate SQL code from user input, thereby preventing SQL injection attacks.

Parameterization: Instead of directly embedding user inputs into SQL queries, use placeholders or parameters. These placeholders are then replaced with the actual values at runtime.

Automatic Escaping: Prepared Statements often handle automatic escaping of special characters, ensuring that user input is treated as data rather than executable code.

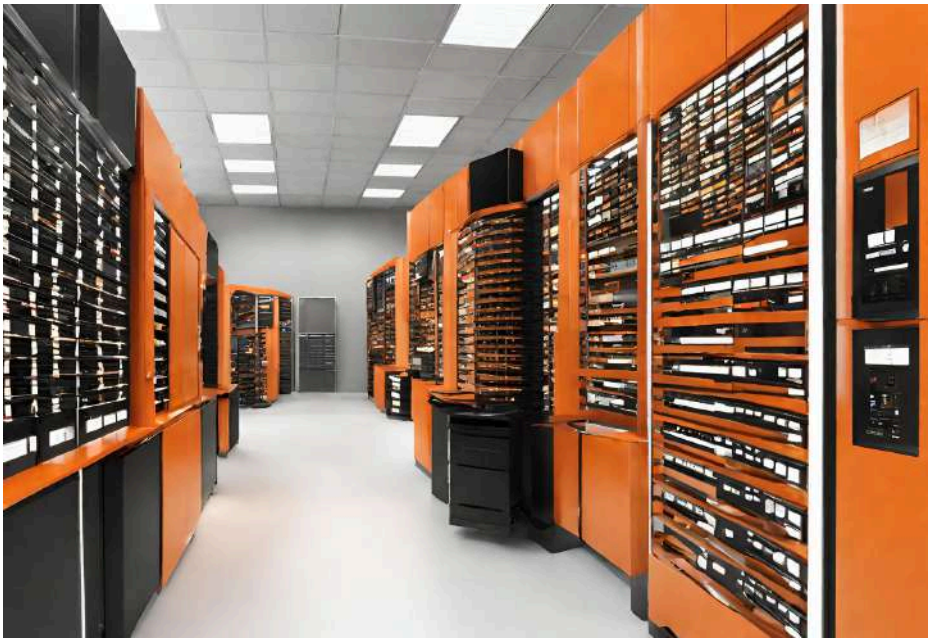
04 Object-Relational-Mapping (ORMs)

Object-Relational Mapping (ORM) tools **automate the generation of SQL queries in a secure manner**, alleviating the burden on developers and significantly **reducing** the **risk** of SQL injections. By using ORM frameworks, developers can work with high-level, object-oriented code, leaving the tool to handle the translation of these objects into SQL queries. ORM frameworks often incorporate best practices for secure database access, such as parameterized queries and proper data escaping.

