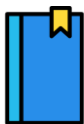




Web Application Penetration Testing Basics

Ethical Hacking Lab Exercise



Take note!

Information contained in this document is for educational purposes only.

Originally by Colin McLean (@Doctor_Hacker)

Adapted and expanded by Jamie O'Hare (@TheHairyJ)

Icons made by [Darius Dan](#) from [Flaticon](#)

Contents

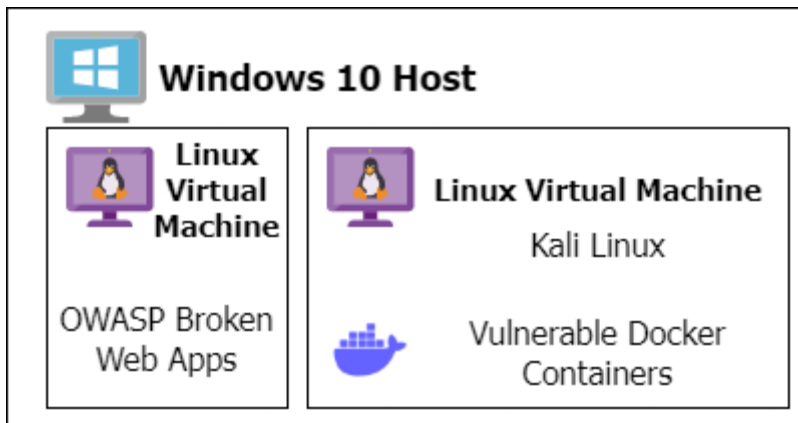
1	The Lab Environment	3
1.1	Topology Overview	3
1.2	Using VMWare.....	3
1.3	Using the Kali Linux Virtual Machine	4
1.4	Using the Web App Virtual Machine	5
1.5	Using the Vulnerable Web App Docker Containers.....	7
1.6	Debugging.....	8
2	Exploring the Vulnerable Web Apps	12
2.1	Exploring the Broken Web Apps Virtual Machine	12
2.1.1	Main Training Applications.....	12
2.1.2	The Backend of the Virtual Machine.....	12
2.1.3	Realistic Web Apps.....	12
2.1.4	Others.....	13
2.2	Examining the Main Training Applications	13
2.2.1	Damn Vulnerable Web App “DVWA”	14
2.2.2	Mutillidae	15
2.2.3	bWAPP.....	17
2.3	Examining the databases.....	18
2.4	Exploring the Rest of the Virtual Machine	19
3	Using Proxies in Web App Penetration Testing	20
3.1	Running ZAP.....	20
3.1.1	Using Intercept in ZAP	22
3.2	Running Burp Suite	22
3.2.1	Using Intercept in Burp Suite	24

1 THE LAB ENVIRONMENT

The following exercises are to aid in your familiarization with our bespoke lab environment. Additionally, through completing this activity, you'll be exposed to the broad range of tools available for learning and testing web application security.

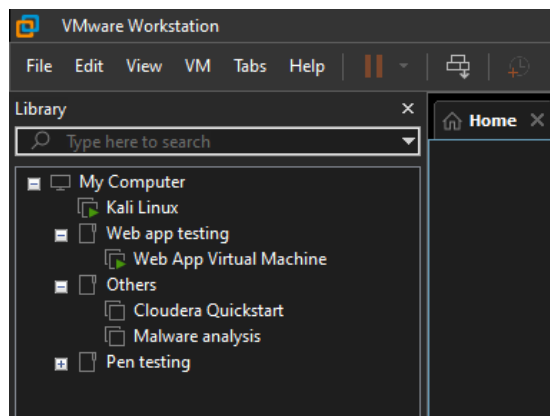
1.1 TOPOLOGY OVERVIEW

For this module's lab exercises, we will primarily be using a standalone vulnerable web server located on a separate virtual machine, however, other local virtualised target systems will be used. On the other side, there will be two attacking systems. One will be the system (Windows 10 Desktop) hosting the Virtual Machines, and the other is the Kali Linux virtual machine. The virtualisation software of choice is VmWare Workstation. A network topology for this setup is found in the figure below.



1.2 USING VMWARE

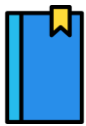
- From the Desktop, launch VMWare Workstation. You should observe a list of Virtual Machines like that shown below.



From this view, it is possible to launch the various Virtual Machines

1.3 USING THE KALI LINUX VIRTUAL MACHINE

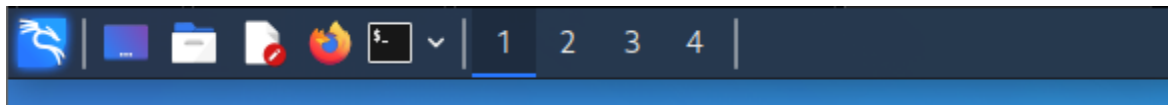
With Kali Linux now launched, let us explore what this offensive security flavoured Linux Virtual Machine. We will have to start by logging in first.



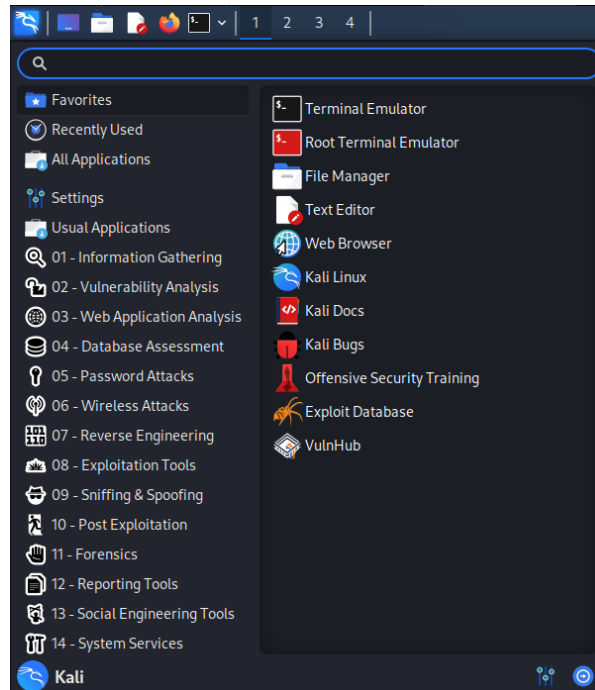
Take note!

The login credentials for this Kali Linux instance are **kali / kali**.

