

Web Security – Day 1 – Lab Setup, Burp (Proxy), and Mapping the Web App

Contents

Web Security – Day 1 – Lab Setup, Burp (Proxy), and Mapping the Web App.....	1
Lab 1: Lab Setup, Vulnerable Web apps and Burp (Proxy)	2
1: The vulnerable web applications	2
2: Kali (the attacker).....	4
3: Burp in Windows VM	5
Lab 2 – Mapping the Web Application	6
1. Gathering information about the target.....	6
2. Spidering the Web Application	7
3. Discovering files and directories on web servers.....	8
Lab 3: Scanning Web apps for Vulnerabilities	11
Homework - Web Security Academy.....	12

Lab 1: Lab Setup, Vulnerable Web apps and Burp (Proxy)

The purpose of this first lab is for you to get familiarised with the lab setup.

Your computer contains 2 Virtual Machines (VMs) as follows:

- Kali (A Debian Linux machine for Penetration Testing and Offensive Security)
- Windows 10 (containing Apache web server, MySQL database, PHP and some vulnerable web applications)

Step 1: Open the VMs (You only need to do this step once)

From the Start menu of your PC, find **VMware Workstation Pro**. Click on **File, Open** and browse to the following VMs (one at a time), then open them:

C:\VMs\WebSecurity\Win10\win10.vmx

C:\VMs\WebSecurity\Kali-2024\kali.vmx

Note about network settings: For security reasons, all the VMs have their network setting set to **Host-Only**. This will allow each VM to be visible to the other VM without being exposed to the outside world.

Step 2: Running the VMs

In VMware, select each VM in turn and click on **"Power on this virtual machine"**. If prompted about whether the VM was **copied**, just click OK.

Step 3: Logging into the VMs

The login details for the VMs are as follows:

- **Windows 10:** user is **websecurity** and the password is **Secur1ty!**
- **Kali:** user is **kali** and the password is **kali**

1: The vulnerable web applications

The Windows 10 machine contains two web apps: DVWA (Damn Vulnerable Web Application) and Mutillidae. These are deliberately vulnerable web apps that have been created for learning purposes. Before we can access them, we first need to start up the Apache web server, the PHP interpreter and the MySQL (in this case MariaDB) database system. These have been installed as part of the XAMPP stack. The web apps themselves are stored as folders under **C:\xampp\htdocs**

Click on the **Start menu**, **XAMPP** and select **XAMPP Control Panel**. Once launched, start the **Apache** then the **MySQL** services.

If you get a warning from Windows firewall, simply click "allow access".

Open a web browser (Firefox) on <http://localhost> to see the XAMPP welcome page and verify that the web server is up and running. In the web page menu, click on **phpMyAdmin** to access a web interface for managing the databases. You will notice the **dvwa** and **mutillidae** databases. You can browse to see the tables they contain.

For example, under **mutillidae**, you can find an **accounts** table that contains the details of several users, and a **credit_cards** table which stores what it says!

As you can see, the database is exposed to anyone who has access to this machine. This is because XAMPP is meant to be deployed as a development (not production) environment. Open the **C:\xampp\readme_en.txt** file (**Right-click then Edit with Notepad++**) and read the lines from 118 to 126 about the security features that are missing in XAMPP. Now go to line 74 to see all the default accounts and passwords that are part of XAMPP.

Now let's access each web app in turn. To access DVWA, simply navigate to: <http://localhost/dvwa>
To login, use **admin** and **password** (can easily be guessed or brute-forced as you can see!).

For noting only: In the left-hand menu, click on **Instructions** and read through the set of notes. This contains useful information about how this app was made deliberately vulnerable so that you know what NOT to do when deploying a web app. Again, from the menu, click on **Setup / Reset DB** to see some configurations that make the web app deliberately open to attack. We will explore the rest of menu elements throughout the rest of this week when we cover each topic in turn (e.g., SQL injections, XSS...).

Click on **DVWA Security**. This is where we will be changing from one security level to the next to check whether an attack will still be possible under that level.

DVWA also allows you to check the PHP code used to secure (or not) against each type of attack.

