“Bonjour”
50 Defect Log

Last Revised: April 4, 2014

Team 7 Members:

Xiangyu Bu

Rishabh Mittal

Yik Fei Wong

Yudong Yang

Introduction

In the document, the first section introduces how to set up the project, and the second section is a list of 50 bugs that we suggest to catch.

Installation and Configuration

Setting up the Server

The following content introduces how to set up the server environment and have the program run.

Step 1: Setting up the Environment

Although the server code works on Windows + Apache + MySQL + PHP (WAMP) environment, we assume to set up a typical environment for PHP: Linux (Ubuntu) + Apache + MySQL + PHP. The necessary components and their versions in this environment are:

Component	Version	Note	Replacement
Ubuntu	13.10	The operating system	• Windows with IIS • For older versions of Ubuntu, one needs to get the PHP 5.5.8 or newer package to replace to PHP 5.3 package in Ubunu Precise repository.
PHP	5.5.8	The script engine. Either newest PHP5-MPM package (for Apache) or newest PHP5-FPM package (for Nginx) should work.	None.
Apache	2.4	Server http daemon	Nginx 1.5.8
MySQL	5.6	Database	MySQL 5.5
PHP extensions	-	php5-curl php5-gd php5-intl php-pear php5-imagick php5-imap php5-mcrypt php5-mysql php5-ming php5-ps php5-pspell php5-recode php5-	

		snmp php5-tidy php5-xmlrpc php5-xsl opcache enabled.	

An article of setting up Ubuntu + Nginx + PHP-FPM + MySQL can be found here: http://xybu.me/setting-up-a-ubuntu-server/

If Microsoft Windows is used, make sure set up smtp configuration correctly to send emails.

Step 2: Getting Source Code

1. Get the source code of server from project coordinator
2. Copy the code in `server/` to the www root directory of the web server. By default, it is the `DocumentRoot` in apache's `httpd.conf` file, or `/usr/share/www/html` for some versions of Nginx.
3. If the server is localhost, make sure `http://localhost/` will show the demo html UI, which is a basic UI for the interfaces.
4. In the file `config.inc.php`, modify the `$db_params` to contain correct credentials for connecting to the MySQL server.

Step 3: Importing Database Structure

There are three `.sql` files in `db/` directory. In each file, change the SQL server host to the actual host from "`localhost`", and

1. Connect to your MySQL server, log in, and create a dedicate user with limited privilege by importing the code in `db/1.mysql_user.sql`. A user named "`bonjour_demo`" with password "`demo`" is created
2. Import the code in `db/2.bonjour_demo.sql` to create the database `bonjour_demo`, and the necessary tables.
3. Import the code in `db/3.grant_priv.sql` as all privileges over the table `bonjour_demo`. Note that this demo user does not have permission over other databases.

Step 4: Basic Check

1. Register a user with the basic UI on `http://localhost/`, making sure the json text with access token is returned; otherwise error connecting to MySQL server will be shown. PHP extension `mysqli.so` (on Unix) or `php_mysqli.dll` is required.

2. If the token is generated successfully, then password hashing functions are proved to work.

Notes:

We have a demo server running already, but we don't accept overwhelming stress test or allow sending emails in it because of the host policy. To black-box test the demo server, contact the project coordinator Mingyuan Wang.

Installing Client App

The client app will be packed to the `.apk` file and can be tested on an Android phone for black box testing. Our testing profile file is Google Nexus 4 on AVD. Using other devices may reveal unexpected issues.

The app source code can be fetched from Github (see Step 2 of previous section). Install the Android plugin for Eclipse to edit it. Create a new Android virtual machine of version >= 4.0 to emulate it. A virtual SD card is necessary to store photos.

Download the ADT Plugin

Start Eclipse, then select Help > Install New Software.

Click Add, in the top-right corner.

In the Add Repository dialog that appears, enter "ADT Plugin" for the Name and the following URL for the Location:

https://dl-ssl.google.com/android/eclipse/

Download the Android SDK

http://developer.android.com/sdk/index.html#download

Notes

Home Activity: This is the homepage of the application after the user sign-in. The homepage will display 3 user profiles. The user profiles should display the name and the hobbies of the users shown. The "More" button currently have no functions. The Manager My Account button will direct the user to the user Profile interface. Currently the user Profile just display a default settings for demo purpose.

