SQL Injection

https://portswigger.net/web-security/sql-injection
5 / 16 labs

https://portswigger.net/web-security/sql-injection/cheat-sheet

1. Definition

- allows an attacker to inject/modify sql commands in query to access DB

2. Impact

- Steal sensitive data and modify/delete data

3. Detecting SQLi

You can detect SQL injection manually using a systematic set of tests against every entry point in the application (and check the app behavior, look for errors). To do this, you would typically submit:

- The single quote character `'` and look for errors or other anomalies.
- Some SQL-specific syntax that evaluates to the base (original) value of the entry point, and to a different value, and look for systematic differences in the application responses.
- Boolean conditions such as `OR 1=1` and `OR 1=2`, and look for differences in the application's responses.
- Payloads designed to trigger time delays when executed within a SQL query, and look for differences in the time taken to respond.
- OAST payloads designed to trigger an out-of-band network interaction when executed within a SQL query, and monitor any resulting interactions.

3. 1. SQL injection in different parts of the query

Most SQL injection vulnerabilities occur within the `WHERE` clause of a `SELECT` query (because you can append ' to stop and inject command then -- to comment the rest)

other common locations where SQL injection arises are:

- In `UPDATE` statements, within the updated values or the `WHERE` clause.
- In `INSERT` statements, within the inserted values.
- In `SELECT` statements, within the table or column name.
- In `SELECT` statements, within the `ORDER BY` clause.

3. 2. SQL injection in different context

SQL injection with filter bypass(WAF) via XML encoding
websites take input in JSON or XML format and use this to query the database (not as normal query).
example
`<stockCheck> <productId>123</productId> <storeId>999 &#x53;ELECT * FROM information_schema.tables</storeId> </stockCheck>`

These different formats may provide different ways for you to obfuscate attacks like encoding

obfuscating your payload using XML entities. One way to do this is using the Hackvertor extension

Bypass the WAF

1. As you're injecting into XML, try obfuscating your payload using XML entities. One way to do this is using the Hackvertor extension. Just highlight your input, right-click, then select **Extensions > Hackvertor > Encode > dec_entities/hex_entities**.
2. `<storeId><@hex_entities>1 UNION SELECT username || '~' || password FROM users<@/hex_entities></storeId>`
   https://portswigger.net/web-security/sql-injection/lab-sql-injection-with-filter-bypass-via-xml-encoding

4. Examples

1. Retrieving hidden data (by removing the where clause)

URL
`https://insecure-website.com/products?category=Gifts`

ACTUAL QUERY
`SELECT * FROM products WHERE category = 'Gifts' AND released = 1`

inject on URL `category=Gifts'+OR+1=1--` to retrieve all

- single quote to end the where statement and append OR 1=1
- -- denotes comment the rest
- + is url encoding for space

2. Subverting(changing) App logic

- Login bypass by adding '-- to end statement and comment rest
  SELECT * FROM users WHERE username = 'administrator '-- ' AND password = ''

3. Retrieved data from other DB

- by appending ' UNION SELECT username, password FROM users--

4. Blind SQL Injection

- Executes query but app does not display results from db

The following techniques can be used to exploit blind SQL injection vulnerabilities, depending on the nature of the vulnerability and the database involved:

- You can change the logic of the query to trigger a detectable difference in the application's response depending on the truth of a single condition. This might involve injecting a new condition into some Boolean logic, or conditionally triggering an error such as a divide-by-zero.
- You can conditionally trigger a time delay in the processing of the query. This enables you to infer the truth of the condition based on the time that the application takes to respond.
- You can trigger an out-of-band network interaction, using OAST techniques. This technique is extremely powerful and works in situations where the other techniques do not. Often, you can directly exfiltrate data via the out-of-band channel. For example, you can place the data into a DNS lookup for a domain that you control.

5. Second-order SQL injection
   First-order SQL injection (get input then process IMMEDIATELY) occurs when the application processes user input from an HTTP request and incorporates the input into a SQL query in an unsafe way.

Second-order SQL injection(get input, store then process in FUTURE) occurs when the application takes user input from an HTTP request and stores it for future use. This is usually done by placing the input into a database, but no vulnerability occurs at the point where the data is stored. Later, when handling a different HTTP request, the application retrieves the stored data and incorporates it into a SQL query in an unsafe way. For this reason, second-order SQL injection is also known as stored SQL injection.

5. Attacks

5.1 EXAMINING DATABASES
Necessary to know the database they're using

- The type and version of the database software.
- The tables and columns that the database contains.

  5.1.1 QUERYING TYPE AND VERSION
  inject using union
  ex. `' UNION SELECT @@version--`

|Database type | Query |
|Microsoft, MySQL | `SELECT @@version` | does not work, see other payload
|Oracle | `SELECT * FROM v$version` | " * " does not work, see other payload
|PostgreSQL | `SELECT version()` |

1. Oracle
   On Oracle databases, every `SELECT` statement must specify a table to select `FROM`. If your `UNION SELECT` attack does not query from a table, you will still need to include the `FROM` keyword followed by a valid table name.

There is a built-in table on Oracle called `dual` which you can use for this purpose. For example: `UNION SELECT 'abc' FROM dual`

@TTACK
ON CATEGORY SELECT APPEND: (include NULL)
PAYLOAD `'+UNION+SELECT+BANNER,+NULL+FROM+v$version--`

2. Mysql and Microsoft
   PAYLOAD `'+UNION+SELECT+@@version,+NULL#`

5.1.2 LISTING DB CONTENTS (table list)
Most db has "information schema" table except oracle

View tables :

- general
  `SELECT * FROM information_schema.tables`
- oracle
  `SELECT * FROM all_tables`

Check Table Schema

- general
  `SELECT * FROM information_schema.columns WHERE table_name = 'Users'`
- oracle
  `SELECT * FROM all_tab_columns WHERE table_name = 'USERS'`

@TTACK
ON CATEGORY SELECT, APPEND: (include NULL)

PAYLOAD

5.2 UNION ATTACKS