For example, click **SQL Injection**. Scroll down the page and click **View Source**. A new window pops-up. Scroll down that window and click **Compare All Levels**. The window will now show the snippets of code corresponding to each security level.

Similarly, to access Mutillidae: <http://localhost/mutillidae>

On the left-hand menu, you find sub-menus for the multiple OWASP top 10 vulnerabilities, from 2017 down to older versions. Clicking through each option will take you to the actual page where you could perform an attack related to one of the vulnerabilities.

The main page has links to useful resources such as video tutorials...

At the top of the page, you have an indication of the current level of security (0 by default), and hints (enabled by default). To change to a more challenging setting, you can press the **"Toggle Security"** and **"Toggle Hints"** buttons. You can click the **Home** button to return to the home page, and login/register to the site (we'll do that in a later lab).

You can develop your own simple web apps to test certain features. All you need to do is make sure that your web app folder is under **C:\xampp\htdocs**

For example, let's create a folder there and call it **myapp**.

Within that folder, create an **index.html** file (for example using **Notepad++**):

```
<html>
<head>
  <title>MyApp</title>
</head>
<body>
  <h1>This is my test webpage</h1>
</body>
</html>
```

Your web page should now be accessible through <http://localhost/myapp/> or <http://localhost/myapp/index.html>

Instead of an html file, we can have a PHP file. Delete the previous file and create a new **index.php** file (under the same folder):

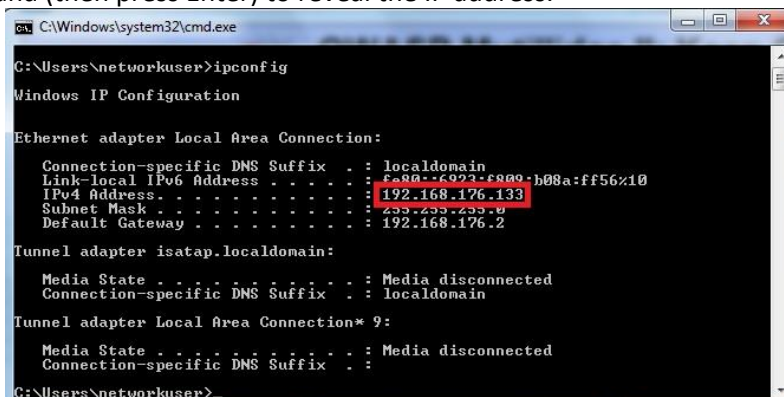
```
<?php
echo "Hello World";
?>
```

Your file should be available through <http://localhost/myapp/> or <http://localhost/myapp/index.php>

Now, let's find out the IP address of this machine so that we can access it from the Kali machine.

Click **Start** menu, under **Windows System** launch the **command prompt (cmd)**.

Type the **ipconfig** command (then press Enter) to reveal the IP address:



```
C:\Windows\system32\cmd.exe

C:\Users\networkuser>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : localdomain
    Link-local IPv6 Address . . . . . : fe80::6923:f899:b08a:ff56%10
    IPv4 Address. . . . . : 192.168.176.133
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.176.2

Tunnel adapter isatap.localdomain:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : localdomain

Tunnel adapter Local Area Connection* 9:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

C:\Users\networkuser>
```

Take note of the IP address value you get and use it in the rest of this lab (instead of the one I use)

2: Kali (the attacker)

In Kali, launch a web browser (top icon in left-hand menu). In the URL address bar, type in the IP address of your Windows VM (in my case 192.168.176.133). This should open the Apache web server welcome page.

Similarly, use the IP address to access the DVWA and Mutillidae web apps:

<http://192.168.176.133/dvwa> and <http://192.168.176.133/mutillidae>

HTTP requests/responses:

When you are browsing a web app, your browser is interacting with the web server using HTTP. We can see that using several tools. The easier one would be the browser's web developer tools. In Firefox' toolbar, select More Tools then select the Developer Tools. Select **Network** to open the Network Monitor.

A window will appear at the bottom of the browser. You may need to click the **Reload** button. You should now see the window populated with the GET Http requests made to the server. The reason why we see multiple GET requests is that although the initial request is for the root document index.html (or index.php) file, that file then links to other documents (css, js, jpeg...). Each of these resources will be fetched using a separate HTTP request.