Planning and Prevention

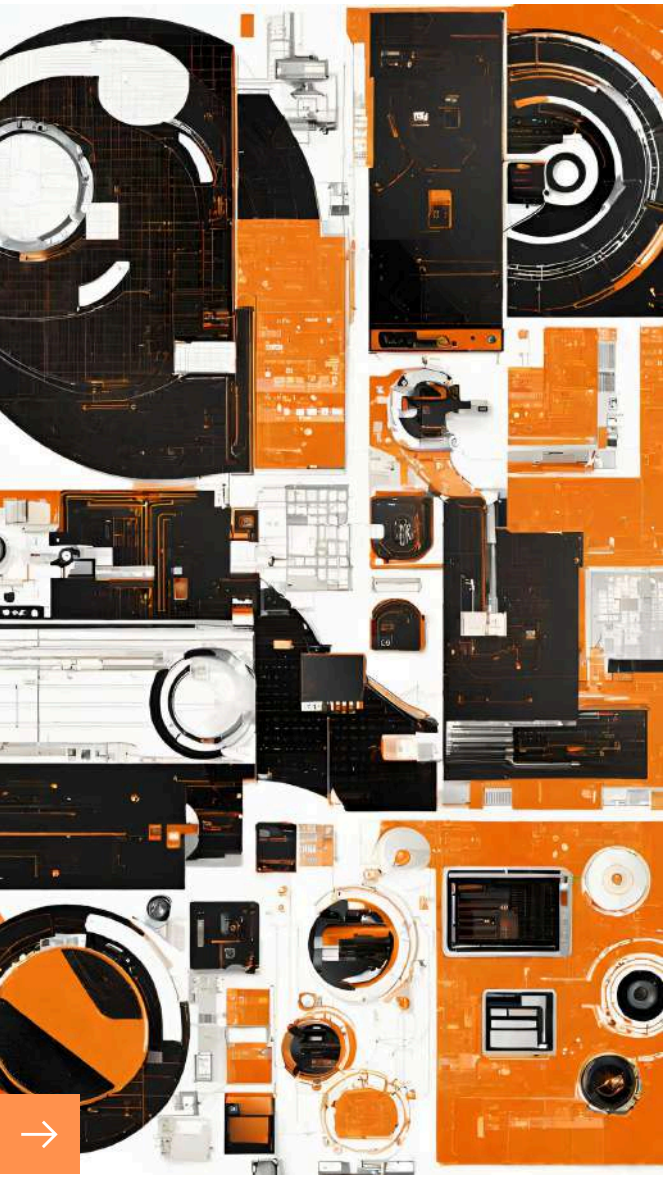
05 Stored Procedures

The use of stored procedures and functions in the database constitutes an effective strategy to add an additional layer of security and reduce vulnerability to SQL injections. Stored procedures are **predefined snippets of SQL code stored in the database**. By executing these procedures from the application instead of sending SQL queries directly, the exposure to potential attacks is minimized. Furthermore, stored procedures allow the centralization of database access logic, simplifying permission management and providing more precise control over data access. Incorporating stored procedures into the database architecture strengthens security and decreases the attack surface, as users interact with the database through controlled and predefined interfaces.



Playbook

Identification



Evaluate Evidence and Determine Compromise Method

- Examine abnormal behavior in the web application.
- Investigate notifications of compromise from external sources.
- Collect evidence such as client reports of anomalous or malicious behavior.
- Categorize the method of compromise (e.g., exploited vulnerability)

Initial Compromise Assessment

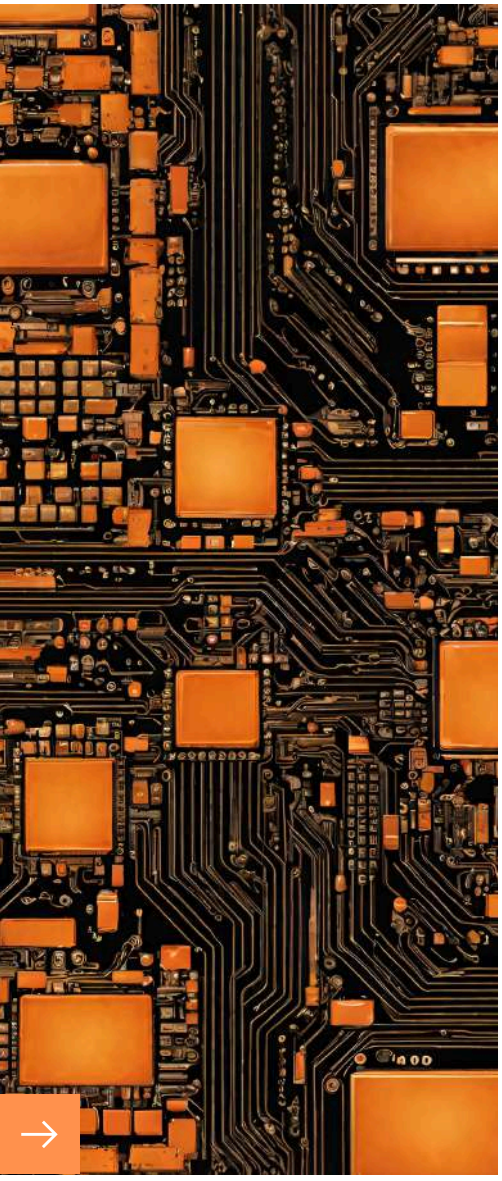
- Interview impacted users, asking specific questions about potential points of compromise.
- Check for abnormal actions on the user's workstation.
- Examine web history for visits to potentially malicious sites.
- Conduct a malware scan on the user's workstation.