List of Suggested Defects

We have listed 18 typical defects for server side that should be revealed with reasonable effort, and 33 client-side defects.

Server-side Defects

#1	Randomly named user avatar images may collide and overwrite existing files.	Type	**Whitebox**	Severity	**3**
Output before seeding	When uploading user images, the file names are named following this rule: public/upload/%s-%d.png, where %s is a 20-character random string and %d is the timestamp of the upload event. There is little chance the files collide.				
Output after seeding	When uploading user avatar images, all were saved in public/upload/ directory with a random name. This means the old files with the same random name, if any, will be overwritten.				
Suggested Correction	In class.User.php, change the naming rule back to the one before seeding.				

#2	User profile can contain code that allows for XSS attack.	Type	**Black box**	Severity	**2**
Output before seeding	The function `filterHtml` in `Core` class will filter out any HTML or Javascript code contained in the fields submitted. So the code will be shown as text and will not be executed.				
Output after seeding	The user profile fields will have the original text sent by the sender (including a faker program).				
Suggested Correction	In `include/Class.Core.php`, finish the filterHtml function to filter out special characters like "`&`" to "`&`", "`>`" to "`>`", "`<`" to "`<`", etc.				

#3	Matching data is subject to injection attack.	Type	**White & Blackbox**	Severity	**2**
Output before seeding	The User class filters latitude, longtitude, and desired distance attributes. If they are not numerical or of invalid range, the matching request will be refused.				
Output after seeding	The request will be sent directly to SQL query string. Thus, if one uses faker to build special matching requests like "`" OR TRUE`" or something as value of latitude and such, server will crash.				
Suggested Correction	In the matching part of `action.php`, after decoding the data, check if the values are numerical or not. If not, refuse the request.				

#4	The `sendEmail` function is disabled.	Type	**Black box**	Severity	**1**
Output before seeding	The `sendEmail` function in `Core` class is a wrapper of several Linux sendmail binaries. It will choose a proper one to handle PHP send email requests.				
Output after seeding	No email will be sent regardless of the situation.				
Suggested Correction	In the function body of `sendEmail`, redirect the arguments to php built-in `sendmail` function. *Note: the demo server disables sendmail because of hosting policy.*				

#5	User password allows for length 33.	Type	**White box**	Severity	**3**
Output before seeding	Server will check the length of the password, and ensure the length to be no less than 6 and no more than 32.				
Output after seeding	Server will allow for a string of length 33 as password.				
Suggested Correction	In `class.Core.php`, in `isValidPassword` function, change the number 33 to 32.				

#6	Server will reveal debug information.	Type	**White box**	Severity	**1**
Output before seeding	Server will hide all errors, failures, and other exceptions from client, but record such problems in server log.				
Output after seeding	Server will print all errors directly to stderr, which is actually the browser or HTTP requester. For a production branch this is dangerous and will be a serious mistake.				
Suggested Correction	Turn off debug mode in `config.inc.php`.				

#7	When database server can't be connected, the user credentials for the database will be revealed.	Type	**Black box**	Severity	**1**
Output before seeding	If such connection failure happens, error will be written to `php_error.log`, and the execution is terminated, leaving the page blank.				
Output after	The connection failure will be printed directly on the page, including the database host and port number, username, and password, and the				

seeding	database name.
Suggested Correction	In function `connect()` of `class.Database.php`, change the code when the database object is not instantiated to `die()` directly, instead of printing the details. Or change `php.ini` to not reveal errors.

#8	Server bypasses the client's desired range when performing matches.	Type	**Black box**	Severity	**2**
Output before seeding	Client will send a parameter indicating the desired range, and server searches in the database for those who are in such range ("`WHERE distance <= :desiredRange`").				
Output after seeding	The `WHERE` condition was removed. Thus server will always search for users regardless of the range limit.				
Suggested Correction	Change `class.User.php` to add the WHERE condition back.				