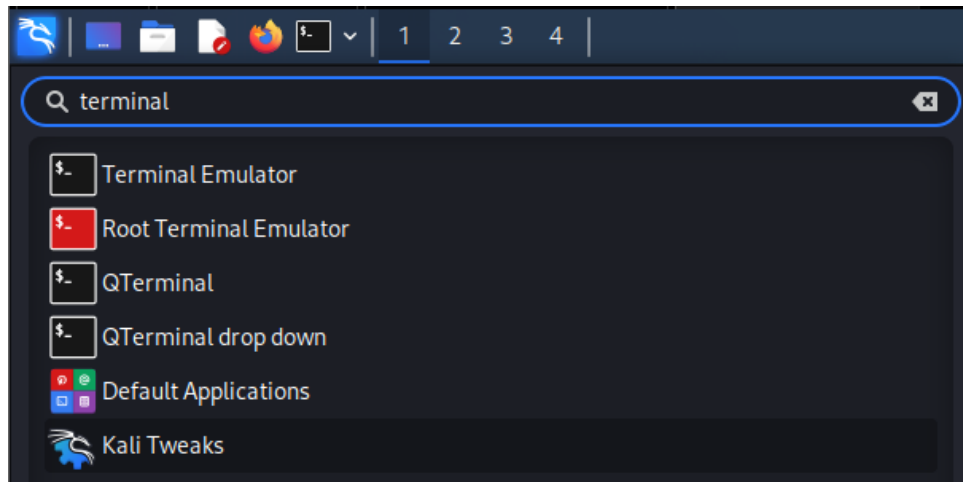
The top left of your Kali Desktop should resemble the figure below. From left to right, these icons represent: System menu, Desktop, File Explorer, Note taking application, Firefox Browser, Terminal, then a chevron to expand the taskbar before a menu allowing the user to jump between 4 different desktop perspectives.



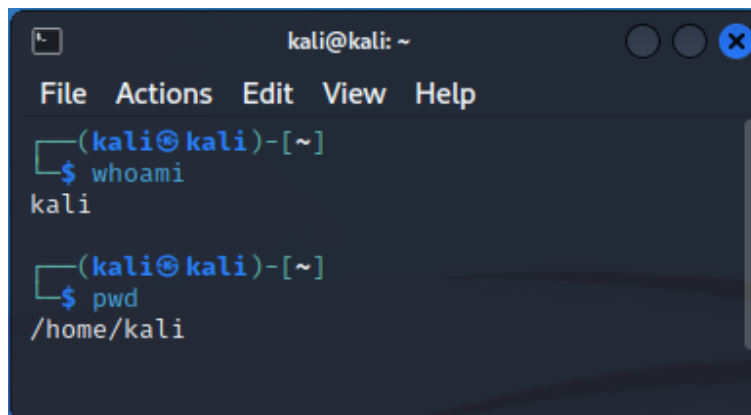
Start by selecting the left most icon featuring the Kali Dragon. After doing so, you should be presented with the following menus. From this menu, you are able to explore the tools Kali has to offer.



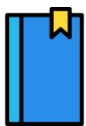
However, if you were looking for a certain tool, you can use the search bar found at the top of this menu to filter available tools, as shown in the figure below.



- Use the search functionality or the appropriate taskbar icon to launch a terminal session.



The terminal is a command line interface. You will use it often throughout this module; therefore, it is imperative you are well versed on how to use a Linux command line interface.



Take note!

Updates to Kali Linux have taken away superuser permissions from the distro's default user. As such, when you must use the **sudo** command, the password is **kali**.

1.4 USING THE WEB APP VIRTUAL MACHINE

Up next on the tour of the lab environment is the Web App Virtual Machine.

- From VMWare Workstation, launch the Web App Virtual Machine. Once done, navigate to the Web App Virtual Machine tab inside VMWare, and you will see the system running (i.e., displaying output inside a terminal).

Welcome to the OWASP Broken Web Apps VM

!!! This VM has many serious security issues. We strongly recommend that you run it only on the "host only" or "NAT" network in the VM settings !!!

You can access the web apps at <http://192.168.1.100/>

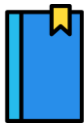
You can administer / configure this machine through the console here, by SSHing to 192.168.1.100, via Samba at \\192.168.1.100\\, or via phpmyadmin at <http://192.168.1.100/phpmyadmin>.

In all these cases, you can use username "root" and password "owaspbwa".

OWASP Broken Web Applications VM Version 1.2

Log in with username = root and password = owaspbwa

owaspbwa login:



Take note!

While not necessary, if you ever need to authenticate with this web server, the credentials are **root / owaspbwa**.

- While displaying the output in the terminal, in the background multiple web applications are accessible. You may have spotted the URL in the terminal output, however, if not it is linked below.

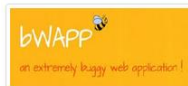
<http://192.168.1.100>

- Browse to this page, either on your Desktop or via Kali Linux and you should be greeted with the following web page.

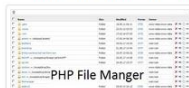
Abertay Hacklab vulnerable web application machine

This virtual machine has been setup for use by Ethical Hacking students at Abertay University, Dundee.

Main training applications.



Examine the virtual machine.



Realistic Web apps.



Others.

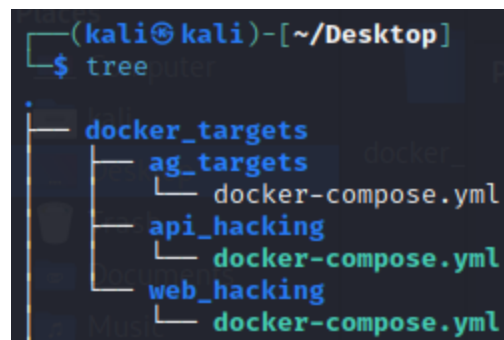


All the applications pictured and linked in the web page (shown in the Figure above) are all running on the Web App Virtual Machine. For now, this is the far as we will explore this web server. However, we will return to have a closer look at each application later.

1.5 USING THE VULNERABLE WEB APP DOCKER CONTAINERS

While some vulnerable applications exist on an isolated Virtual Machine, some are much closer to home! Through the power of containerisation (thanks to Docker!), there are multiple vulnerable web applications available on the Kali Linux Virtual Machine.

