Database Administration Security

Cybersecurity Department

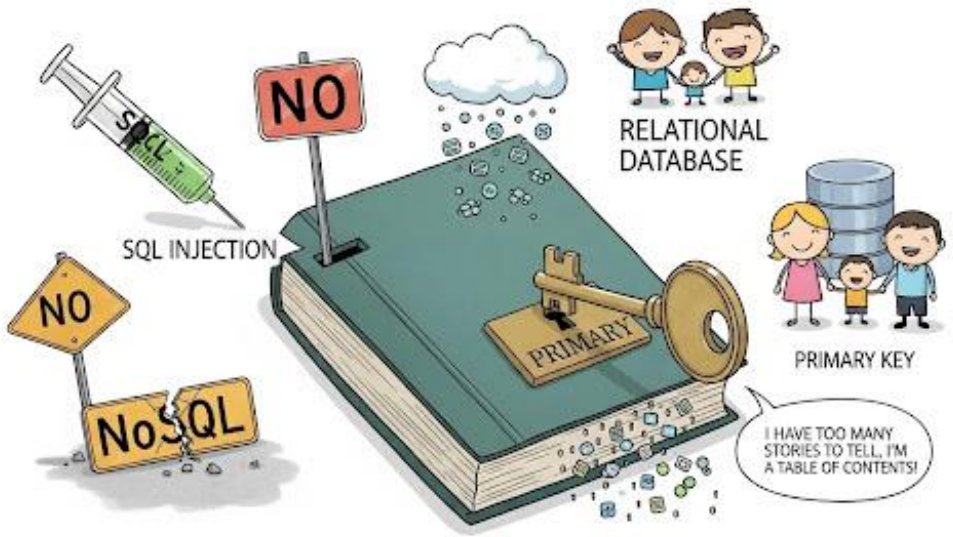
Course Code: ***CBS 214***

***Theoretical Lecture 6 and 7: Types of SQLi and* SQL Injection Preventions**

Halal Abdulrahman Ahmed

Agenda

- SQL Injection Security Levels
- Types of SQL Injection (SQLi)
- How to detect each type of SQLi?
- How to prevent each type of SQLi?
- Real-World Examples of SQL Injection Attacks



Learning Outcomes

By the end of this lecture, students will be able to:

- **Explain SQLi Mechanics:** Identify how unsanitized user input manipulates database queries to bypass authentication or steal data.
- **Classify Attack Types:** Differentiate between In-band (Error/Union), Blind (Boolean/Time), and Out-of-band SQL injection categories.
- **Compare Security Levels:** Evaluate how security measures in environments like DVWA (Low, Medium, High) affect vulnerability and exploitation.
- **Implement Defenses:** Apply primary prevention techniques, specifically parameterized queries (prepared statements), to separate SQL logic from user data.
- **Apply Global Security Policies:** Utilize a defense-in-depth approach involving Web Application Firewalls (WAFs), the Principle of Least Privilege, and secure error handling.

How an SQLi attack happens?

