**Group 5 Analysis:**

**The App tryreverseengineerthis:**

This app is a basic scheduling app. It allows the user to check into a ‘lab’. Only the 702 lab is supported. The way it does this is by checking the current time with the predefined times of the 702 classes.

**Obfuscation Techniques:**

Group 5’s obfuscation techniques were to use base64 encoding on their strings and to add branch statements which will never execute one of the branches. Along with this they also used some basic name obfuscation.

The base64 strings could be de-obfuscated quite easily as can be seen in Figure 1.

tag1: 20k20dk20ASD2d==..........Content-Length

tag1: amereparkingWarden..........goldfish

tag1: ajd202ASsd20L025..........17

tag1: ask20asdj20jd9..........charset

tag1: Judy.SheWasGoingTo..........google\_sdk

tag1: SheDidNotRealise..........sdk

tag1: asd202d0asD2==..........174.76

tag1: wasJustGoingtobe..........vbox

tag1: apsojdojaspdjaspo..........No classes today for

Figure 1. An example of base64 decoding from the group 5 app.

The misleading branch statements were not particularly difficult to de-obfuscate since the Boolean values were constant and it wasn’t hard to work out what those constants are. Figure 2 shows such a branching statement, where the method ‘fakebranch’ will always return false for the value 1086.

**private** **static** **boolean** fakeBranch(**int** i) {

**if** ((((**double**) i) / 2.0d) % 2.0d != 1.0d) {

**return** **true**;

}

**return** **false**;

}

**public** **void** onAttach(Context context) {

**super**.onAttach(context);

**if** (C0926A.*fakeBranch*(1086)) {

C0592E.*L1L1L1L1L1L1L*().L1L1L1L1L1L1L1L();

} **else**

**if** (context **instanceof** C0590B) {

**this**.L1L1L1L1L1L1L1L1L1L1L1L1L1L1L1L1L1L1L1L1L1L1L1L1L1L1L1L1L1 = (C0590B) context;

} **else** {

**throw** **new** RuntimeException(context.toString());

}

}

Figure 2. An example of a misleading branch in the ListActivity class.

**De-Obfuscation Techniques:**

Through the use of tools such as BytecodeViewer.2.9.8 and <http://www.javadecompilers.com/apk>, which is an online decompiler, reverse engineering this app was relatively easy. Next we used a tool, Android application x, to decode the base64 encoded strings. Then through the use of an IDE like Eclipse we worked through the program logic removing the misleading branch statements.

**Obfuscation Evaluation:**

This obfuscation technique was moderately difficult to de-obfuscate due to the number of distractions in the code base like the misleading branch statements.

The base64 encoding was quite easy to de-obfuscate since it is a well-known encoding method used in web applications for quite some time now.

Although we did not de-obfuscate all the variable and class names given more time to spend reading and understanding the code we would have been able to (almost exactly) recover the original source code.