Click on any GET request to see its details as well as the response from the web server. The following link (open it from your host machine) provides a comprehensive description of the interface:

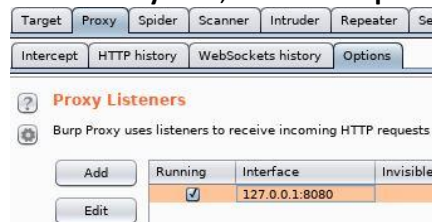
https://developer.mozilla.org/en-US/docs/Tools/Network_Monitor

Another way to interact with a web server would be to use a network tool such as Netcat (**nc** command) or **curl**. You can look these up in your own time if interested.

Working with an HTTP Proxy (Burp Suite):

A video walkthrough is available on Moodle, if you'd like to watch it before/while reading the following section.

In Kali, find and launch Burp Suite (from the top left menu). Click **Next** to start a temporary project, then click **Start Burp**. Start a new temporary project. Click the "**Proxy**" tab, then the "**Options**" sub-tab.



If you see a tick under "Running" then you are set.

Note: If there is no such tick, then this could be because the port is used by another service, in that case it's just a matter of clicking Edit and setting a different port number until you see a tick appear.

Now, go to your web browser and configure it to work with the Burp Suite proxy. In Firefox, select Settings in the toolbar. In the search box, type **proxy**. This will take you to "Network Settings". Select "Manual proxy configuration" and enter 127.0.0.1 for the proxy and 8080 for the port (or whatever port is set in Burp) then click OK:



From now on, when you try to access a web page, Burp Suite will act as a proxy, so you need to click on "**Forward**" (under the **Intercept** tab) to send the request across to the web server. Make sure "**Intercept is on**".

Give it a try by accessing any of the web pages on the DVWA or the Mutillidae web applications in Firefox. Spend a few minutes exploring the various tabs and options available in Burp Suite. We will make use of it in future labs.

At any time, you can instruct Burp Suite to stop acting as a proxy by clicking on "**Intercept is on**" to turn it off. You do NOT need to do this here.

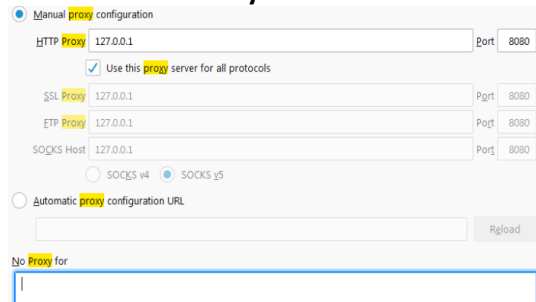
If you want to disassociate your browser from the proxy, you can go back to the browser preferences, and set the option "No proxy" instead of "Manual proxy configuration". **You do NOT need to do this here.**

3: Burp in Windows VM

Burp Suite is also installed in the Windows VM. This means you have the option of using it here instead of Kali, if you prefer using Windows to using Linux. All you need is to set it up to work with your browser in the same way described above. Give it a go.



When configuring the browser make sure you use the same port number (e.g., 8080), tick the “Use this proxy server for all protocols” and clear the text box under “No Proxy for”:



Now, in your web browser, go to the DVWA or Mutillidae web app and see how Burp intercepts the HTTP traffic.

Note for your information only: Initially, when working with Burp and Firefox on the same machine, Burp will not intercept requests from Firefox. To fix this issue I had to configure Firefox as follows:

1. In Firefox's URL bar I typed: **about:config** then clicked "Accept the Risk" button.
2. In the resulting page's search bar, I typed: **proxy**
3. Looked for "**network.proxy.allow_hijacking_localhost**" and double clicked it to turn it to "**true**".

Lab 2 – Mapping the Web Application

The purpose of this lab is to review techniques and tools for gathering information about a target web application. Go ahead and power on your Kali and Windows VMs. Launch XAMPP in the Windows VM, and start Apache and MySQL. As in Lab1, run an ipconfig command to get the Windows 10 VM's IP address. Use it instead of the one I use in this lab.