1. Vulnerable query

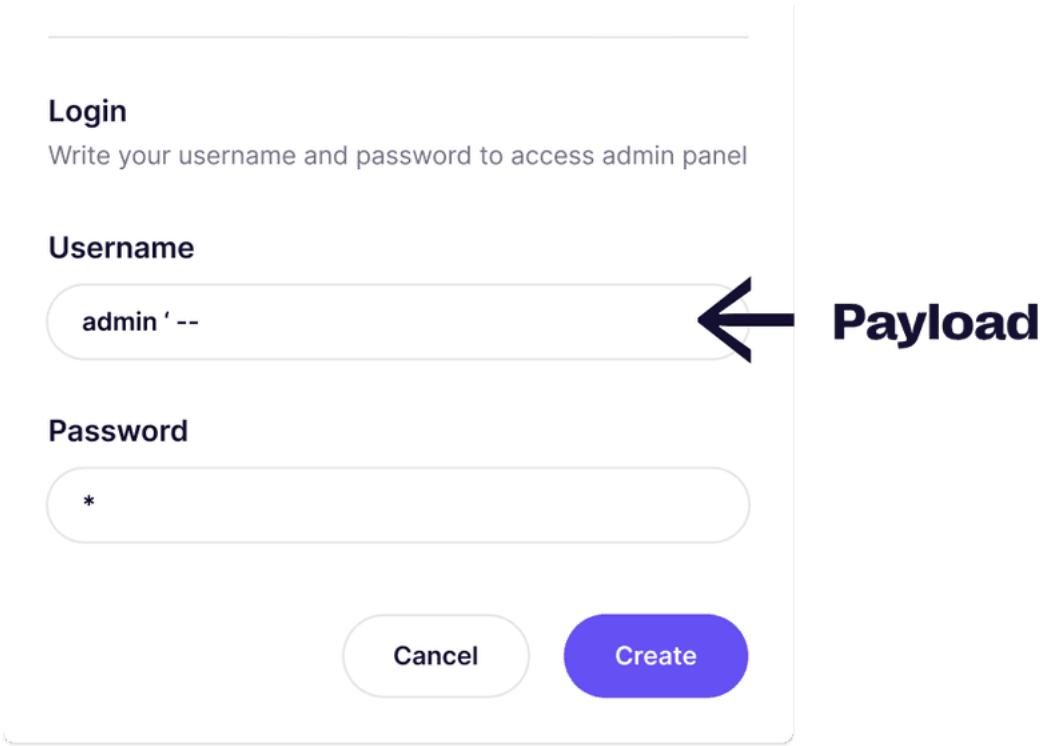
Example of a vulnerable query where user input is directly used in the query

```python
# Python code with an insecure SQL query
def login(username, password):
    query = f"SELECT * FROM users WHERE username = '{username}' AND password = '{password}';"
    # Execute the query (insecurely)
    # Assume some function execute_query(query) is called here
    return execute_query(query)
```

How an SQLi attack happens?

2. SQL injection attack

SQL is injected into the user input field of an *authentication page*



How an SQLi attack happens?

3. Query run with payload

Payload comments out the password check so the *query ignores this step*

```
# SQL payload injected
def login(username, password):
    query = f"SELECT * FROM users WHERE username = 'admin' -- AND password = '{password}';"
    # Execute the query (insecurely)
    # Assume some function execute_query(query) is called here
    return execute_query(query)
```

SQL Injection Security Levels (DVWA)

DVWA (Damn Vulnerable Web Application) provides different security levels for SQL Injection to help learners understand how security measures affect vulnerability and exploitation techniques.

- **Low Security**

At this level, user input is directly inserted into the SQL query without validation or sanitization, making it highly vulnerable to SQL injection.

SQL Injection Security Levels (DVWA)

```
SELECT first_name, last_name
FROM users
WHERE user_id = '$id';
```

What attackers can do:

- Input: ‘

Breaks query → SQL error revealed

- Input: 1' OR '1'=‘1

Query becomes always true → returns all users

- Input: 1' UNION SELECT user_name, password FROM users—

Can expose sensitive data

SQL Injection Security Levels (DVWA) (cont.)

- **Medium Security**

Some applications try basic filtering (like escaping quotes), but SQL Server can still be vulnerable depending on implementation. Example after escaping:

```
SELECT first_name, last_name
FROM users
WHERE user_id = '1'' OR 1=1';
```

Behavior:

- Single quotes are partially escaped
- Simple ' injection may fail

But still risky:

- Logic-based inputs like 1 OR 1=1 may still affect results if not handled properly in application logic

SQL Injection Security Levels (DVWA) (cont.)