Playbook

Identification

Indicators of Compromise (IoC)

- Use IoCs from the initial compromise to search for other potential victims.
- Utilize SIEM or log searches for IP addresses, URLs, workstation names, etc.
- Review logs in account login systems for anomalies, including unusual locations or browser fingerprints.
- Document all systems accessed by the attacker.



Use Tools like...

Netsparker: An automatic scanning tool that identifies and explores vulnerabilities, including SQL injections. Provides an automated approach to discovering security vulnerabilities in web applications.

Acunetix: A web security scanning tool that detects SQL injections, among other vulnerabilities. Performs automatic audits and provides detailed reports on vulnerabilities found.

Security AppScan: A web application security tool that helps identify and remediate vulnerabilities, including SQL injections. Provides automated scanning and testing to evaluate the security of web applications.

Playbook

Contact AEPD

In the event of a data security incident that impacts our clients' information, it is imperative that we take swift and decisive action to ensure compliance with regulatory requirements. As part of our commitment to transparency and adherence to data protection laws, it is crucial to promptly communicate and collaborate with the relevant authorities. In the context of our operations in Spain, this involves contacting the **Agencia Española de Protección de Datos** (AEPD). In this page, we outline the necessary steps to initiate contact and facilitate a timely and effective response to any potential data breaches.

Contact Information

Address:

C/ Jorge Juan, 6. 28001 - Madrid
28001 Madrid
Spain

Phone:

900 293 183

Email:

dpd@aepd.es

Website:

https://www.aepd.es

Containment

Block and Mitigate:

Identify and **disconnect** the affected system from the network to prevent the infection from spreading to other systems.

Maintain **constant monitoring** of traffic using tools like **Wireshark** or **tcpdump** to capture and analyze web traffic for patterns of malicious requests.

Utilize **role-based access control (RBAC) features** to temporarily disable access to compromised areas of the web application.

Restrict access to the affected system only to authorized personnel while resolving the issue.

Change passwords for affected systems and databases to prevent unauthorized access.

Playbook

Containment

Malicious Input Blocking:

Temporarily **shut down** the compromised **SQL service and** application to halt the execution of any injected malicious code.

Review and **Enhance** of input validation mechanisms to prevent future SQL injection attacks. Implement or enhance validation measures, such as using parameterized queries or prepared statements.

Perform a quick **forensic analysis** to understand the scope of the attack and how it occurred, which can help take corrective action.