#9	Possible problem of letting username and email fields be used interchangably.	Type	**Whitebox**	Severity	**3**
Output before seeding	When handling the request, server only matches username field sent to the server with user_email field in the database.				
Output after seeding	Server matches the username field with either user_name or user_email field. Thus if the username happens to another person's user email, things can go wrong.				
Suggested Correction	Change the SQL queries in `class.User.php` to only match user email. Since server blocks invalid username or email string, black box will not find the logical problem.				

#10	Password retrieval token never expires.	Type	**White box**	Severity	**2**
Output before seeding	The password retrieval token expires in one day.				
Output after seeding	The password retrieval token will never expire because it is hashed with base seed being username. i.e., as long as the username does not change, the token will not expire. Attackers can retrieve such tokens and usernames to conduct attacks.				
Suggested Correction	Change the hash base to user_lastActiveTime or other volatile or frequently changed fields.				

#11	User access token does not expire	Type	**White**	Severity	**2**

	unless the user logs out.		**box**		
Output before seeding	The user access token automatically expires if the last active time is a time older than one month.				
Output after seeding	User access token is always valid unless the user logs out when the last active time, which is the base of tha hash, is refreshed.				
Suggested Correction	In `class.User.php`, add check for time difference bewtween `now()` and `user_lastActiveTime` which returns false when such value is larger than 30 (days).				

#12	The original message or error prompt, is written in a relatively vague way from user's perspective.	Type	**Black box**	Severity	**3**
Output before seeding	The message will contain the error reason and possible fix.				
Output after seeding	The message is mostly vague error reason, which depends on the client to parse to user-friendly sentences.				
Suggested Correction	Rewrite all error messages in `User` class and `action.php`. This is a usability issue.				

#13	The error messages are loosely written in the script and are hard to maintain.	Type	**White box**	Severity	**3**
Output before seeding	There is a class that handles parsing a 2-tuple of error event and error number to an error message.				
Output after seeding	The class is removed and all its accesses are replaced with very briefly rewritten sentences. This causes severe maintenability issue when the set of defined errors gets increasingly large.				
Suggested Correction	1) Remake such `Dictionary` class and instantiate it in `Core`. 2) Let `Core` fetch the error message from `Dictionary` instead of getting messages as hard-coded parameters. This is a maintenability issue.				

#14	When the matching list of users is long, the response will be extremely long with the information of all matching users included.	Type	**Black box**	Severity	**1**
Output	Before the bug injection, only at most 20 users ordered by last active				

before seeding	time are listed in the response. Also only the email field of the users is revealed.
Output after seeding	By listing all matching users and including all their profiles, the response will be very large and is likely to cause network issue on mobile network, and the client may halt rendering the list. Also the user information is leaked.
Suggested Correction	Change the SQL query in `getMatching` function in `User` class so that: • Limit the maximum number of matches (records) (append `LIMIT 20;`) • Fetch only the user_email column

#15	Attackers can try to brute-force the user password because there is no ceiling for the number of password failures.	Type	**Black box**	Severity	**1**
Output before seeding	When the password is typed wrong for five times, the user account will be locked for 24 hours.				
Output after seeding	The user can try unlimited times to find the correct password by brute-force.				
Suggested Correction	Re-add the `user_locked` field in database or PHP session, and add a checker that • When the user tries wrong password, record this user in PHP session, setting the value to be 1 • Every time the user tries a wrong password, increment this value • When the number reaches 5, lock the user by setting `user_locked` for the user record to be current date and time, and forbid accessing this account until `user_locked` is an older timestamp compared to `now()`.				

#16	Server will not forbid flood requests.	Type	**Black box**	Severity	**1**
Output before seeding	Before this change, the main server will be able to detect *abnormally frequent* requests and forbid the requester for a period of time, thus protecting the server from DDoS attacks.				
Output after seeding	The protection is removed, so the server will try its best to serve every request it receives. Because there is no ceiling for the number of requests a user can send in a period of time, server will suffer from DDoS attacks and thus fail stress test.				
Suggested Correction	The way to enable such server-wide check for flood requests is included in the article about setting up server.				