1. Gathering information about the target

There are two types of approaches:

- Passive reconnaissance: collects data from publicly available resources without directly interacting with the target web site. Data includes IP address, Domain name info, supported technologies, people working for the organisation (their contact details, social media profiles...)
- Active reconnaissance: collects data by directly probing the target web site to collect data such as DNS records, available services, open ports...

1.1 Passive reconnaissance:

Pick a web site of your choice (for example: www.itsecurityguru.org), and use the online tools below to gather information about it. Do this from your host machine because the VMs are not connected to the Internet.

- Whois Lookup (<http://whois.domaintools.com/>): find info out about the owner of the target...
- Netcraft (<https://sitereport.netcraft.com/>): shows technologies used on the target...

Discovering web sites on the same server:

One server can serve a number of websites, so gaining access to one website can help gaining access to others. For example, the Whois Lookup above would show that www.itsecurityguru.org is hosted on the same server as 60 other sites:

IP Address
104.26.14.185 - 60 other sites hosted on this server

A quick way of finding the web sites sharing the same IP address as your target, is to use the IP search operator in Microsoft's **Bing** search engine. Go to Bing in your browser and type in: **ip: 104.26.14.185**

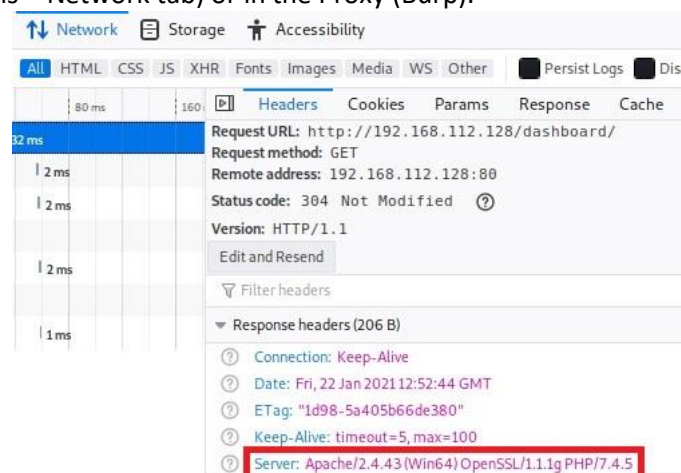
phpinfo

If your target's web server is not configured well, it will expose the phpinfo.php file that contains all sorts of configuration data including the name and version of the web server and its host operating system. For example, in Kali (attacker's machine) you can access: <http://192.168.176.133/dashboard/phpinfo.php>

Note: Make sure the browser in Kali is not configured to use the proxy from last week's lab. If it is taking a long time to load the page, switch to the Windows VM and visit: <http://localhost/dashboard/phpinfo.php>

Notice the name and version of the OS (Windows 10...), web server (Apache 2.4.43...), OpenSSL library (1.1.1), and PHP (7.4.5).

Technologies used by a web site can also simply be exposed by observing HTTP responses from the server (using the browser's web developer tools – Network tab) or in the Proxy (Burp).



Exercise 1: Search online how to stop Apache from advertising this information (server version, OS information...).
There is no need to implement this for now.

Homework (in your own time, not in this lab!): Use the CVE (Common Vulnerabilities and Exposures) database <https://www.cvedetails.com/> to check for vulnerabilities related to some of the above technologies. Can you find any? Does our web server use the latest version of each technology?
You can then (at home!) check exploit databases such as www.exploit-db.com (this will be blocked if accessed from RGU) to check for the existence of exploits against some of the vulnerabilities identified above.

1.2 Active reconnaissance

In Kali, open a command window and type the following command (use your Windows VM's IP address instead):

nmap -sC -sV 192.168.176.133

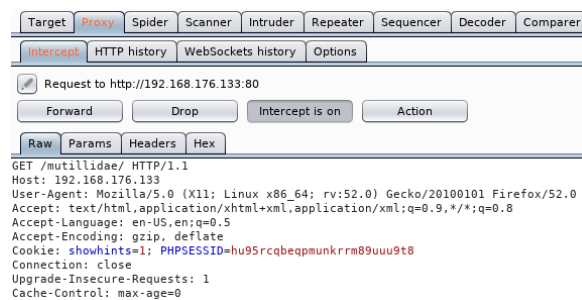
The tool nmap will give you details of each open port and running service with their corresponding version.