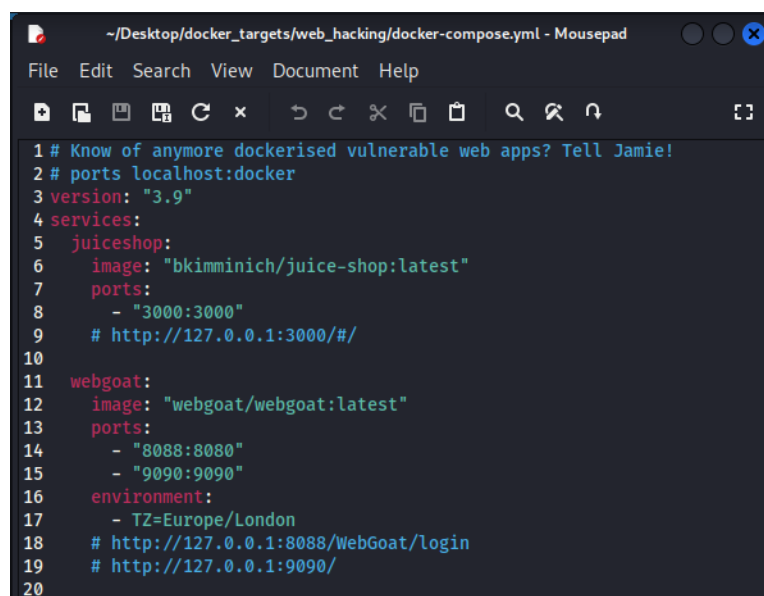
- Return to the Kali Linux Virtual Machine and launch a terminal. From this terminal navigate to the directory `docker_targets` found on the Desktop. Use the `tree` command to look at what is located inside this directory, it should resemble something like the Figure below.



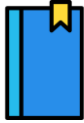
```
(kali㉿kali)-[~/Desktop]
$ tree
.
├── docker_targets
│   ├── ag_targets
│   │   └── docker-compose.yml
│   ├── api_hacking
│   │   └── docker-compose.yml
│   └── web_hacking
│       └── docker-compose.yml
```

- Navigate into the `web_hacking` directory to discover a `docker-compose.yml`. Examine this file closer using the following command, it should resemble the output below. (Figure output is using Mousepad to allow for reference to line numbers).

`cat docker-compose.yml`



```
~/Desktop/docker_targets/web_hacking/docker-compose.yml - Mousepad
File Edit Search View Document Help
1 # Know of anymore dockerised vulnerable web apps? Tell Jamie!
2 # ports localhost:docker
3 version: "3.9"
4 services:
5   juiceshop:
6     image: "bkimminich/juice-shop:latest"
7     ports:
8       - "3000:3000"
9     # http://127.0.0.1:3000/#/
10
11   webgoat:
12     image: "webgoat/webgoat:latest"
13     ports:
14       - "8088:8080"
15       - "9090:9090"
16     environment:
17       - TZ=Europe/London
18     # http://127.0.0.1:8088/WebGoat/login
19     # http://127.0.0.1:9090/
20
```

**Take note!**

Each application is bound to a different port (listed in the file examined) Yet some tools may use the same ports causing a collision – be aware of this possibility.

When executed this file launches all the web applications listed within. Each application is a different vulnerable web application, like Broken Web Apps Virtual Machine. To launch the applications, follow the instructions below.

```
sudo docker-compose up
```

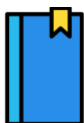
Once launched the different applications can be found at their respective local host addresses (lines 9, 18 and 19 in the above figure). Rather than entering the different unique URLs for each application, open the Firefox browser on Kali and you should find a Folder on the bookmarks bar named Web Hacking with all the applications recently launched linked inside.



- Click through some of these applications to see what is on offer.

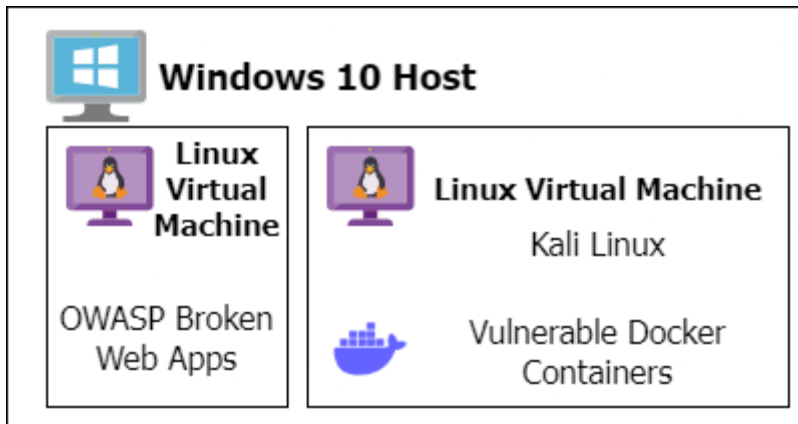
1.6 DEBUGGING

As is possible during penetration testing engagements, it is likely you may need to spend time debugging networking issues. This following section outlines how to perform some rudimentary debugging to allow you to get your network back online yourself.

**Take note!**

The following section uses the Hacklab environment as an example. If working within a different environment, please consider the appropriate differences.

Referring to topology diagram found earlier in this exercise, as shown below. There are many points of failure.



The main issue you will likely have to debug is the various networked VMs and host not being able to correspond with one another. All networked components should be on the same network. In the Hacklab this should be the **192.168.1.x/24 network**.

For the **Windows** host, the Virtual Adapter has been set to **192.168.1.254**. To confirm this, you can issue the ipconfig command, as show in the output below.

```
C:\Users\admin>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

    Connection-specific DNS Suffix  . : uad.ac.uk
    Link-local IPv6 Address . . . . . : fe80::efc4:9bbf:a2b9:cf5d%11
    IPv4 Address. . . . . : 10.1.1.23
    Subnet Mask . . . . . : 255.255.248.0
    Default Gateway . . . . . : 10.1.0.1

Ethernet adapter VMware Network Adapter VMnet1:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::168f:2f8d:4489:1116%12
    IPv4 Address. . . . . : 192.168.10.254
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 

Ethernet adapter VMware Network Adapter VMnet8:

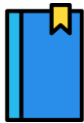
    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::abc6:f49:397c:add%9
    IPv4 Address. . . . . : 192.168.233.1
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 

Ethernet adapter VMware Network Adapter VMnet2:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::d2ee:7c92:269d:31bc%7
    IPv4 Address. . . . . : 192.168.1.254
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . :
```

