Review of Group a by Group b

w x y z

. . .

Page limit: 18 pages.

Contents

1 Review of the External System

1.1 Background

Developers of the external system: *x', y', z', ...*

Date of the review: ...

1.2 Completeness in Terms of Functionality

Does the system meet the requirements given in the assignment?

Table 1: Completnes

No	Completness	
1	The system should allow the user to upload pictures	yes
2	The user can share his own pictures with other named users on a pictureby-picture basis	yes
3	The user can view his own pictures and pictures other has shared with him	yes
4	The user can comment on any picture he can view	yes
5	The user can view comments on any picture he can view	yes

1.3 Architecture and Security Concepts

Study the documentation that came with the external system and evaluation. Is the chosen architecture well suited for the tasks specified in the requirements? Is the risk analysis coherent and complete? Are the countermeasures appropriate?

1.4 Implementation

While looking at the solution, we found some vulnerabilities.

WEB SITE

- **Cleartext submission of passwords (login):** -passwords are sent over unincrepted connection, this may let someone listening to the network traffic aqcuire the user's password. this would especially be dangerous if the user uses a public wi-fi. even if in this case the web service does not contain sensitive data, a lot of people re use the same passwords on different platforms and even for online banking.

- **Access to images:** Anyone can access all the shared images on the fakestagram website on www.fakestagram.com:8080/img/"imagename" even without been logged in, the user has to enter the image name he wants to see,(one can surely guess some easy ones fx: me.jpg or dog.jpg) this violates the confidentiality requirement

- **Cross-site scripting (reflected)**

  The value of the username request parameter is copied into the HTML document as plain text between tags. The payload ¡script¿alert(1)¡/script¿ was submitted in the username parameter. This input was echoed unmodified in the application's response.

  This proof-of-concept attack demonstrates that it is possible to inject arbitrary JavaScript into the application's response.

  To solve this issue, a very good way is to validate user input. For example, personal names should consist of alphabetical and a small range of typographical characters, and be relatively short; A year of birth should consist of exactly four numerals; And so on.

SYSTEM

- **phpinfo() output accessible**

  Impact
  Some of the information that can be gathered from this file includes: The username of the user who installed php, if they are a SUDO user, the IP address of the host, the web server version, the system version(unix / linux), and the root directory of the web server.

  Solution
  Delete them or restrict access to the listened files.

- **php Multiple Vulnerabilities**
  Installed Version: 5.5.9

  #1
  CVE: CVE-2015-4148, CVE-2015-4147, CVE-2015-2787, CVE-2015-2348, CVE-2015-2331
  Impact
  Successfully exploiting this issue allow remote attackers to obtain sensitive information by providing crafted serialized data with an int data type and

to execute arbitrary code by providing crafted serialized data with an unexpected data type.

Solution
Upgrade to php 5.4.39 or 5.5.23 or 5.6.7 or later. For updates refer to http://www.php.net

- **#2**
  CVE: CVE-2015-4026, CVE-2015-4025, CVE-2015-4024, CVE-2015-4022, CVE-2015-4021

  Impact
  Successfully exploiting this issue allow remote attackers to cause a denial of service , bypass intended extension restrictions and access and execute files or directories with unexpected names via crafted dimensions and remote FTP servers to execute arbitrary code.

  Solution
  Upgrade to php 5.4.41 or 5.5.25 or 5.6.9 or later. For updates refer to http://www.php.net

- **#3**
  CVE: CVE-2015-3329, CVE-2015-3307, CVE-2015-2783, CVE-2015-1352

  Impact
  Successfully exploiting this issue allow remote attackers to cause a denial of service, to obtain sensitive information from process memory and to execute arbitrary code via crafted dimensions.

  Solution
  Upgrade to php 5.4.40 or 5.5.24 or 5.6.8 or later. For updates refer to http://www.php.net

- **#4**
  CVE: CVE-2015-6831, CVE-2015-6832, CVE-2015-6833

  Impact
  Successfully exploiting this issue allow remote attackers to execute arbitrary code and to create or overwrite arbitrary files on the system and this may lead to launch further attacks.

  Solution
  Upgrade to php version 5.4.44 or 5.5.28 or 5.6.12 or later. For updates refer to http://www.php.net

- **#5**
  CVE: CVE-2015-3330

  Impact
  Successfully exploiting this issue allow remote attackers to cause a denial of service or possibly execute arbitrary code via pipelined HTTP requests.

  Solution
  Upgrade to php 5.4.40 or 5.5.24 or 5.6.8 or later. For updates refer to http://www.php.net

- **php Multiple Remote Code Execution Vulnerabilities**
  CVE: CVE-2015-0273, CVE-2014-9705

  Impact
  Successfully exploiting this issue allow remote attackers to execute arbitrary code via some crafted dimensions.