Warning: This type of scan is likely to leave evidence in the logs of your target machine, so be careful not to use this tool against live systems unless authorised to do so. Hackers will usually scan in stealth mode and using all sorts of detection evasion techniques.

Something else a hacker might do at this stage is website mirroring. This allows them to create a complete copy of the site's directory structure, links... so that they can probe the website offline at their leisure. Tools include HTTrack which can be installed into Kali.

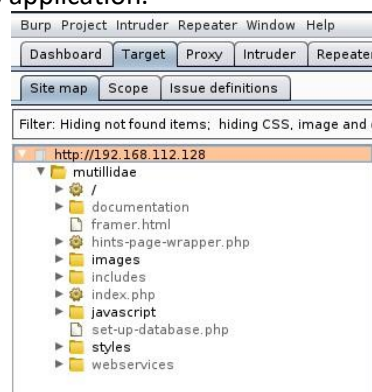
2. Spidering the Web Application

Start Burp Suite in Kali and configure Firefox to work with it. In your web browser, navigate to the mutillidae web application. For example (make sure you use the IP of YOUR Windows VM): <http://192.168.176.133/mutillidae>
When you go back to Burp, under the **"Proxy"** tab, then the **"Intercept"** tab, you should see that the HTTP requests have been intercepted. Click the **"Forward"** button until you see that the web browser has rendered the mutillidae web page.



Clicking the **"Intercept is on"** button will stop Burp acting as a proxy and the web requests going straight to the server.

In Burp, click on the **"Target"** tab to see that Burp captured the traffic to our web server and web application and has presented a sitemap of the mutillidae web application.



3. Discovering files and directories on web servers

In the Windows VM, go to `c:\xampp\htdocs\mutillidae` to see the content of the web app (folders and files). You will see that there are many files and directories that make up this web application. Compare these with the result of Burp spidering. [Can you identify any that were not picked up by Burp?](#)

Some of these files and directories are accessible through the web browser (via links and menu options), but some are not. However, if discovered, one can simply navigate to them directly.

Obviously, this is cheating, because we are not supposed to have access to the machine hosting the web server to be able to browse the web app folders and files.

However, we can use some tools to help us discover some of those files and directories that are not accessible through spidering. We do that by guessing the names of such files and directories using word lists.

We are going to use the OWASP **dirb** tool (directory buster!) to scan for files and directories in the website. dirb does a brute force based on a word list (one that you provide or the one that comes with the tool).

Open a Command window in Kali and type in: **dirb http://192.168.176.133/mutillidae/**

Note: here we used the default word list provided by dirb, which you can find in the **common.txt** file stored in the directory **/usr/share/dirb/wordlists/**. If you had your own wordlist, simply add it to the command above.

The tool is able to find a number of files and directories. For example, the **phpinfo.php** file, which we have encountered earlier, and should not be accessible on the webserver (but it is here deliberately):

<http://192.168.176.133/mutillidae/phpinfo.php>



System	Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
Build Date	Jan 6 2010 21:50:12
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini path)	/etc/php5/cgi
Loaded Configuration File	/etc/php5/cgi/php.ini
Scan this dir for additional .ini files	/etc/php5/cgi/conf.d
additional .ini files parsed	/etc/php5/cgi/conf.d/ldap.ini, /etc/php5/cgi/conf.d/mysql.ini, /etc/php5/cgi/conf.d/pdo_mysql.ini, /etc/php5/cgi/conf.d/pdo_mysql.ini
PHP API	20041225
PHP Extension	20060613

Another interesting file is the **robots.txt** which contains webpages that web developers do not want search engines (e.g., Google) to index: <http://192.168.176.133/mutillidae/robots.txt>

```
User-agent: *
Disallow: passwords/
Disallow: config.inc
Disallow: classes/
Disallow: javascript/
Disallow: owasp-esapi-php/
Disallow: documentation/
Disallow: phpmyadmin/
Disallow: includes/
```

As you can see, there are a number of directories and files that the web admin is trying to hide from search engines.