- **High Security**

At this level, the application uses prepared statements (parameterized queries), which separate SQL logic from user input. Using parameterized queries in Microsoft SQL Server:

- **Security behavior:**
- @id is a parameter (not SQL code)
- Input is treated only as **data**
- Injection attempts like:
  - ' OR 1=1
  - UNION SELECT
  - '

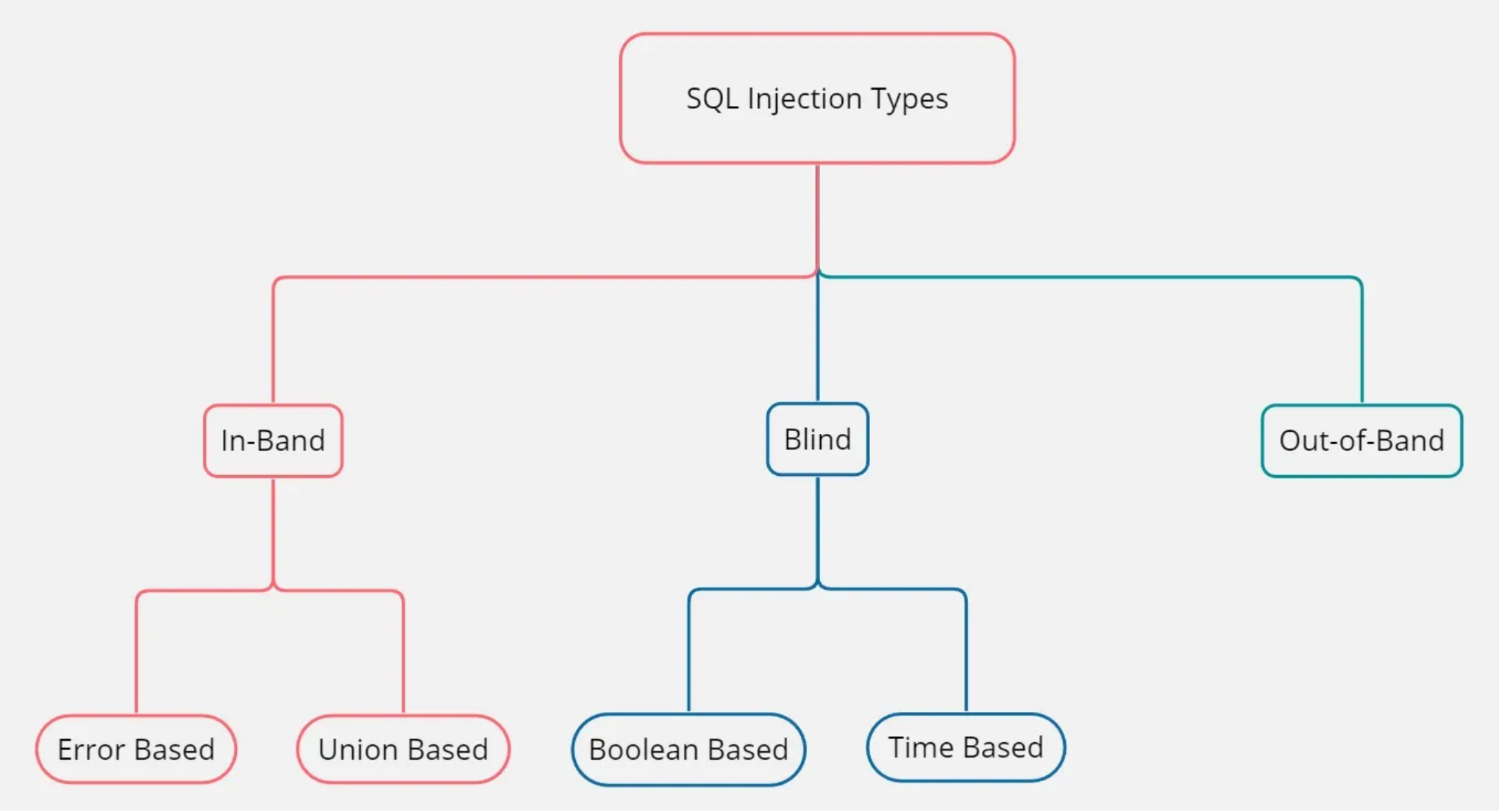
```
SELECT first_name, last_name
FROM users
WHERE user_id = @id;
```

will NOT change query structure

Types of SQL Injection (SQLi)

SQL Injection can be used in a range of ways to cause serious problems. By levering SQL Injection payloads, an attacker could bypass authentication, access, modify and delete data within a database. In some cases, SQL Injection can even be used to execute commands on the operating system, potentially allowing an attacker to escalate to more damaging attacks inside of a network that sits behind a firewall.