  Solution
  Upgrade to php 5.4.38 or 5.5.22 or 5.6.6 or later. For updates refer to http://www.php.net

- **php Use-After-Free Remote Code Execution Vulnerability**
  CVE: CVE-2015-2301

  Impact
  Successfully exploiting this issue allow remote attackers to execute arbitrary code on the target system.

  Solution
  Upgrade to php 5.5.22 or 5.6.6 or later. For updates refer to http://www.php.net

- **php Use-After-Free Denial Of Service Vulnerability**
  CVE: CVE-2015-1351

  Impact
  Successfully exploiting this issue allow remote attackers to cause a denial of service or possibly have unspecified other impact.

  Solution
  Upgrade to php 5.5.22 or 5.6.6 or later. For updates refer to http://www.php.net

- **php 'serialize_function_call' Function Type Confusion Vulnerability**
  CVE: CVE-2015-6836

  Impact
  Successfully exploiting this issue allow remote attackers to execute arbitrary code in the context of the user running the affected application. Failed exploit attempts will likely cause a denial-of-service condition.

  Solution
  Upgrade to php version 5.4.45, or 5.5.29, or 5.6.13 or later. For updates refer to http://www.php.net

- **php 'phar_fix_filepath' Function Stack Buffer Overflow Vulnerability**
  CVE: CVE-2015-5590

  Impact
  Successfully exploiting this issue allow remote attackers to execute arbitrary code in the context of the PHP process. Failed exploit attempts will likely crash the webserver.

  Solution
  Upgrade to php version 5.4.43, or 5.5.27, or 5.6.11 or later. For updates refer to http://www.php.net

- **php Multiple Denial of Service Vulnerabilities**
  CVE: CVE-2015-7804, CVE-2015-7803

  Impact
  Successfully exploiting this issue allow remote attackers to cause a denial of service (NULL pointer dereference and application crash).

  Solution
  Upgrade to php 5.5.30 or 5.6.14 or later. For updates refer to http://www.php.net

- **php Out of Bounds Read Memory Corruption Vulnerability**
  CVE: CVE-2016-1903

  Impact
  Successfully exploiting this issue allow remote attackers to obtain sensitive

information or cause a denial-of-service condition.

Solution
Upgrade to php version 5.5.31, or 5.6.17 or 7.0.2 or later. For updates refer to http://www.php.net

- **Apache HTTP Server Multiple Vulnerabilities**
  CVE: CVE-2015-3185, CVE-2015-3183

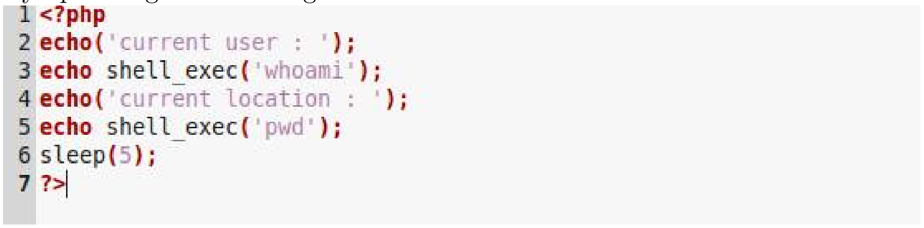
  Impact
  Successful exploitation will allow remote attackers to bypass intended access restrictions in opportunistic circumstances and to cause cache poisoning or credential hijacking if an intermediary proxy is in use.

  Solution
  Upgrade to version 2.4.14 or later, For updates refer to http://www.apache.org

1.5 Backdoors

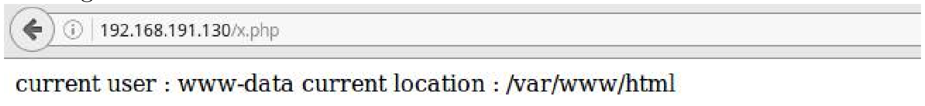
The system is running the website on a WildFly service, at port 8080. On the website (fakestagram), it is possible to upload images to the sever (and almost every other type of file), which will be saved, and can then be found at http://[IP]:8080/img/. There is also running a version of apache on the system. It is looking at the same location on the system that the files get uploaded to. Therefore it is possible to upload PHP files from fakestagram, on the WildFly service, and execute the PHP with apache.

By uploading the following PHP code:



```php
<?php
echo('current user : ');
echo shell_exec('whoami');
echo('current location : ');
echo shell_exec('pwd');
sleep(5);
?>
```

We can see from from what user that executes the code, and from where it is being done.



192.168.191.130/x.php

```
current user : www-data current location : /var/www/html
```

If defining a backdoor as a way to gain root access to the system. Then we didn't manage to exploit any.
But we do believe that by poking a bit more around in the system from the remote code execution with PHP, we might have gained root access.

1.6 Comparison

Compare your system with the external system you were given for the review. Are there any remarkable highlights in your system or the external system?