#17	There is no built-in caching mechanism in the code at all.	Type	**White box**	Severity	**3**
Output before seeding	We did have some caching mechanism in Database class, which caches the queries according to their types. We also use opcache to cache the binary code of php.				
Output after seeding	When the server load is low, there is no obvious difference in performance, but as the size of database and the complexity and frequency of queries grow, the performance punishment for not having a cache will be very obvious.				
Suggested Correction	It is so atypical that a server side code does not any caching mechanism. With cache extension like `php5_apcu`, build some caching reasonable mechanisms in `Database` class. *The severity level is 1 when the load is heavy.*				

#18	Internal function `Core::isValidDate(str)` is subject to array out-of-bound bug.	Type	**White box**	Severity	**3**
Output before seeding	The function first checks if the string `str` has format YYYY-`mm-dd` where Y, m, d are digits, and then check if YYYY is a valid year, mm is a valid month, and dd is a valid day of the month.				
Output after seeding	The function splits the string using delimiter "-", and check if YYYY, mm, dd forms a valid date. Thus if the string has less than two "-"s, PHP will throw invalid array index access error.				
Suggested Correction	Check if the string has two **"-"** and keep the remaining code should fix the problem.				

Client-side Defects

#1	The passwords sent by client are not encrypted.	Type	**Black**	Severity	**1**
Output before seeding	The password was encrypted before sending so that even if a hacker gets the HTTP packets, the data is mostly useless to him.				
Output after seeding	In HTTP POST data of events like login or registration, the original password will show up. E.g., if the `password` is "`123456`", there will be a field like "`password=123456`".				
Suggested Correction	In `APIHandler` class, add `md5` encryption to the related functions like `login` or `register`.				

#2	The ListView in home activity cannot be displayed correctly	Type	**Black**	Severity	**1**
Output	The ListView should be displaying the user icon and several user info				

before seeding	
Output after seeding	The ListView display nothing
Suggested Correction	Modifying the home activity class, adding adapter

#3	The server shutdown caused client response string to null and handle the string incorrectly	Type	**Black box**	Severity	**1**
Output before seeding	When the server shutdown, the null value of the response string is correctly handled and a login error message showed on the screen				
Output after seeding	When the server shutdown, the null value caused NullPointerException and ignored by the client.				
Suggested Correction	Add the correct null value check before accessing the response string				

#4	The network availability is not checked by the client, if a network is not available, it will cause the crash of the app.	Type	**Black box**	Severity	**1**
Output before seeding	If the network is not available, the client will prompt the message and login or register cannot be executed.				
Output after seeding	If the network is not available, the client will still try to access the network and cause the client crashed.				
Suggested Correction	Use the network availability method to check the network before trying to access the network.				

#5	In ListView layout, the TextView displaying details of hobbies may overlap with the User Image if the list of hobbies are too long.	Type	**Black box**	Severity	**3**
Output before seeding	The hobbies should not overlap with the user Icon				
Output after	The hobbies overlap with the user Icon if it is too long				

seeding	
Suggested Correction	Modify the position of TextView of hobbies

#6	The line overlapped with the top of ListView.	Type	**Black box**	Severity	**3**
Output before seeding	Overlapping should not be happening				
Output after seeding	Overlapping happened				
Suggested Correction	"android:layout_below="@+id/textViewLine"				

#7	The button will disappeared if the username is too long	Type	**Black box**	Severity	**2**
Output before seeding	The button should not disappear if the username is too long. It will overlap with the username.				
Output after seeding	The button is being pushed out of the boundary due to the wrong align.				
Suggested Correction	Modified android:layout_marginRight=”21dp”				

#8	The ListView cannot display the last item correctly	Type	**Black box**	Severity	**1**
Output before seeding	The ListView should display the same contents for every items.				
Output after seeding	Some contents is being cut-off due to the size of ListView				
Suggested Correction	Modify the size(width, height) of the ListView				

#9	The orientation of the signupActivity does not change to the horizontal when the orientation of the phone change to the horizontal	Type	**Black box**	Severity	**3**
Output	The signupActivity changed to the horizontal when the phone changed				

before seeding	to the horizontal.
Output after seeding	The signupActivity does not change the direction when the phone changed to the horizontal.
Suggested Correction	Add the code to handle the orientation changes.