Types of SQL Injection (SQLi)



In-band SQLi (Classic SQLi)

In-band SQL Injection is the most common and easy-to-exploit of SQL Injection attacks. In-band SQL Injection occurs when an attacker is able to use the same communication channel to both launch the attack and gather results. There are two types of this attack:

- Error Based
- Union Based

Example In-band SQLi (Classic SQLi)

A cyber attacker inputs an SQL statement into the search field of an app or site. The output of the statement is also displayed on the same webpage.

How It Works?

To carry out a classic or in-band SQL injection attack, the attacker finds and exploits poorly created SQL queries of an application. They insert malicious SQL statements into input fields to alter the original logic of the query. And when they are successful at this, they can:

- Bypass authentication to gain access to the application
- Manipulate or steal confidential information
- Control administrative operations, such as changing/deleting records, creating new users, extending access privileges, making malicious backdoors for persistent access, etc.

How to Detect Classical SQLi?

- **Detection:** To detect a classic SQL injection attack, look for suspicious or unusual app activities and signs that could indicate the presence of a classic SQLi attack.
- **Abnormal app behavior**: Unusual or abnormal behavior in the app, such as unauthorized modifications to records, records added/deleted, sudden data leaks, attempts to break authentication, etc., could be due to an attacker.
- **Unexpected errors**: If an app returns database errors, such as invalid SQL statements, syntax errors, etc., someone could be altering the app's query logic.

How to Detect Classical SQLi? (cont.)

- **Suspicious SQL commands**: If you find suspicious SQL commands in your application and database logs, this could be SQLi. Look for SQL statements with special characters, such as UNION SELECT, ‘ OR ’1’ = ‘1’ to detect SQLi attacks.
- **Scanners**: Use cybersecurity tools, such as automated vulnerability scanners to detect SQL injection vulnerabilities.

How to Prevent Classical SQLi?

To prevent classic SQL injection attacks, take precautionary measures to remove vulnerable queries and other elements from your app. Some tips to prevent SQLi attacks:

- **Sanitize and validate your inputs:** Use secure libraries in your commands and avoid special characters, such as ‘ OR ‘1’ = ‘1’. Try whitelisting expected characters and reject unexpected characters to validate your inputs.
- **Use parameterized queries**: Use parameterized queries (eg. func(username, password)) to separate your SQL query code and user input data, instead of concatenating user input into your SQL queries.

How to Prevent Classical SQLi? (cont.)

- **Least privilege access**: Allow a minimum amount of access permission to users and accounts, enough to complete their tasks.
- **Use WAFs**: Use web application firewalls (WAFs) to filter different types of SQL injection and block them before they get to your app database.
- **Don’t expose error messages**: Try not to expose database errors to end users in detail as attackers could be one of them. With this information, anyone may plan attacks anytime soon.
- **Patches and audits**: Carry out security audits periodically to find and fix vulnerabilities faster. Keep your database updated.

In-band SQLi (Classic SQLi) (cont.)

- **Error-based SQLi**

Error-based SQLi is an in-band SQL Injection technique that relies on error messages thrown by the database server to obtain information about the structure of the database. In some cases, error-based SQL injection alone is enough for an attacker to enumerate an entire database. While errors are very useful during the development phase of a web application, they should be disabled on a live site, or logged to a file with restricted access instead.

How it Works:

A malicious actor inputs an SQL command that generates error messages intentionally from the database server. This lets them gain knowledge about the structure of the target database. They can also determine data values, column names, and table names with this method.

