Nabeel Khan
Nkhan26@learn.senecac.on.ca

OpenVULab Setup / Steps taken

Listed in this documentation were the steps taken to get OpenVULab and rascal to run properly either locally or on the server. All files were included in the source code given, and one can get OpenVULab to run without any additional downloads.

Using PHP Version 6.0 on OpenVULab

The web is now moving to PHP 6.0, and the current version (PHP 5.3) is taking huge steps towards this move. The code downloaded is works for seems designed for PHP 5.0, and uses a lot of methods that are deprecated, and is not recognized for PHP 5.3.

One of the major changes is that PHP will not recognized the short tag "<?" anymore. While this settings can be changed in the "php.ini" file, it is recommended to change all short tags from "<?" to "<?php", as PHP 6.0 will not recognize the PHP short tags anymore. More information on this can be found on: http://kevin.vanzonneveld.net/techblog/article/prepare_for_php_53/. All files require these changes, especially the "lib" files in the admin folder. Short tags also include changing all "<?=" to "<?php echo(parameters);.

In addition, the PHP functions "Split", "ereg_replace" and "ereg" are now deprecated, and "preg_split", "preg_replace" and "preg_match" must now be used instead. The files requiring these changes will be noted with error messages when running the admin/signup.php. Another deprecated function is the "set_magic_quotes_runtime" method, which there is no current replacement for, and is removed altogether in the admin/include/lib/esp18n.inc file.

The output_buffer in the php.ini file needed to change; else white space at the end of PHP files will create a header error. The default value is "Off" and just needs to be switched to "On", with the buffer size at default 4096. The error trapping in PHP is greater than previous versions, as it will pick up errors not seen before, or make the file not perform as it was to. For example, in the admin/manage.php file, there is not closing "?>" PHP tag, which will make the manage.php page not work properly. This was the same for a few other files in the include/lib folders.

Some of the methods used in the admin/include/head.inc are deprecated, and these needed to be corrected before the manage.php page can work correctly. For some reason, the error checking, which is on by default stopped working, and this needed to be address (see below). The ereg and ereg_replace functions needed to be replaced with the preg_match and preg_replace functions respectively. In addition to the include/lib files, the files in the admin/include/where folders needed to be changed. The files in this folder deals with the survey and survey tool (after a user log in on the manage.php page).

Error Reporting

Error reporting is turned on by default in the php.ini file, but it will not work in most of the PHP pages for OpenVULab. However, this can be changed by include this code in the page you are currently working on:

```
error_reporting(E_ALL);
ini_set('display_errors', '1');
```

This becomes very useful to correct errors. The “head.inc” file was given errors, but it went unnoticed until this code was inserted. This then displayed the notices and warnings, which was needed to fix the errors.

The errors obtained were all of version compatibility, as fixing the errors included more of the same: the “<?php” start tag, changing “<?=” to “<php echo(”, and replacing deprecated methods. In addition, errors will include invalid array indexes, as they needed to be changed. For example, the “$_SESSION[acl]” in the “manage.inc” file need to be changed to “$_SESSION['acl']”.

Understanding Rascal

Rascal is the recording component that creates a video capturing the tester’s screen. It takes screen captures along with the mouse pointer of the user’s desktop, and stores it temporarily in a folder (currently rascal.vulab.yorku.ca/files). There is also a voice recording component to it, as it records from the microphone of tester’s computer. These are the steps that the rascal component follows:

- Before the user is about to start the capturing, there is a digital signature warning that the user has to accepted for the applet to work.
- The applet then initializes and waits for the user to click the “Start Test” link
- After link is clicked, the screen capture begins, storing images temporarily in a directly created. It does not stream these images, as images may be stored temporarily (or in memory) of the user’s pc, and transferred slowly to the directory. As a result, even after the capture is done, it will still be transferring files. If a user closes the page while the “please wait” message is still being displayed, then it will not complete the transfer. The audio is uploaded quickly after the “Stop test” link is clicked, but images are still being uploaded. This step, as currently set up, may take quite some time, as it may takes twice the amount of time it took to record (eg. recording 2 minutes of screen capture may take 5 minutes to upload all files).
- After all the images are uploaded, a “Done.all” file is created, which informs the Mencoder software that the video can be created now. Mencoder uses the screen capture by name, and compiles all in to a video, with the addition of sound. This creation event is triggered by a crontab task created to execute the “rascal_task.sh” script, which will run the mencoder command “mf” in the “files” directory, and delete the folder which the video was created from. The video id is then uploaded to the server and can then be access via link rascal.vulab.yorku.ca/files/vidid.avi.