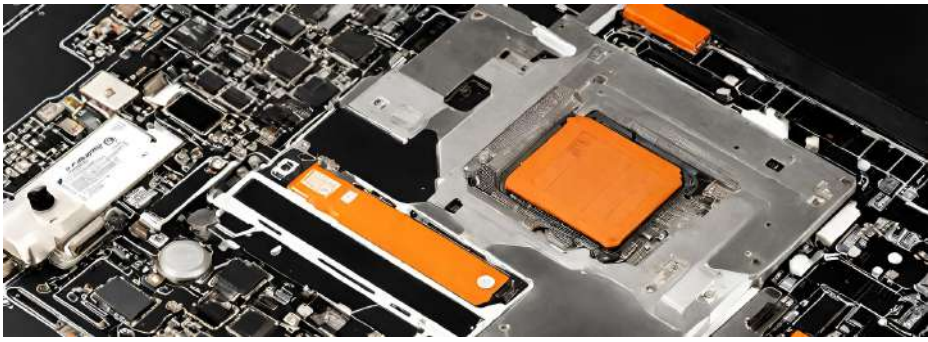
Eradication

Vulnerability Fixing

- Use static code analysis tools like **SonarQube** or **Checkmarx** to identify and address SQL injection vulnerabilities in the source code.
- Additionally, update the code with what has been detected in this attack, including the discovered vulnerability and record the attacker's IP address for further analysis and potential actions.
- Modify SQL queries to use **parameterized queries**, for example, changing string concatenation queries to prepared statements in languages like PHP with PDO or MySQLi.

Software Update

Applying patches and security updates to the web application and database management system is critical for maintaining system integrity and protecting against known vulnerabilities. Patch management tools like **Ansible** or **Chef streamline** this process by automating the deployment of updates across multiple systems.



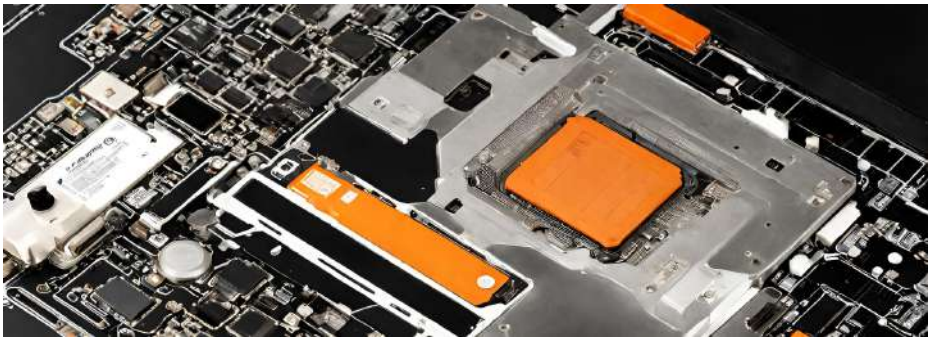
Recovery

Database Cleanup:

Use **SQL scripts to search for and remove malicious** records from the database. This may involve querying the database to identify any unauthorized or suspicious entries and deleting them. It's important to carefully review the data to ensure that only malicious records are removed, without affecting legitimate data.

After cleaning the database, **restore it from a recent backup** using tools like **mysqldump** or **pg_dump**. Ensure that the backup is from a point in time before the attack occurred to eliminate any compromised data.

Additionally, consider implementing **database integrity checks to verify the consistency and accuracy** of the data after the restoration process. This can help identify any discrepancies or abnormalities that may have been introduced during the attack or the restoration process.



Analysis

Affected Hosts Review:

Employ **forensic analysis tools** like **Volatility (logical acquisition from lime) and Autopsy (physical acquisition from dd)** to investigate affected hosts for additional indicators of compromise (IOC) and ensure they are patched and protected. These tools can help identify any remnants of the attack, such as malware, suspicious processes, or unauthorized access, and provide insights into how the attack occurred and what data may have been compromised.

During the forensic analysis, it's **essential to gather as much information as possible about the attack**, including timestamps, file modifications, network connections, and system logs. This information can aid in understanding the attack vector, the extent of the compromise, and any potential data exfiltration.

Playbook

POST-INCIDENT REVIEW

Conduct a post-mortem review meeting with the security and development team to analyze the response to the SQL injection incident.

Assess the effectiveness of incident response procedures and identify any shortcomings or areas for improvement.

Determine corrective actions to prevent similar incidents in the future.

Update the cybersecurity playbook with new recommendations and best practices identified during the post-incident review.

Disseminate lessons learned from the incident across the organization to foster a culture of continuous improvement and enhance overall cybersecurity awareness.

Conduct regular drills and simulations based on the updated playbook to validate the effectiveness of the incident response process and ensure readiness for future incidents.