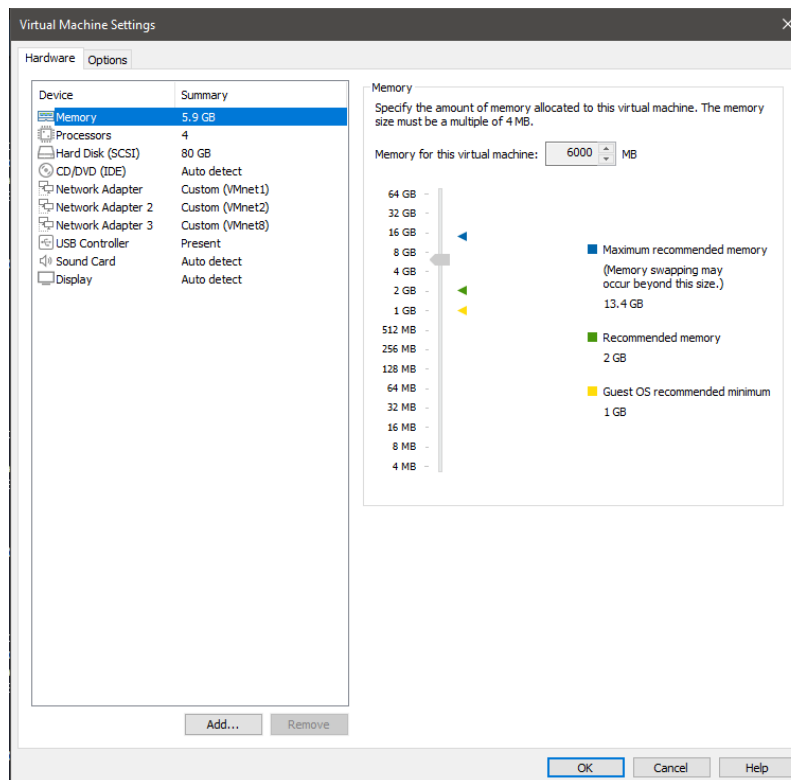
However, as you can see in there is network adapters (ethernet, VMnet1, VMnet 8, and VMnet 2) in the command output. How do we know if the correct adapter has the correct address? Well for that we need to consult the VMWare settings.

Initially navigate to the settings of the VMWare in question. The Figure below displays the settings for the Kali Linux VM. Inside the left-hand pane, you will notice 3 network adapters. These have been configured to use a customer VMnet adapter.

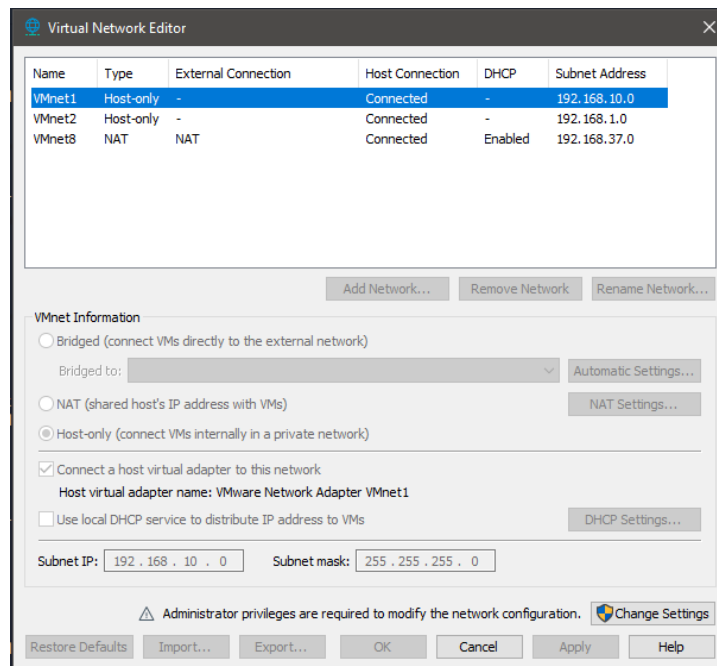


Take note!

On the Kali Linux VM, these respective network adapters are in the same order as displayed on Virtual Machine Settings.



Once you have a grasp of the specific network adapters used by the VM, you will next want to turn to the Virtual Network Editor also present in VMWare Workstation.



From inside the Virtual Network Editor, you can edit the respective virtual network adapters. Ensure the appropriate VMnets are configured properly, and one uses the appropriate **192.168.1.x/24** subnet.

The next step is to ensure the Virtual Machine has configured the IP addresses correctly. To show how to perform this step, we will use Kali Linux as an example. From a terminal, use the **ifconfig** command to inspect the network adapters. For the Hacklab machine, we are particularly interested in the eth0, eth1, eth2 interfaces.



Take note!

