**Group 7 Analysis:**

**The App Musicplayer:**

This app finds .mp3 and .wav files and puts them in a list. This list is controlled by the MainActivity. The user can then choose a section from this list as a playlist. The playlist is controlled by the PlayListActivity and PlayList classes. The user can then choose to clear or play the list. If they choose to play the list then the media player is started. This is controlled by the PlayerActivity and the Player classes which implement all the functionality of the media player such as the play/pause, next and previous buttons.

**Obfuscation Techniques:**

Group 7’s primary obfuscation technique was name obfuscation. They appear to have used an algorithm such as the sha1 hashing algorithm to produce garbled names for methods and variables. The purpose of this was to try to obscure their meanings and make it harder for humans to differentiate between variables.

Group 7’s secondary obfuscation technique was some minor flow control obfuscation. This ‘confused’ the reverse engineering tool we used and produced source code which did not compile. See Figure 1:

…

switch (i)

{

default:

case 2131492975:

case 2131492977:

case 2131492976:

do

{

do

{

return;

this.mze7416cgy77np69y47cp778m8232qy8483swkjf9072g92lux21koe2296lz74887161.setText(this.olaht97301llc9484596423b27h61n20vo2343to2179q17h84p091ir51lgir024s7069.mkt70q60zvmf6489w627028964896so91g14543313dju31930na43w2498py150yrz12329());

return;

} while (!this.olaht97301llc9484596423b27h61n20vo2343to2179q17h84p091ir51lgir024s7069.eqd5820wr18sfa694n94cw9517de22lj4627ug513mq647818192d2163xnk79817670s30());

…

Figure 1. An example of un-compilable code from the onClick method of PlayerActivity class

**De-Obfuscation Techniques:**

Through the use of tools such as BytecodeViewer.2.9.8 and jd-gui-1.4.0 reverse engineering this app to obtain the obfuscated source code was relatively easy. Then through the use of an IDE like Eclipse renaming and editing the flow controls was quite easy.

Figure 2 shows how the un-compilable code from Figure 1 has been changed so that it is now compilable and the names are human readable. The compilable code was obtained by using a few different reverse engineering tools and comparing the results of each. The names were assigned by reading through the code and picking names that made general sense.

…

switch (i) {

default:

case R.id.btPlay:

this.playPauseButton.setText(this.player.pauseOrUnpause());

return;

case R.id.btFf:

if (this.player.fastward5Seconds()) {

setAlbumArt(this.player.setSongAndGetItsURI(this.player.getSong()));

this.songTitle.setText(this.player.getSong().getName().replace(".mp3", "").replace(".wav", ""));

this.theSeekBar.setMax(this.player.getDuration());

this.playPauseButton.setText("||");

}

return;

case R.id.btRw:

if (this.player.rewind5Seconds()) {

setAlbumArt(this.player.setSongAndGetItsURI(this.player.getSong()));

this.songTitle.setText(this.player.getSong().getName().replace(".mp3", "").replace(".wav", ""));

this.theSeekBar.setMax(this.player.getDuration());

this.playPauseButton.setText("||");

}

return;

…

Figure 2. An example of de-obfuscated code from the onClick method of PlayerActivity class

**Obfuscation Evaluation:**

These obfuscation techniques were not particularly strong. The name obfuscation could be overcome quite easily once the app’s programme logic had been recovered. This is done by reading through the logic, understanding what the variable, method or class does and then giving it an appropriate name.

The obfuscation technique used to change the control flow was not effective against most of the reverse engineering tools we tried, such as some of the decompilers built into BytecodeViewer.2.9.8. This suggests that the technique is either ineffective or was done unintentionally and it just so happens that the decompiler JD-GUI, and some others, were ‘confused’ by this switch statement.

The obfuscation technique used to change the control flow seemed to be the most unique part of this obfuscation since we did not observe this behaviour in any other groups’ apps. However we were able to decompile the code more effectively with other tools and recover the app’s programme logic. We could then read through it and de-obfuscate the variable names relatively easily. Therefore these were not very effective obfuscation techniques.