How to Detect Error-Based SQLi?

- **Detection**: To detect error-based SQL injections, resolve them immediately, and limit the damages. Consider these tips:
- **Unexpected error messages**: If you enter a query into the database and you see unexpected error messages, it could signal an SQLi attack.
- **Suspicious logs**: Check whether the application and database have any suspicious logs or unauthorized activities.

How to Detect Error-Based SQLi? (cont.)

- **Scan for vulnerabilities**: Scan your application frequently for SQLi vulnerabilities using automated tools to save time.
- **Penetration testing**: Run penetration testing on your applications to reveal security loopholes like a hacker does.

How to Prevent Error-Based SQLi?

- **Prevention**: Prevent error-based SQLi attacks from harming your organization in terms of finances, reputation, and customer trust with these tips:
- **Disable error messages**: Avoid revealing error messages to end users in detail. Attackers can use this weakness to carry out an SQLi attack. Try disabling error messages after an application or site is live. If you strictly need to, configure your app to display generic details.
- **Log error messages safely**: You can use a file to log error messages in it. Now, enforce access restrictions to this file to prevent it from falling into the wrong hands.

How to Prevent Error-Based SQLi? (cont.)

- **Use WAFs**: Use a web application firewall to block malicious SQL injection and prevent them from harming your database.
- **Audit and update**: Always keep your applications up-to-date with the latest version to prevent attackers from exploiting security vulnerabilities. You must also run periodic audits to find and fix hidden weak links.

In-band SQLi (Classic SQLi) (cont.)

- **Union-based SQLi**

Union-based SQLi is an in-band SQL injection technique that leverages the UNION SQL operator to combine the results of two or more SELECT statements into a single result which is then returned as part of the HTTP response.

How it Works?

First, an intruder tries identifying how many columns are there in the target database query. Until they see an error, the intruder keeps submitting different variations of this command to find the column count:

- *ORDER BY 1 - -*
- *ORDER BY 2 - -*
- *ORDER BY n - -*

Next, they use the *UNION* command to merge multiple SELECT statements into one to obtain more database information.

How to Detect Union-Based SQLi?

- **Detection**: Detecting a union-based SQL injection lets you combat and neutralize the threat before it becomes a full-blown attack. Here's how to detect union-based SQLi:
- **Analyze logs**: Check your database logs to find suspicious SQL commands, especially UNION statements. If you find one, start a deep investigation immediately.
- **Track page behavior**: Monitor your web page/application for unusual behavior, such as displaying additional details that you did not request.
- **Test**: Conduct security tests, such as penetration testing, on your application to understand if it has SQLi vulnerabilities. It tells you how secure your app is by acting like an attacker.

How to Prevent Union-Based SQLi?

- **Prevention**: If you want to reduce the chances of union-based SQL injections, prevent them with these methods:
- **Use prepared statements**: Secure your code from hackers with prepared statements or parameterized queries (eg. function(username, password)).
- **Restrict database permissions**: Limit people from accessing your database to prevent them from inserting ‘UNION’ queries.

How to Prevent Union-Based SQLi? (cont.)

- **Validate inputs**: Whitelist special characters and avoid suspicious ones to reduce the chances of hackers manipulating your codes.
- **Use advanced tools**: Use tools, such as vulnerability scanners, IDP/IPS, and WAFs to block SQL injection from reaching your application’s database.

Blind SQL Injection

A blind SQL injection attack occurs when the attacker injects malicious SQL commands into database fields “blindly,” meaning without directly obtaining the output of the command from the application, unlike classic SQLi. Instead, they look for indirect clues, such as HTTP responses, response times, app behavior, etc., to infer the result of the command. This is why they are also called blind SQL injection. The two types of inferential SQL Injection are *Blind-boolean-based SQLi* and *Blind-time-based SQLi*.

Blind SQL Injection

Example: A hacker may inject conditional statements to check if the database contains a specific piece of information based on how the app responds.