Currently, the rascal component is still hosted on the vulab.yorku.ca server, which will be changed to the local server later. To make the rascal component run on the York server, the “rascal.jar” file must be placed in the “/public/” folder in the http root folder.

Nabeel Khan
Nkhan26@learn.senecac.on.ca

Set Up OpenVULab Locally

Install XAMPP on Local Machine

XAMPP is an easy to install Apache distribution that creates MySQL, PHP and Perl server. It can be installed with instructions from: http://www.apachefriends.org/en/xampp.html. For windows, the files must be placed in the %ROOTDIR%\xampp\xampp\htdocs folder to be recognized locally, ie. http://localhost/hostfiles.php.

Set up database

In the XAMPP admin, you can use "phpMyAdmin" to set up the database and tables required for OpenVULab. While you can give your own database and table names, it is easier to use the database and table names specified by the source code in the admin\config.live.php file. This has a database name of "eblake_vudev", along with a username "eblake_vudbuse" and password "vudbpass". All of these attributes can be created in phpMyAdmin by creating the database first, choosing the database, and then creating privileges. You can then import the "database.sql" file from the root folder, which will create all tables required for OpenVULab. The database can be tested right away from the admin\signup.php file, even though the page formatting is currently incorrect.

Fixing/Updating/Setting up OpenVULab on Server

The server is currently running Fedora 7, and the files itself needed to be updated. This was done using the "yum update" command, for the server files itself. This is followed by "yum update php", which updated the PHP and MySQL versions for the system. However, since this is Fedora 7, it only goes up to PHP 5.2.6. There are ways through RPM to get it to PHP 5.3, but it will require changing some PHP configuration files. Since the code used in this project is PHP 6.0 compatible, it is also "good" PHP code, which means it is backward compatible with any PHP 5.x versions.

The MySQL database then needed to be setup. This can be done in the following steps:

- Log in to MySQL: mysql –u root –p
- Enter MySQL password
- Create database with same properties in config.live.php file: create database eblake_vudev
- Creating same user: CREATE USER 'eblake_vudbuse'@'localhost' IDENTIFIED BY 'vudbpass';
- Grant privileges: GRANT ALL PRIVILEGES ON *.* TO 'blake_vudev'@'localhost' WITH GRANT OPTION;
- Import the database.sql file (from shell): mysql -u root -p -h localhost eblake_vudev < database.sql

After these steps were completed, the database tables were setup, and now ready to be used. The PHP files were then uploaded to the /var/www/html/ folder, and permissions for some files and folder needed to be changed to "755". After this, the files are now on the server and now ready to be used. You may need to restart the server: service httpd restart.

Nabeel Khan
Nkhan26@learn.senecac.on.ca

Install JVM on Server

The Rascal component requires JVM 1.5 or higher, which meant that Sun Java had to be installed on the server. However, it was not as simple as installing the Sun JDK alone, as it needed to become the default java compiler on the system. Two files were needed, and the steps were followed from: http://www.ils.unc.edu/~rcapra/rhel5-tomcat-solr.php. One of the major issue is that the version of the JDK installer (eg. jdk-6u2-linux-i586-rpm.bin), had to be the same as the JPackage Sun Compat file (eg. java-1.6.0-sun-compat-1.6.0.02-1jpp.i586.rpm), and this required searching for these files separately, as the sun-compat file versions only went up to ".06-1jpp", while the java JDK file is at 6u18. Therefore the files had to be searched for directly, and downloaded from separate sources. The files installed on this server were the "6u3" JDK version, along with the ".03-1jpp" sun-compat file.