So let's see what is inside the passwords directory: <http://192.168.176.133/mutillidae/passwords/>

This reveals a file **accounts.txt** which when opened seems to contain info about usernames and passwords:

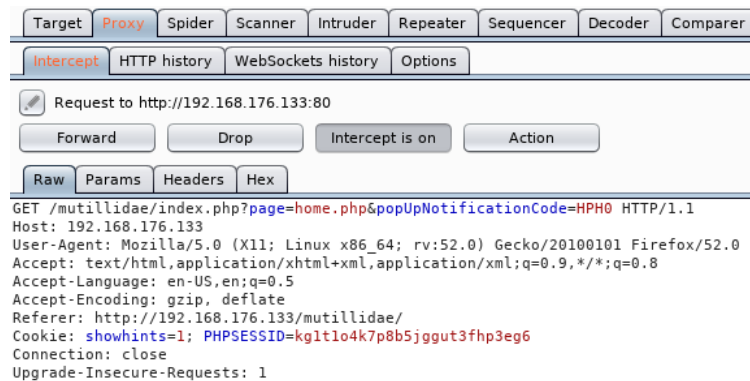
```
'admin', 'adminpass', 'Monkey!!!'
'adrian', 'somepassword', 'Zombie Films Rock!!!'
'john', 'monkey', 'I like the smell of confunk'
'ed', 'pentest', 'Commandline KungFu anyone?'
```

Exercise 2: Using the information revealed by the by robots.txt, retrieve information about the database.

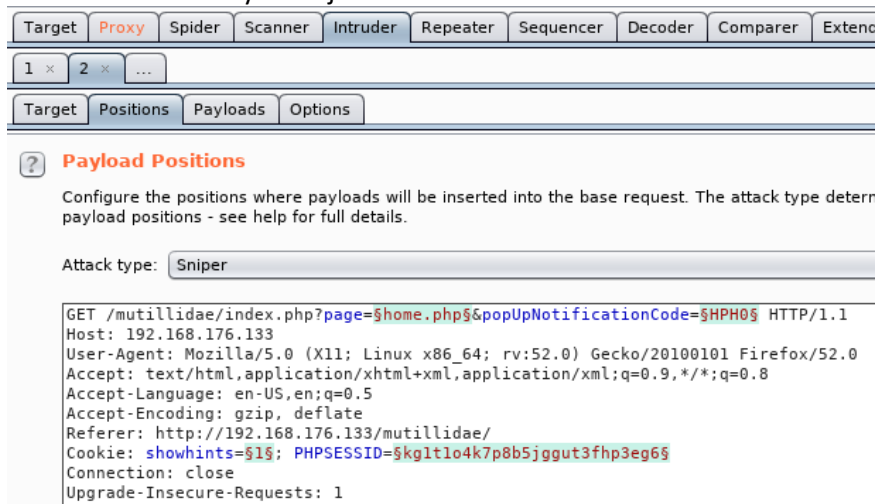
3.1 Using Burp Suite to discover content

Instead of using dirb, this time I would like you to use Burp Suite. The following steps show you how to do that.

Navigate to <http://192.168.176.133/mutillidae/> then, In Burp, make sure the **intercept is on**. Back to Mutillidae in your web browser, click on the **Home** link (top left in the menu). Now back to Burp to see it has intercepted that request:



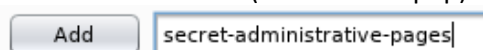
Right click anywhere inside this window and select **“Send to Intruder”**. You should see the **“Intruder”** tab turn orange to mean that it has received what you’ve just sent. Click on the **“Intruder”** tab then the **“Positions”** tab.