How It Works?

Blind SQLi attacks are definitely dangerous but not so common since they take quite a while to succeed. Since the app/site doesn’t reveal/transfer data to the hacker, the hacker sends harmful payloads to the database to learn the outputs themselves. They craft SQL queries to modify app behavior. After injecting the command, they observe how the app responds to the command to extract information.

How to Detect Blind SQLi?

- **Detection**: Detecting blind SQL injections can be difficult as they show no error warnings, unlike classic SQLi. But there are ways to detect them:
- **Monitor logs**: Set up security monitoring systems to track application logs. Review your database to detect queries that seem unusual or suspicious and investigate them immediately.
- **Check app behavior:** If the app slows down suddenly or returns unexpected responses for different patterns of queries, there could be a blind SQLi.

How to Detect Blind SQLi? (cont.)

- **IDS**: Intrusion detection systems (IDS) is a security solution to flag suspicious queries and validate them with deeper investigation.
- **Automated scanners**: Use automated vulnerability scanners to identify the presence of blind SQL injections in your app queries.

How to Prevent Blind SQLi?

- Finding and fixing blind SQLi are important, so no one can tamper with your database queries and gain sensitive data. The following measures should help you prevent them:
- **Prepared statements/parameterized queries**: Treat user input as data, instead of just code. Add parameters in queries to separate user input values from SQL code.
- **Error handling**: When you detect errors in your database, don’t expose them to the public just yet. First, find the fix and secure your database, so hackers don’t get to the vulnerability before you patch it.

How to Prevent Blind SQLi? (cont.)

- **Restrict access**: Limit access privileges by enforcing policies, such as least privilege access, zero trust, and role-based access controls to prevent unauthorized access.
- **Continuous monitoring**: Monitor your application and database for threats and fix them before they become an SQL injection attack.
-

Blind SQL Injection (cont.)

Boolean-based (content-based) Blind SQLi

- Boolean-based SQL Injection is an inferential SQL Injection technique that relies on sending an SQL query to the database which forces the application to return a different result depending on whether the query returns a TRUE or FALSE result.
- Depending on the result, the content within the HTTP response will change, or remain the same. This allows an attacker to infer if the payload used returned true or false, even though no data from the database is returned. This attack is typically slow (especially on large databases) since an attacker would need to enumerate a database, character by character.

Blind SQL Injection (cont.)

Time-based Blind SQLi

- Time-based SQL Injection is an inferential SQL Injection technique that relies on sending an SQL query to the database which forces the database to wait for a specified amount of time (in seconds) before responding. The response time will indicate to the attacker whether the result of the query is TRUE or FALSE.
- Depending on the result, an HTTP response will be returned with a delay, or returned immediately. This allows an attacker to infer if the payload used returned true or false, even though no data from the database is returned. This attack is typically slow (especially on large databases) since an attacker would need to enumerate a database character by character.

How it works?

- In this method, the hacker sends an SQL statement to make the target database execute a time-intensive task or wait for a few seconds to respond. Next, the hacker observes how much time (in seconds) the app takes to respond to the query to determine if the query is true or false.
- If the app responds immediately, the query is false and if it responds after waiting for a few seconds, the query is true.

How to Detect Time-based Blind SQLi?

- **Detection**: To detect time-based blind SQL injection, use the same methods that we discussed in blind SQL injection. Let's summarize them:
- **Analyze response times**: Send different SQL queries to the database to analyze their response times. Significant delays could indicate a time-based blind SQLi present in the code logic.
- **Check logs**: Check the app logs to detect signs of suspicious activities or unexpected delays.

How to Detect Time-based Blind SQLi?

- **Behavior analytics**: Track app behavior using anomaly detection tools. These will flag abnormal slowdowns and responses that indicate a vulnerability.
- **Vulnerability scanners**: Scan your application and database for SQLi vulnerabilities and resolve them immediately before they convert into attacks.