To only get the one interface when using the **ifconfig**, provide a specific network adapter as an argument of the command as shown below.

```
(kali@kali)-[~]
$ ifconfig eth1
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.253 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::20c:29ff:fe5e:a3c8 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:5e:a3:c8 txqueuelen 1000 (Ethernet)
    RX packets 739 bytes 956881 (934.4 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 409 bytes 41333 (40.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

All this information to align. IP addresses of Virtual Machine, VMnet, and Host should use the same network. Use the menus and commands found above to help identify the errors and resolve any networking issues.

2 EXPLORING THE VULNERABLE WEB APPS

2.1 EXPLORING THE BROKEN WEB APPS VIRTUAL MACHINE

As previously seen, the Broken Web Apps Virtual Machine is pre-installed with several deliberately vulnerable applications. This Virtual Machine is based on the OWASP Broken Web App Virtual Machine; however, a few alterations have been made.

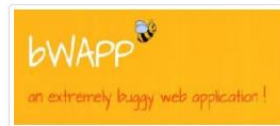
For the forthcoming exercise, ensure that the Broken Web App Virtual Machine is running (ideally from the “Booted” snapshot).

- Firstly, browse to <http://192.168.1.100>. Once there, you’ll be greeted by a front-page divided into 4 sections.

2.1.1 Main Training Applications

These applications are those that will be examined in the subsequent lab exercises. These applications have categorised vulnerabilities, allowing us to concentrate on specific types of flaws.

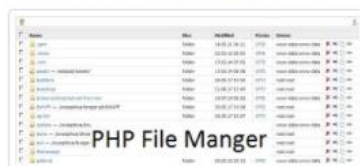
Main training applications.



2.1.2 The Backend of the Virtual Machine

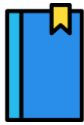
These applications allow us to quickly examine the web application’s server side, such as the source code.

Examine the virtual machine.



2.1.3 Realistic Web Apps

The next set of applications are deliberately vulnerable websites. The vulnerabilities found on these applications are uncategorised, so it should replicate trying to find them in the wild. These applications are titled as “realistic”, however, that would only be true if it was still 2009! Despite this tongue and cheek comment, remember that while many Internet-facing websites are modern, many legacy systems and intranet sites don’t!



Take note!

Old doesn't mean bad, we set out to teach you the underlying principles. Teaching these principles in more up-to-date software introduces complexities which adds complications to the learning process – Don't freight! you are exposed to modern technologies in the containerised web applications.

Realistic Web apps.



2.1.4 Others

Finally, the last of the sections is titled “Others”, this is web apps included for independent research activities.

Others.



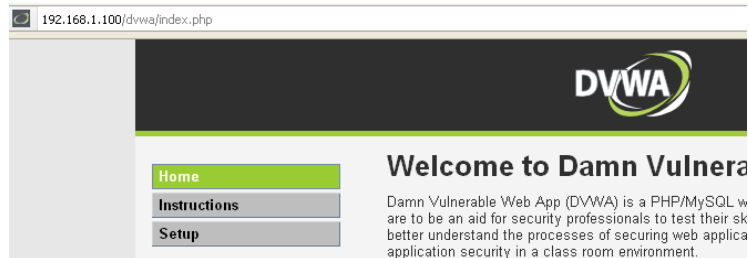
2.2 EXAMINING THE MAIN TRAINING APPLICATIONS

In this section, we will examine how to access each of the main training applications from a browser, as well as how to examine their PHP source code. The main applications we will be using are :-

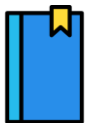
1. DVWA.
2. Mutillidae
3. bWAPP
4. SQLi Labs

For each of the following subsections, choose the appropriate menu item from the main web page.

2.2.1 Damn Vulnerable Web App “DVWA”



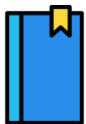
This excellent application was developed by Ryan Dewhurst (an Ethical Hacking graduate from Northumbria University). The application has been developed to illustrate common PHP web application vulnerabilities and their solutions.



Take note!

The DVWA credentials are **admin / password**.

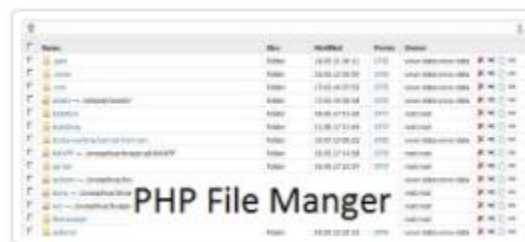
- Authenticate with the application and take time to explore the DVWA environment.



Take note!

The DVWA application has a Security menu, where the difficulty of exercise can be set to Low/Medium/High - simulating different skill/awareness of web coders.

As we use, test, and break this application, we may want to examine the underlying PHP code. This can be accomplished using the File Manager application linked on the main page.



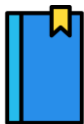
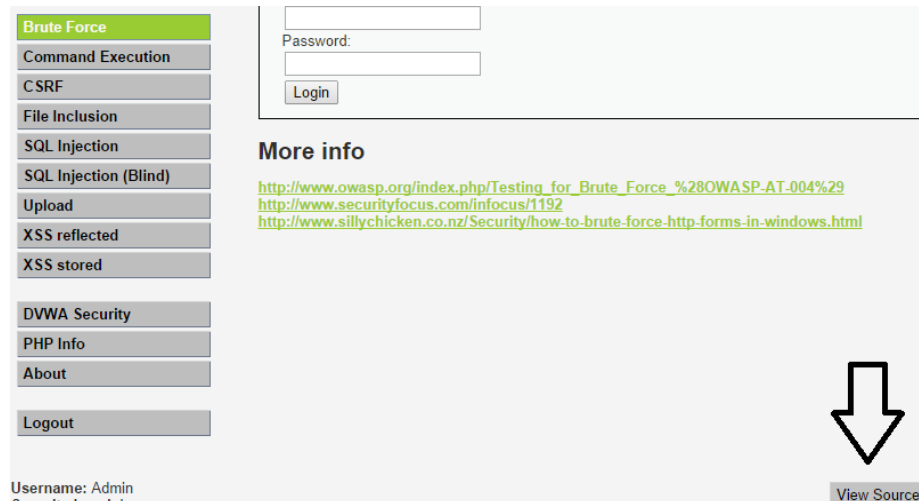
- Browse to the File Manager application, either through the link on the main page or from the URL below.

<http://192.168.1.100/filemanager/filemanager.php>

- On this application, browse to the dvwa folder. Once there, open the folder named vulnerabilities, and then the folder “brute”. This folder relates to the Brute Force vulnerability exercise. Examine the source folder and the files within it.
- During your exploration of these files, you will come across the PHP code responsible for altering the exercise’s difficulty.

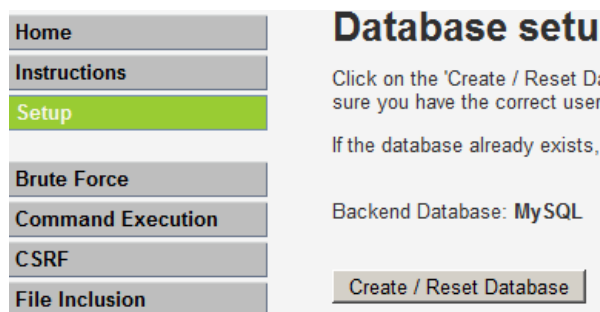
Going through this process to examine the source code of each page is rather time consuming. Thankfully, there is a much easier and straightforward alternative.

- In each exercise of the DVWA application, there is a button at the bottom of the page (as shown below) which links to the relevant source code.



Take note!

Some exercises involve modifying or destroying data, therefore, to complete other activities after these alterations, the database must be reset. This can be accomplished by using the options found inside the Setup menu, as seen below.



2.2.2 Mutillidae

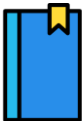
Mutillidae is a free, open-source PHP web application, like DVWA in that it is deliberately vulnerable. It also illustrates how the PHP code should be written.



- Navigate to the Mutillidae application from the main page, or from the URL below.

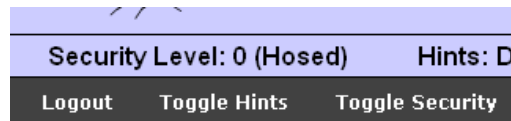
<http://192.168.1.100/mutillidae/>

- Register a user with the credentials **hacklab / hacklab**, then authenticate with this new account, and explore the application.



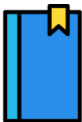
Take note!

Similar to DVWA, Mutillidae has a Security Level toggle, with the levels of 0, 1 and 5.



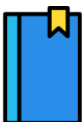
The underlying PHP code can be examined on the Virtual Machine, showing the effect of the Security Level toggle.

- Examine a PHP file in `/var/www/mutillidae` using the File Manager application. During your investigation of this file, you should see a series of select case statements at the top of the page. This is the code responsible for Security Level toggle.



Take note!

In Mutillidae, hints can also be toggled.



Take note!

Like DVWA, the underlying database for Mutillidae can be reset. Useful if we alter or delete user data.

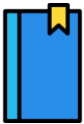
2.2.3 bWAPP

The Broken Web Application (bWAPP) is a deliberately vulnerable application with an abundance of activities in a categorised manner.



- Navigate to the bWAPP application from the main page, or from the URL below.

<http://192.168.1.100/bWAPP/login.php>



Take note!

The bWAPP default credentials are **bee / bug**.

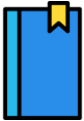
- Authenticate with the application and take time to explore the bWAPP menus and environment.
- Examine a PHP files in the `/var/www/bWAPP` folder using the File Manager application. During your investigation, you should come across a series of select case statements, like `Mutillidae`. This is the code responsible for Security Level toggle.

2.3 EXAMINING THE DATABASES.

Many of the applications on the virtual machine use "AMP" tech stack (i.e., Apache Web Server, MySQL and PHP). The Apache Web Server services the handles serving the various web applications, PHP is the language the applications are written in, and MySQL is the backend database. The MySQL database can easily be managed using the phpMyAdmin application.

- Navigate to the phpMyAdmin application from the main page, or from the URL below.

<http://192.168.1.100/phpmyadmin/>



Take note!

The phpMyAdmin credentials are **root / owaspbwa**.

- Authenticate with the application and take time to explore the menus and environment. From this viewpoint, you will be able to view, examine, and manipulate the underlying databases.