Set Up MPlayer Mencoder

MPlayer (http://www.mplayerhq.hu/) is a multi-platform multimedia player with codec, which can play, make or edit videos. This player is used to create the videos captured form the audio and video of the rascal component. A very useful guide for setting this up is here: http://www.mjmwired.net/resources/mplayer-fedora.html. A summary of the steps used are:

- Download and install lame-3.97.tar.gz
- Download and install xvidcore-1.1.3.tar.bz2
- Download and install MPlayer codec pack: essential-20071007.tar.bz2
- Download and install MPlayer-1.0rc2.tar.bz2 (no GUI used)
- Configure and make MPlayer, then complete installation with: su -c 'make install'

One can check if it is set up properly by using the 'man mplayer' or 'man mencoder' command, as the new command should exist.

Setup Rascal

There are a few files that are required to get the rascal component going. As mentioned earlier, Mencoder and Sun Java were both required for this component to work. In addition, few files were required and needed editing for this component to work.

Create File Folder

A file folder was created for storing the images captured by rascal. This folder needed full permission granted (777), as this is the only way new folders will be able to be created using the rascal component. For this project, a folder named "files" was created in the root of the "html" directory. Therefore, it can be accessed via path: http://serveraddress/files.

Edit "uploader.php"

The path for upload directory in the "uploader.php" file needed to be changed to the path of the directory where the files will be stored. The "uploader.php" file was stored in the html root folder (serveraddress/uploader.php), and therefore the path for the folder created above was "./files". In addition, some PHP tags needed to be converted to PHP 6.0 tags (<?php).

Nabeel Khan
Nkhan26@learn.senecac.on.ca

Edit "rascal_task.sh"

The "UPLOAD_DIR" path in the rascal_task.sh script file needed to be changed to the path of the folder used to store the files (same folder as above). Both script files were placed outside the html directory and in the "/etc" folder. Therefore the path in the task file for the upload directory was an absolute path to the files directory (var/www/html/files).

Once MPlayer is set up successfully, "mencoder" will become a command on the server, making the "MENCODER_LOCATION" variable unnecessary. Therefore where the tow "MENCODER_LOCATION" variables are mentioned, it can be changed to just "mencoder". This script files also needed execute permissions for all, and then it can be executed by just calling the file name.

The script checks the folder containing the capture, checking for a "Done.all" file, which is created when all files are finished uploading. Once a "done.all" file is found, it then runs the mencoder command creating a video based on the files in the folder. It then changes the "done.all" file to "done.processed", as to not create another video.

Create Crontab Task

Since a user will be unable to access and run the script file, it needed to be an automated task that runs very often. For this server, it ran the script every five minutes. A crontab task needed to be created to do this. To edit crontab, the "crontab –e" command was used, which opened the crontab task file in a VI editor. More information on crontab can be found here: http://www.htmlbasix.com/crontab.shtml.

For this server, the "rascal_task.sh" file needed to be run every five minutes, therefore the crontab entry was "*/5 * * * * /etc/rascal_task.sh > dev/null". After the crontab is saved, the "crontab –l" command was used to view all entries in the crontab, to check if the entry was created successfully.

Edit "rascal.jar"

The last step needed to get the rascal component going was editing the "rascal.jar" file. This file location is in the public folder in the html root (/public). A new project was created in eclipse, and the source code was imported from the existing "rascal.jar" file. In the "org.fluidproject.vulab.rascal.utils.http.HTTPStreamerConstants.java" class, the rascal path needed to be changed to the path of the uploader.php file (in this case, http://serveraddress/uplaoder.php). This can then be tested by run the project as a JApplet, which should create screen shots and save them in the upload folder. Once working successfully, the project was exported as a jar file.

For the jar file to work on a server, it required a signature, or else permission problem will occur. More information on this can be found here:
http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/keytool.html and
http://java.sun.com/j2se/1.3/docs/tooldocs/win32/jarsigner.html.

However, only two basic commands was really needed to sign this jar:

```
	keytool -genkey -alias duke -keypass dukekeypasswd –keystore "path-to-
	store-key/keystorename.keystore"
```

Using the information above, the jarsigner was then called:

Nabeel Khan
Nkhan26@learn.senecac.on.ca

```
jarsigner -keystore “path-to-store-key/keystorename.keystore” -
storepass myspass -keypass dukekeypasswd rascal.jar duke
```

Once signed, execute permissions needed to be added to the “rascal.jar” file, and the rascal component should now be fully working with the project survey.