- **Prevention**: Time-based blind injections harm your organization by stealing confidential data and compromising systems. Here are some tips to prevent them:
- **Validate inputs**: Keep database inputs sanitized by creating an allowlist of expected inputs while rejecting unexpected ones or special characters.
- **Parameterized queries**: Use parameterized queries (eg. function (a,b)) to separate user data and code and prevent hackers from exploiting your database queries.
- **Test regularly**: Conduct vulnerability assessments and penetration testing on your application regularly to detect and remove vulnerabilities.
- **Update**: Keep your application and database updated to their latest versions to ensure you use secure programs, free of bugs and errors.
- **Limit access**: Restrict access privileges so that only authorized people get to access sensitive data.
- **Use advanced tools**: Use advanced security tools, such as vulnerability scanners, powerful WAFs, and intrusion prevention systems (IPS) to prevent threats.

Out-of-band SQLi

- Out-of-band SQL Injection is not very common, mostly because it depends on features being enabled on the database server being used by the web application. Out-of-band SQL Injection occurs when an attacker is unable to use the same channel to launch the attack and gather results.
- Out-of-band techniques, offer an attacker an alternative to inferential time-based techniques, especially if the server responses are not very stable (making an inferential time-based attack unreliable).
- This method is particularly **useful** when traditional SQL injection techniques, such as error-based or time-based attacks, are not effective due to **restrictions** in the environment or countermeasures implemented by the **defender**.

Out-of-band SQLi (cont.)

Out-of-band SQL injection (OOB SQLi) is a technique where an attacker doesn't rely on the web page to return data. Instead, they trick the database server into sending the data directly to them over the internet. This works because some databases can make network requests like DNS or HTTP. For example, in Microsoft SQL Server, a function called xp_dirtree can be abused to trigger a DNS request to a server controlled by the attacker, which can include sensitive information like the database name. Similarly, in Oracle Database, the UTL_HTTP package can send HTTP requests containing data to an external server. Tools like MySQL Workbench are not directly involved in this process they are just interfaces for managing databases. The key factor is the database server itself (like Microsoft SQL Server) and whether it is allowed to make external network requests.

How it Works?

Hackers carry out out-of-band SQLi in secure IT environments where the applications are configured to block direct database responses. In this scenario, they are unable to retrieve data using the same channel they used to insert malicious code. So, they opt for an external, more sophisticated communication mode instead, such as HTTP or DNS requests to fetch data from less secure databases.

How it Works? (cont.)

When an attacker injects harmful SQL commands into the database, it triggers or forces the database to connect to an external server that they have control over. This way, the attacker retrieves sensitive information such as user/system credentials, and analyzes and exploits it to control the database.

How to Detect Out-of-Band SQLi?

- **Detection**: To detect out-of-band SQLi to be able to fix them before any compromise happens, consider the following ways:
- **Monitor external traffic**: Since out-of-band SQLi requires an external connection, monitor your outbound traffic, such as HTTP or DNS requests. It helps you catch unusual traffic that could be an out-of-band SQLi.
- **Log database**: Frequently check your database logs to spot suspicious quests to known or external servers.
- **IDS**: Use intrusion detection systems (IDS) to detect the presence of unauthorized access attempts to an external server.

How to Prevent Out-of-Band SQLi?

- **Use allowlists**: Create an allowlist of authorized IP addresses and domains that your data must communicate with, rejecting all others. This restricts your database from connecting with malicious servers and exposing data.
- **Disable external communications**: Restrict access to your database, so external URLs, file systems, network connections, etc., cannot interact with it.
- **Use PoLP**: Limit database access permissions by applying the principle of least privilege (PoLP). It reduces the chance of malicious actors executing functions/commands, such as ‘load_file’.