```
• .svn
• badstoredb (4)
• bricks (1)
• bwapp (4)
• citizens (1)
• cryptomg (3)
• dvwa (2)
• gallery2 (57)
• getboo (21)
• ghost (1)
• gtd-php (16)
• hex (1)
• information_schema (28)
• isp (1)
• joomla (88)
• mutillidae (11)
• mysql (23)
• nowasp (12)
• owaspbwa (90)
```

2.4 EXPLORING THE REST OF THE VIRTUAL MACHINE

The previous exercises have explored the main training applications which you will encounter over the course of the module, however, other applications are worth of exploring at this early stage. It is worthy highlighting the other applications not linked on the main page.

The OWASP “Broken Web Applications” virtual machine has a lot of different web applications installed.

From the main menu, (<http://192.168.1.100/>), browse the rest of the applications by clicking on the following image at the bottom of the page.



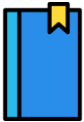
- Navigate to the owaspbwa application from the image at the bottom of the main page, or from the URL below.

<https://192.168.1.100/oldindex.html>

- Explore these applications and keep them in mind for future independent research.

3 USING PROXIES IN WEB APP PENETRATION TESTING

Proxies are a mainstay in web application penetration testing. Over the course of the module, you will make ample use of them to conduct a wide variety of activities. There are many different proxies that can be used – each will have slightly different features and applications. The following exercises concentrate on two of the most common proxies - OWASP Zap and Burp Suite.



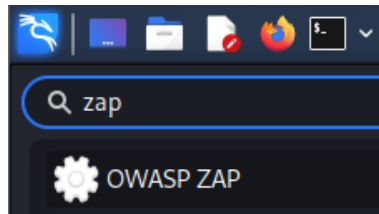
Take note!

For context, an associated “Using OWASP ZAP to Intercept Traffic” video is available wherever you obtained this lab exercise.

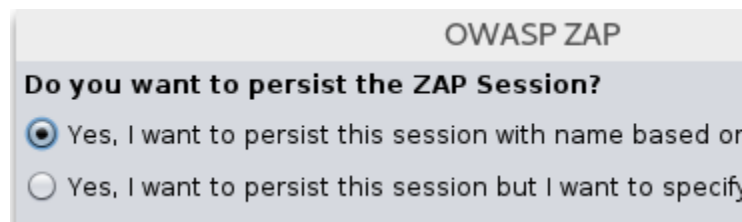
3.1 RUNNING ZAP

OWASP Zap proxy is a free and open source MITM proxy software. It allows viewing of HTTP and HTTPS and allows information to be intercepted and altered.

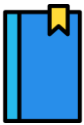
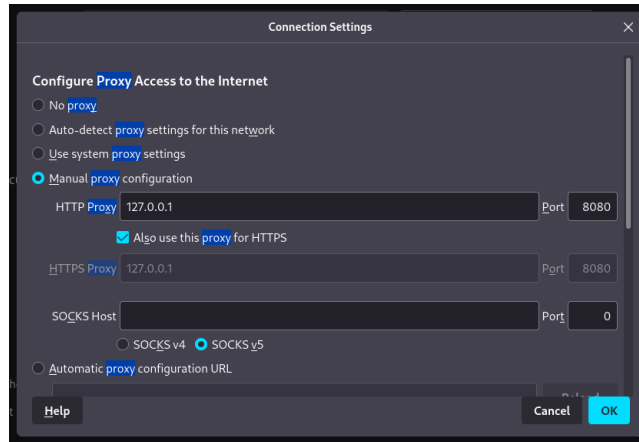
- In Kali Linux, run the OWASP ZAP proxy, this can be accomplished by searching for it in the menus, as seen in the Figure below.



- When prompted, Select the first choice regarding ZAP persistence.



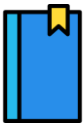
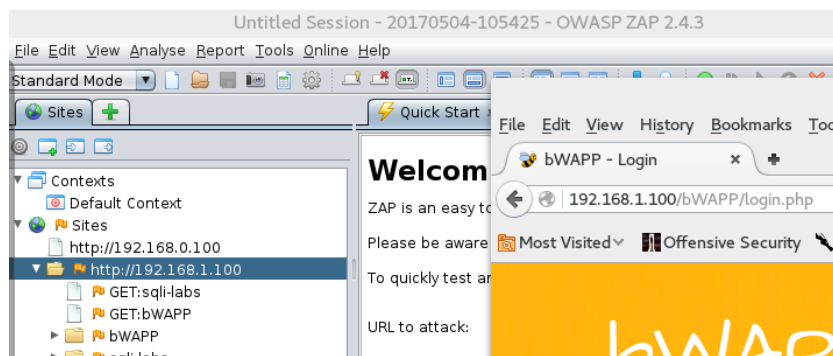
- By default, OWASP ZAP's proxy listens on port 8080, so we must configure our browser to send the web traffic to this port. From Kali's Firefox browser, change the settings to point towards ZAP's proxy as seen the Figure below.



Take note!

Ensure the “Also use this proxy for HTTPS” is enabled!

- After configuring the browser’s settings, navigate to any of the Broken Web Applications. Accepting any of the “I understand the risks” pop-ups and adding exceptions where necessary. Explore an application of choice for a while, clicking links to new pages, and then return to the OWASP Zap window. You should observe the Sites tab populating with the URLs you browsed.



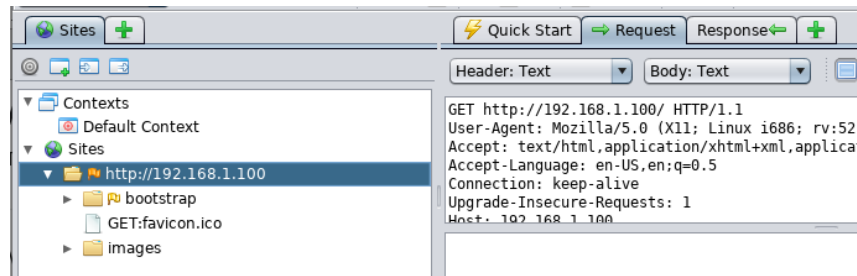
Take note!

If the Sites tab does not populate, it is likely due to your proxy configuration, give it another look to check over. If you are sure the configuration is correct, then it might be the pesky “ZAP HUD” feature.

Ensure the green icon, 3rd from the right on the taskbar at the top of the window, is inactive. It should look something like the picture below.

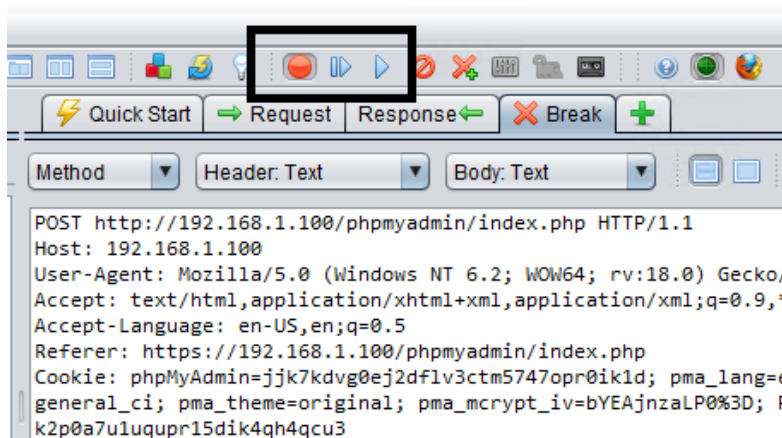


- Additionally, all requests and responses generated during your browsing have been captured. Examine both the Request and Response tabs.



3.1.1 Using Intercept in ZAP