Notice the highlighted fields in the above request. Each parameter that can have a particular value is highlighted and enclosed within two \$ signs. The intruder allows you to change those values and send them to the web server. For example, the **“page”** parameter in the first line of code refers to the page name that was requested from the server. This is what we would like to brute force by providing our own word list. Therefore, we need to keep the page name between \$ \$ but clear the other parameters (simply select each in turn and click the **“clear”** button to the right). However, note that the \$\$ signs enclose both the page name and the .php extension. We want to change that so that only the actual name (home) appears between the two \$ signs. As shown below:

```
GET /mutillidae/index.php?page=$home$.php&popUpNotificationCode=HPH0 HTTP/1.1
Host: 192.168.176.133
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.176.133/mutillidae/
Cookie: showhints=1; PHPSESSID=kg1t1o4k7p8b5jggut3fhp3eg6
Connection: close
Upgrade-Insecure-Requests: 1
```

We now move to the **“Payloads”** tab. Here you can load your word list if you have any, or enter words directly using the interface. For example, we know from the previous exercise that Mutillidae has a page called **secret-administrative-pages.php**, so we can add that into our list (without the .php) and click Add:



We can try other words randomly (well, at least we think they might appear in common web applications anyway), such as admin, administration... Let's just add those three for now. Click the "Options" tab then click the "**Start attack**" button. If you get a warning message, just ignore it and click OK. This will not take long because we only have a few words in our list. You should see a result like this:

Results Target Positions Payloads Options					
Filter: Showing all items					
Request	Payload	Status	Error	Timeout	Length
0		200	<input type="checkbox"/>	<input type="checkbox"/>	53308
1	secret-administrative-pages	200	<input type="checkbox"/>	<input type="checkbox"/>	49543
2	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	138170
3	administration	200	<input type="checkbox"/>	<input type="checkbox"/>	44553

The result shows the response of the web server to the requests to the home page (on the first line), followed by a request to each of the pages we are trying to guess. Notice the size (length) of each response. Sometimes this can be a quick way of determining whether such page exists or not (actual pages tend to have bigger size than empty or error pages). Click on each line in turn then click the "Response" tab at the bottom of the page. You can then either see the results in "Raw" or "Render" mode, which will show the web page as rendered by the browser. Hence, both **admin** and **secret-administrative-pages** are actual pages that have been discovered.

We can now use a more extensive list of words that are commonly used in web development. There are many online resources that offer such lists. We are going to use a popular one called Fuzzdb and the actual file is called **WordlistSkipfish.txt**

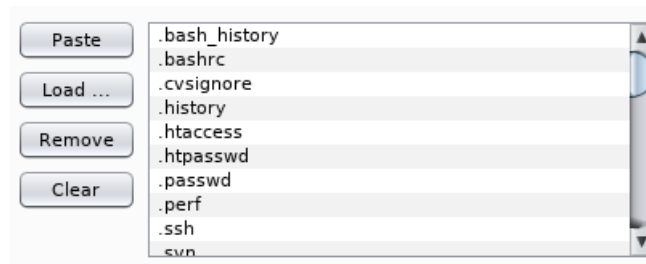
If you are working from your Windows VM, the file has been copied for your convenience under **C:\Temp** so you don't need to download it.

If you are using Kali, then you need to download it first (from your PC and not from Kali) by visiting <https://github.com/fuzzdb-project/fuzzdb>

Navigate through the following folders: **discovery/predictable-filepaths/filename-dirname-bruteforce**

Right-click on the **WordlistSkipfish.txt** file and save it (into your PC). You can now drag it from your PC (host machine) and drop it inside Kali (just on the desktop would be fine).

Repeat the same steps described above (intercept the request to mutillidae home page in Burp then send it to the Intruder). Under "Payloads", click "Load" and select the word list file:



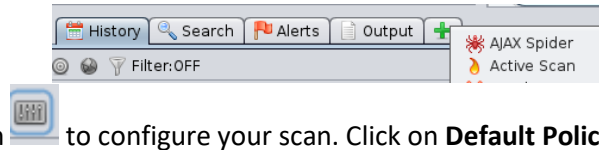
Start the attack, and watch the results coming back. However, the file contains 1918 entries, so no need to wait for all of them!


Note: If we had the (licenced) Burp Suite Pro Edition (which we don't!) then we wouldn't have needed to configure the intruder the way we did above. We would simply right-click our web site and select "Discover content".

Lab 3: Scanning Web apps for Vulnerabilities

We are going to use a tool called ZAP to scan our target website for vulnerabilities.