#10	In ListView layout, the TextView displaying details of hobbies may overlap with the User Image if the list of hobbies are too long.	Type	**Black box/ White board**	Severity	**2**
Output before seeding	They should not be overlapping and should be displayed correctly				
Output after seeding	They are overlapped with each other				
Suggested Correction	Modified by using align.				

#11	In the Sign Up Upload Fragment, when no image on the phone, the app does not have error message shows up	Type	**Black box**	Severity	**3**
Output before seeding	If no image on the phone, a message box should show up and tell the user to upload some photos to the phone				
Output after seeding	If no image on the phone, nothing will show up and signup activity cannot be finished.				
Suggested Correction	Add a method to show up a message box in the signup fragment when there is no image on the phone.				

#12	The user_profile is not displaying the correct user information	Type	**Black box**	Severity	**1**
Output before seeding	The user_profile should be able to display the username and hobbies.				
Output after seeding	The user_profile just display the default output.				

Suggested Correction	Should implement adapter to do so.

#13	The signup fragment does not check the validity of the email address	Type	**Black box**	Severity	**2**
Output before seeding	If the email is in the incorrect form, an error message will show up.				
Output after seeding	If the email is in the incorrect form, no error message will show up and the client will send the incorrect user email information to the server.				
Suggested Correction	Add the email validity check before sending the information to the server.				

#14	The signup fragment does not check the age of the user	Type	**Black box**	Severity	**3**
Output before seeding	The signup fragment will show up an error message when the user age is in the incorrect form				
Output after seeding	The signup fragment will not show up any error and incorrect user age will send to the server.				
Suggested Correction	Add the age validity check before sending the information to the server.				

#15	The signup fragment does not check whether the retyped password equals to the password	Type	**Black box**	Severity	**2**
Output before seeding	If the password is not equal to the retyped password, an error message will show up				
Output after seeding	If the password is not equal to the retyped password, no error message will show up				
Suggested Correction	Check the equivalence with the password and the retyped password				

#16	The default userIcon should be bonjour_icon.bmp	Type	**Black box**	Severity	**3**
Output before seeding	The default userIcon is bonjour_icon.bmp				

Output after seeding	The default userIron is wrongly set to nouser.jpg
Suggested Correction	Modify the reference to bonjour_icon.bmp

#17	The signup fragment does not check if the gender is set or not	Type	**Black box**	Severity	**2**
Output before seeding	If no gender is chosen, an error message will show up and the user cannot click the register the account button				
Output after seeding	If no gender is chosen, no error message will show up and the user can click the register button and cause the client crashed				
Suggested Correction	Check the validity of gender				

#18	The signup fragment does not check whether the retyped password equals to the password	Type	**Black box**	Severity	**2**
Output before seeding	If the password is not equal to the retyped password, an error message will show up				
Output after seeding	If the password is not equal to the retyped password, no error message will show up				
Suggested Correction	Check the equivalence with the password and the retyped password				

#19	In the SQLHandler getAllFriendsByUserId method, the cursor is not moved to the first, so data can be obtained from this method	Type	**White box**	Severity	**2**
Output before seeding	The getAllFriendsByUserId method can return correct information from the database.				
Output after seeding	The getAllFriendsByUserId method cannot return the correct information.				
Suggested Correction	Move the cursor object to the first entry.				

#20	HomeListLayoutActivity.java is not doing anything and has no uses	Type	**White box**	Severity	**3**
Output before seeding	This class should be removed after modifying the the implementation of HomeActivity.class				
Output after seeding	This class is not being removed				
Suggested Correction	Remove the class				

#21	The SQLHandler. addFriend method does not check the Bundle is null or not	Type	**White box**	Severity	**3**
Output before seeding	If a null value passed to the addFriend method, it will not be executed				
Output after seeding	A null value is not checked and will cause a NullPointerException when a null value passed to this method.				
Suggested Correction	Check the null value before accessing it				

#22	SQLHandler.addFriend method needs a writable database, but it obtains a readable database	Type	**White box**	Severity	**2**
Output before seeding	The method will correctly add the friend to the database				
Output after seeding	The method cannot add the friend to the database and cause the crash of the app				
Suggested Correction	Change the readable database to writable database.				