In some scenarios, you may wish to intercept these requests and responses. To do so, you must stop them before they reach their destination - thankfully, this is bread and butter for a Man-in-the-Middle proxy. The red “record button”, alongside the pause and play buttons found on the application’s taskbar are used to control this intercepting feature.



While we touch on this feature in the module later, you may wish to test out the capability of this powerful feature now. To do so, follow the steps below.

- Set the Intercept on, therefore the red button should appear active. Then browse to a page, this will populate the request tab. Navigate to this tab, altering the request before sending it on and examining the response.



Caution!

Make sure to revert the browser’s proxy settings, or else you may complicate further exercises.

3.2 RUNNING BURP SUITE

Burp Suite is an enterprise level web application assessment tool and proxy. While we don’t have access to the paid Enterprise edition, we do have access to the free Community edition, which we will make ample use of in this module. The Community Edition does not have as many features as the Enterprise

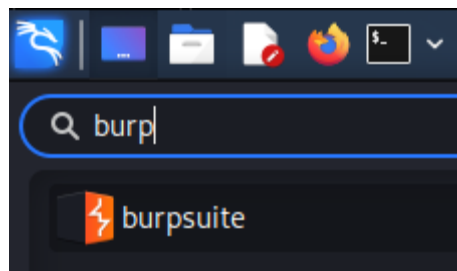
Edition, and the Community edition often lacks functionality found in the free alternative OWASP Zap. However, it is a wildly popular tool used in industry, so worthy of focusing our attention on.



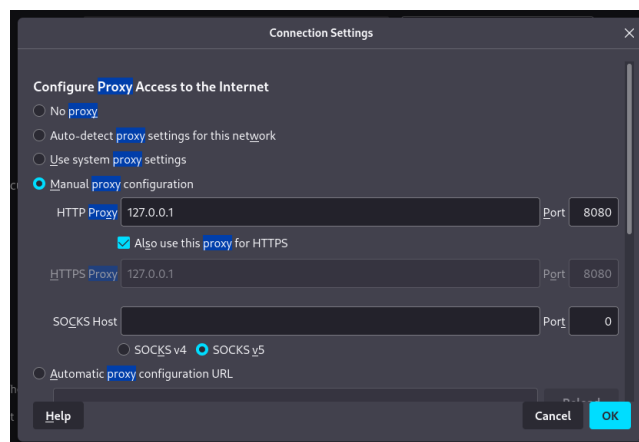
Caution!

Make sure OWASP ZAP is not running, as it will likely conflict with Burpsuite.

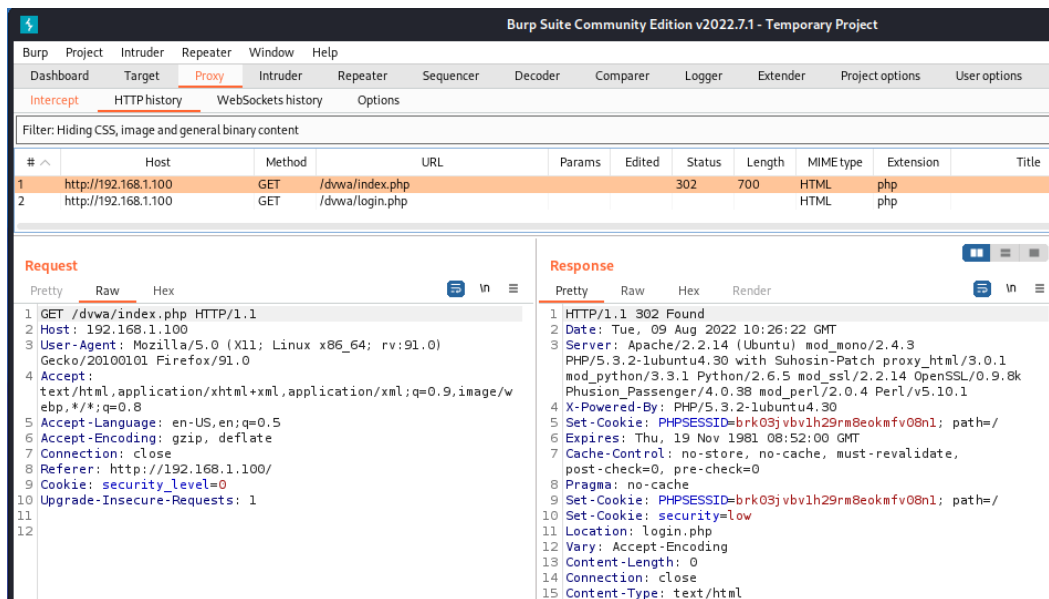
- In Kali Linux, run Burpsuite, this can be accomplished by either searching for it in the menus as seen in Figure below, or by entering the command Burpsuite.



- Then, you will have to configure the Firefox browser to point towards the proxy, as seen the Figure below. Thankfully, burpsuite's proxy also listens on port 8080, so you should not have to change the settings from the previous exercise.



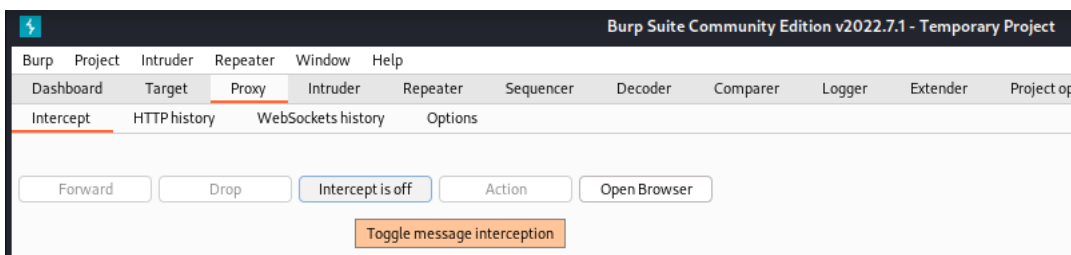
- After configuring the browser's settings, navigate to any of the Broken Web Applications. Accepting any of the "I understand the risks" pop-ups and adding exceptions where necessary. Explore an application of choice for a while, clicking links to new pages, and then return to the burpsuite window. You should observe the requests populating HTTP history tab within the Proxy window.



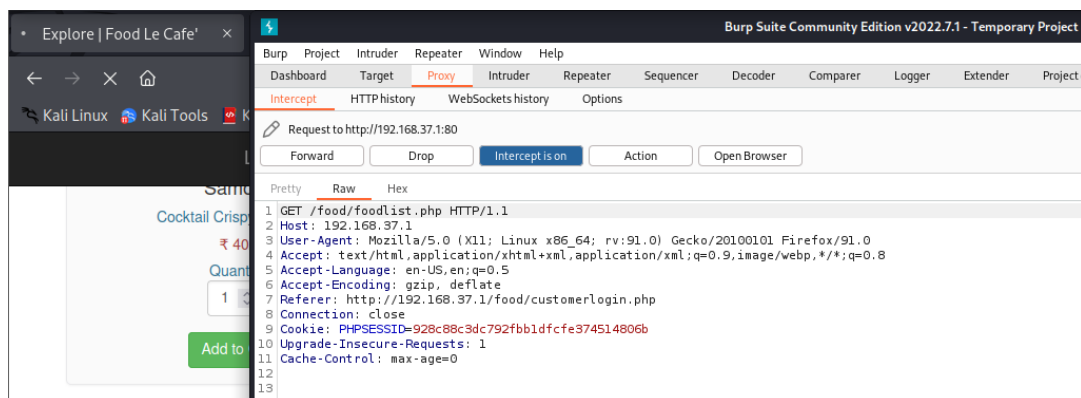
3.2.1 Using Intercept in Burp Suite

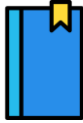
As mentioned previously, in some scenarios you may wish to intercept these requests and responses. To do so, you must stop them before they reach their destination - thankfully, Burp Suite's community edition has this functionality.

- On Burpsuite, navigate to the proxy tab, and the Intercept subtab as shown in the figure below.



- Toggle the intercept functionality by clicking the "Intercept is off" button to enable the proxy stopping destined packets. After doing so, navigate to a webpage or simply refresh the page you are currently on. It should result in the Burp Suite application looking like this.





Take note!

The Browser will show feedback indicating it is awaiting a response. On Kali's Firefox, this can be seen in the Favicon space in the tabs.

To demonstrate how powerful this tool is, we will examine what is possible to alter. To do so, we will be interacting with a website within the UniServerZ application. Run the UniServerZ from **D:\Others\UniServerZ** within the hacklab.

- Start by navigating to the food application located on your main Windows machine, likely **192.168.10.254/food**. Authenticate with the credentials **hacklab / hacklab**. Once you have authenticated, make sure to have toggled intercept on. Then, add the Chocolate Hazelnut Truffle to your cart. The interceptor should cat the request pictured below.