In Kali, from the top left menu, type **ZAP** in the search box then launch the tool. Keep the pre-selected option “No, I do not want to persist this session...” and click Start. When prompted to install updates, click Close. You can customise your scan in a number of ways. For example, in the middle of the owasp-zap screen click on the + sign and add Active Scan:



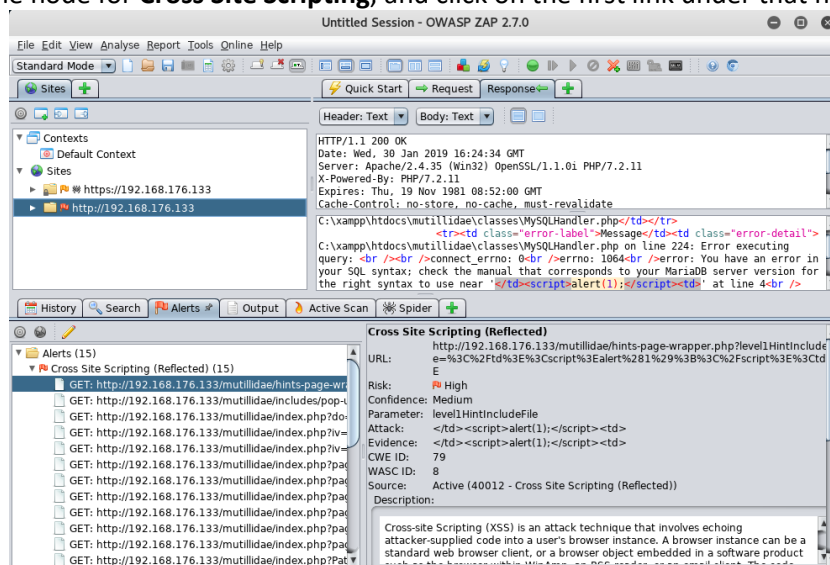
Now click on the following icon  to configure your scan. Click on **Default Policy** then click on **Modify**. Note the “Medium” values set for the alert threshold and strength. These can be customised for all aspects of the scan listed on the left hand screen. For example, click on Injection in the left-hand side window. You will see a list of all the types of injections that the tool will deploy. You can change the default values to anything you like. Say you were more interested in SQL injection, you can make sure the tool exerts more effort on this type of injection by setting the values to High. For now, just leave everything as **Default**.

Go back to the main window of ZAP, and let's type in the URL of our target: **http://192.168.176.133/mutillidae/** then click **Attack**.

The tool will first try to find all the URLs under this web site (using a Spider) then start scanning for vulnerabilities. This may take several minutes.

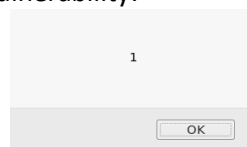
The Top-left window (under **Sites**) lists the discovered web pages (also available under the **Spider** tab), and the **Alerts** window displays the discovered vulnerabilities categorised in order of severity (red, orange, yellow and blue flags). You can expand the node associated with any vulnerability then click on any of the links below that to see details of how the vulnerability was discovered: the HTTP request sent and the response received.

For example, expand the node for **Cross Site Scripting**, and click on the first link under that node:



The tool was able to identify a web page that is vulnerable to XSS attack. See the HTTP request and response at the top, the content of the page displayed in the middle, the URL used to access the file, and a description of the vulnerability, how it was exploited and how you can fix it. You can also see a score of the risk (High) and how confident the tool is about the existence of this XSS vulnerability.

You can right-click the first link under “Cross Site Scripting” (highlighted in blue in the above screen), and select “Open URL in System Browser” to exploit this vulnerability:



Finally, please note that scanning tools (such as ZAP) do leave a trail in the web server logs, which can be useful for the security analyst to pick up. In your Windows VM, go to the **C:\xampp\apache\logs** folder and open the **access** log file and inspect it for indicators of use of ZAP (simply search for entries containing this keyword).

This is the end of today's Labs. You can now power off both virtual machines.

Homework - Web Security Academy

From your host machine (not the VM) or from your own laptop, sign up to PortSwigger (the company behind Burp Suite): <https://portswigger.net/web-security>

Once logged in, click “Academy” in the menu bar to find various learning paths, which include practical labs. Try to complete all available paths in your own time over the next few weeks.