“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 19 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 “`&amp;`”, “>” to “`&gt;`”, “<” to “`&lt;`”, 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 or security answer 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()`. • Add the same lock for checking security answer.				

#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	The way to enable such server-wide check for flood requests is included				

Correction	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.				

#19	The `age` field and `birthday` field in user profile may not match.	Type	**White & Blackbox**	Severity	**3**
Output before seeding	Server calculates `age` from the `birthday` given by user, or if `birthday` is not set, use `age` directly. Ban registration if the `age` is less than 13.				
Output after seeding	`age` and `birthday` become two separate and *unrelated* fields. Server will accept invalid data like `birthday` "2014-01-02" but `age` "18". Also there is no minimum age anymore.				
Suggested Correction	In user registration and and edit profile parts of action.php, add the sanity check code back.				

Client-side Defects

#1	The passwords sent by client are not encrypted.	Type	**Black box**	Severity	**3**
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 Retrieve Lost Password Activity does not check the validity of the email address and set error to the EditText	Type	**Black box**	Severity	**2**
Output before seeding	If an email has an incorrect form, it will setError to the EditText				
Output after seeding	If an email has an incorrect form, it will still send the request to the server.				
Suggested Correction	Check the validity of the email before sending the information to the server.				

#3	The server cannot be connected, the client cannot show the user about the server connection problem	Type	**Black box**	Severity	**3**
Output before seeding	When the server cannot be connected, the null value of the response string is correctly handled and a server cannot be connected message showed on the screen				
Output after seeding	When the server shutdown, FileNotFound exception raised and the null value ignored by the client, a password incorrect error may show up.				
Suggested Correction	Add the server connection check and shows up the server cannot be connected information				

#4	The network availability is not checked by the client during the registration, if the network is not available, it will cause the app crashed.	Type	**Black box**	Severity	**2**

Output before seeding	If the network is not available, the client will prompt the message and any network operation 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 “isNetworkAvailable” method to check the network before trying to access the network.

#5	Matching desired range should be a number, however the client does not check if it is not	Type	**Black box**	Severity	**2**
Output before seeding	A character rather than a number cannot be typed into the desired range input dialog				
Output after seeding	Any character can be typed into the desired range input dialog				
Suggested Correction	Add the inputType to the EditText				

#6	Security questions set of editProfile activity is retrieved directly from the server, if a network is not available, it will crash the program	Type	**Black box**	Severity	**2**
Output before seeding	If the security question set cannot be retrieved, it will denied the profile editing.				
Output after seeding	If the security question set cannot be retrieved, a crash will happen in the program				
Suggested Correction	Add the network availability check when retrieving the questions				

#7	SQLHandler created a friend table in the database, but it is not updated	Type	**White box**	Severity	**2**
Output before seeding	A friend table should be updated where a match found for the user and user chooses some matched users as friends				
Output after seeding	User cannot choose a user as friend and update the friend table in the local database				

Suggested Correction	Call appropriate methods provided to add friends after a match

#8	The addFriend method in the SQLHandler implemented but not used	Type	**White box**	Severity	**2**
Output before seeding	A friend can be added to the user's friend database table after a match.				
Output after seeding	User cannot add a friend to his/her friend table				
Suggested Correction	Call the addFriend function in the MatchFragment				

#9	The orientation of any activity does not change to the horizontal when the orientation of the phone change to the horizontal	Type	**Black box**	Severity	**3**
Output before seeding	Any activity should changed to the horizontal when the phone changed to the horizontal.				
Output after seeding	Activities does not change the direction when the phone changed to the horizontal.				
Suggested Correction	Add the code to handle the orientation changes.				

#10	Does not check the network availability when starting the editProfile and Profile activity	Type	**Black box/ White**	Severity	**2**
Output before seeding	If network is not available, the editProfile or Profile activity should show an error and return back to previous				
Output after seeding	The program will not prompt an error about network availailbity, the default value will show on the screen				
Suggested Correction	Check the network availability before accessing the network				

#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	**2**
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 or set the default icon				
Output after seeding	If no image on the phone, nothing will show up and signup activity cannot be done				
Suggested Correction	Add a method to show up a message box in the signup fragment when there is no image on the phone and set the icon to default.				

#12	The editProfile does not display the correct user information of hobby	Type	**Black box**	Severity	**3**
Output before seeding	The editProfile activity should be able to display the hobby.				
Output after seeding	The editProfile activity just display the empty output of hobby.				
Suggested Correction	Should read the bundle returns from the APIHandler and set correct fields corresponds to the TextViews				

#13	The signup fragment does not check the validity of the email address correctly	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	Only check if it contains @, If the email is in the incorrect form like a@b+b+b+b+c, no error message will show up and the client will send the incorrect user email information to the server.				
Suggested Correction	Correct the email validity check before sending the information to the server.				

#14	The signup fragment does not check the validity of age	Type	**Black box**	Severity	**2**
Output before seeding	The signup fragment will show up an error message when the user age is in the incorrect form				
Output	The signup fragment will not show up any error and incorrect user age				

after seeding	will crash the program or will be sent directly 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 editProfile Activity changed security anwser to empty if the field is empty	Type	**Black box**	Severity	**2**
Output before seeding	If security answer is empty, an error message should show up and change cannot be made.				
Output after seeding	It will update the security question to empty without any warning				
Suggested Correction	Check the validity of the security answer before submitting the information to the server				

#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 gender setted wrongly				
Suggested Correction	Check the validity of gender				

#18	The signup fragment does not check the password length should longer	Type	**Black box**	Severity	**2**

	or equal to 6				
Output before seeding	If the password length is not longer or equal to 6, an error message will show up				
Output after seeding	If the password is not not longer or equal to 6, no error message will show up				
Suggested Correction	Check the length of the password				

#19	In the SQLHandler getAllFriendsByUserId method, the cursor is not moved to the first, so data cannot 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	activity_help.xml is not be used anywhere and has no uses	Type	**White box**	Severity	**3**
Output before seeding	This xml definition should be removed				
Output after seeding	This xml is not being removed				
Suggested Correction	Remove the xml file				

#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	A null value is not checked and will cause a NullPointerException when				

after seeding	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	When set the security answer in the register activity, if no answer in the field, it will set the answer to empty	Type	**Black box**	Severity	**2**
Output before seeding	If the answer of a security question is empty, it will shows an error and register cannot be done.				
Output after seeding	If the answer of a security question is empty, it will continue the registration process and an empty answer will be setted.				
Suggested Correction	Check the answer field before submiting the register information				

#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	MatchingFragment registered a locationListener, but it does not remove after the fragment removed	Type	**Black box/White Box**	Severity	**2**
Output before seeding	The locationListener should be removed before the fragment removed.				
Output after seeding	The locationListener is not removed and a GPS information change can cause a NullPointerException				
Suggested Correction	Remove the listener before removing the fragment				

#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	**Black 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	**1**
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	Verify the security answer requires a md5 encoded answer sent to the server, however, the original answer sent to the server without encoded	Type	**White box**	Severity	**2**
Output before seeding	The encoded md5 anwser should be sent to the server and server can verifies it correctly				
Output after seeding	The original answer sent to the server, server cannot verify it correctly and cause the verify failed.				
Suggested Correction	Encoded the answer by md5 before sending to the server				

#33	In the login activity, String in the password field can longer than the maximum length	Type	**Black box**	Severity	**3**
Output before seeding	The password field should restrict the password length to maximum length				
Output after seeding	The password field can enter a string longer than the maximum length of the password.				
Suggested Correction	Add maximum length restriction to the password field.				