```
1 POST /food/cart.php?action=add&id=60 HTTP/1.1
2 Host: 192.168.37.1
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 94
9 Origin: http://192.168.37.1
10 Connection: close
11 Referer: http://192.168.37.1/food/foodlist.php
12 Cookie: PHPSESSID=928c88c3dc792fbb1dfcfe374514806b
13 Upgrade-Insecure-Requests: 1
14
15 quantity=1&hidden_name=Chocolate+Hazelnut+Truffle&hidden_price=99&hidden_RID=3&add=Add+to+Cart
```

- Examine the request closely. Notice the functions found on line 15. These functions include information about our dear Chocolate Hazelnut Truffle, including the quantity price, and more. We can alter this information as the request is sent in plaintext using the HTTP protocol. Change the hidden price value to anything you would like. However, I have heard free cake tastes much better.

```
1 POST /food/cart.php?action=add&id=60 HTTP/1.1
2 Host: 192.168.37.1
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 94
9 Origin: http://192.168.37.1
10 Connection: close
11 Referer: http://192.168.37.1/food/foodlist.php
12 Cookie: PHPSESSID=928c88c3dc792fbb1dfcfe374514806b
13 Upgrade-Insecure-Requests: 1
14
15 quantity=1&hidden_name=Chocolate+Hazelnut+Truffle&hidden_price=00&hidden_RID=3&add=Add+to+Cart
```

- After altering the hidden price value to 00, forward the request from Burp Suite to Firefox. Where the shopping cart page should load revealing the price is 0, as seen below.

Your Shopping Cart

Looks tasty...!!!

Food Name	Quantity	Price Details	Order Total	Action
Chocolate Hazelnut Truffle	1	₹ 00	₹ 0.00	Remove
Total			₹ 0.00	



Caution!

Make sure to revert the browser's proxy settings, or else you may complicate further exercises.