#23	Some of the content may not be displayed correctly in other devices	Type	**Black box**	Severity	**3**
Output before seeding	The display should support most of the devices				

Output after seeding	Half of the button at the bottom of login interface is not displayed in 320x480 resolution
Suggested Correction	Modify the layout to make it support as many devices as possible

#24	SQLHandler. setUserAccessToken uses an integer value to represent a user id, but a user id can be a long value	Type	**White box**	Severity	**3**
Output before seeding	If a user id is larger than an integer value, it will update the database correctly				
Output after seeding	If a user id is larger than an integer value, it causes an overflow and incorrect value will be inserted to the database.				
Suggested Correction	Change the userId parameter to the long type.				

#25	The coding in HomeActivity.java is messy and is hard for maintenance	Type	**White box**	Severity	**3**
Output before seeding	The code should be organized well.				
Output after seeding	The code is messy and hard to maintain.				
Suggested Correction	Re-organize the code.				

#26	SQLHandler. getUserAccessToken method uses an integer to represent the userId, but a user id can be a long value	Type	**White box**	Severity	**3**
Output before seeding	If a user id is larger than an integer value, it will update the database correctly.				
Output after seeding	If a user id is larger than an integer value, it causes an overflow and incorrect value will be inserted to the database.				
Suggested Correction	Change the userId parameter to the long type				

#27	SQLHandler. getUserNameByUserId method should obtain a readable database, but a writable database is obtained.	Type	**White box**	Severity	**2**
Output before seeding	The method will obtain the username correctly by the user id.				
Output after seeding	The method cannot obtain the correct username and sometimes causes a crash.				
Suggested Correction	Obtain a readable database instead of a writable database.				

#28	In LoginActivity class, it checks if the length of the password less than 4, but the password should contain at least 6 characters	Type	**White box**	Severity	**2**
Output before seeding	If the password is less than 6, an error message will show up				
Output after seeding	If the password is less than 4, an error message will show up, but if the length of a password is 5 characters, it will not show up the error message.				
Suggested Correction	Change the checking condition to the 6				

#29	In the signup activity, the age field does not restrict the length of the field	Type	**Black box**	Severity	**3**
Output before seeding	If the age field contains more than three characters, it will restrict the text field and no more character can be inputted to the field.				
Output after seeding	Any length of the age can be typed into the text field.				
Suggested Correction	Write the condition to check the length of the age text field.				

#30	SQLHandler.addUser cannot insert the url encoded username to the database	Type	**White box**	Severity	**3**

Output before seeding	The username and password should be decoded before inserted into the database.
Output after seeding	Encoded username and password are trying to insert into the database and cause the Database exceptions
Suggested Correction	Decode the username and password before inserting into the database.

#31	SQLHandler.addUser does not add quotes between a query string and causes a SQL syntax problem	Type	**White box**	Severity	**2**
Output before seeding	A user should be added to the database correctly.				
Output after seeding	The email address contains a '@' symbol and causes a syntax problem of the SQL.				
Suggested Correction	Add quotes to the query string.				

#32	The button "Manage my account" does not direct to user Profile interface	Type	**White/ Black box**	Severity	**2**
Output before seeding	The Manage should be direct to the user Profile interface				
Output after seeding	The button is not connected to the user profile				
Suggested Correction	Add "intend" in order to direct from home_activity to user Profile interface				

#33	In Home_activity_list_class, the setHobbies function is useless	Type	**White box**	Severity	**3**
Output before seeding	The class is for storing the data that will be used in ListView in HomeActivity. It should not have the function setHobbies to change the hobbies.				
Output after seeding	The function is implemented in the class.				
Suggested	Remove the function.				

Correction	