SQL Injection Attack Types

In-band

- Error-based
- Union-based

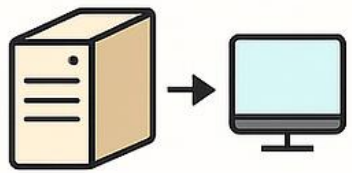
The attacker uses the same communication channel to launch and retrieve the attack

Blind

- Boolean
- Time-based

The attacker infers information by observing the application's behavior and responses

Out-of-band



The attacker uses an external channel (eg. a DNS or HTTP request) to retrieve data from database

Do not attempt illegal testing.

Real-World Examples of SQL Injection Attacks

- **Equifax data breach:** Equifax, a credit reporting company, faced an SQLi attack in 2017. Attackers found and exploited a less secure web app and executed malicious SQL queries to gain unauthorized access to confidential customer information.
- This exposed 143 million customers' data including their names, birthdays, addresses, and social security numbers. The company could not detect the attack for months and when it did, it was too late. Equifax had to pay more than $1.4 billion in fines and settlements.

Real-World Examples of SQL Injection Attacks

- **Yahoo:** Yahoo suffered a massive SQLi attack in 2014 where attackers injected malicious code into its vulnerable apps. With this attempt, the attackers successfully gained 500 million accounts of Yahoo's users.
- With this data breach, attackers compromised usernames, passwords, security questions, and more. As a result, Yahoo's valuation went down, which affected Yahoo during its acquisition by Verizon. It once had a $100 billion valuation, but Yahoo had to sell it for only $4.83 billion.

How to Prevent SQL Injection Attacks?

A single application with malicious SQL injection is enough to allow cybercriminals to access a company's database and confidential data. This will lead to financial and reputational damage. But you can protect your applications from different types of SQL injection attacks by following these tips:

- Avoid embedding user inputs directly in SQL queries. Instead, use limited queries and prepared statements to differentiate code from data. This will execute queries securely and prevent SQLi attacks.
- Restrict your SQL database access based on roles and permissions. Allow users minimum privileges, necessary to perform actions to reduce the impact of the breach.

How to Prevent SQL Injection Attacks? (cont.)

- Use stored procedures to control the interaction between queries and the database to reduce direct exposure to SQL commands from the user inputs.
- Execute strict validation rules to use only expected data formats to block malicious SQL code from entering your system.
- Regularly update applications and check security patching to minimize security risks.
- Use advanced cybersecurity tools to detect and block SQLi attempts before they enter your database. The tools monitor unusual database queries, query patterns, and logging access to eliminate risks.

Summary

Different types of SQL injection attacks target web applications that use SQL databases to gain unauthorized access. Cybercriminals often succeed because of the vulnerabilities in applications, such as unsanitized user inputs. Once they get access, they can steal customer information, confidential company data, financial records, etc., and harm your business' reputation.

Summary (cont.)

- The key preventive measures you need to implement include input validation, continuous monitoring using advanced security tools, prepared statements, limited access, and more. Attackers always refine their strategies to gain access, so you must stay ahead using penetration testing, threat detection, and regular security audits.
- Don’t wait for a cybercriminal to take action first. Instead, be the first to secure your applications from different types of SQL injection attacks.

Research Tasks (Homework)

- What is xp_dirtree ?
- What is **SQL injection cheat sheet?**
- What is **sqlmap**, and how does it automate the detection of the types we studied?
- What is the difference between a Web Application Firewall and a Database Firewall?
- Where does SQLi currently rank in the **OWASP Top 10** vulnerabilities list?

References

- SentinelOne. (2025, July 24). *7 types of SQL injection attacks & how to prevent them*. https://www.sentinelone.com/cybersecurity-101/cybersecurity/types-of-sql-injection/
- GeeksforGeeks. (2025, July 23). *Types of SQL injection (SQLi)*. https://www.geeksforgeeks.org/ethical-hacking/types-of-sql-injection-sqli/
- Fortinet. (n.d.). *SQL injection (SQLi)*. Fortinet Cybersecurity Glossary. https://www.fortinet.com/uk/resources/cyberglossary/sql-injection
- Aikido Security. (2025). *The state of SQL injections*. https://www.aikido.dev/blog/the-state-of-sql-injections


Any
Question