**G8 app analysis:**

**G8 uses very limited source-code obfuscation techniques in their application (no JVM/DEX bytecode obfuscation technique used):**

1 adding thousands of junk classes/methods. this is really boring and can be easily fixed by just removing the useless stuff with many easy tools. This method brings very bad overhead for the application which will be unacceptable by the industry since their apps are normally much complexer than the example.

2 class/method rename, which is basically useless since it just adds a few reading difficulties. Also it is the most common obfuacation technique used in industry.

3 adding an infinite loop in an unreachable branch marked with string “opening” to confuse the reader, but it can be easily discovered by checking the running status of the app.

4 they mirrored all the classes, there are two sets of exact same classes and both of them are invoked, so the whole footprint are doubled finally with no functionality added.

The cracked source code is attached, it is quite clear developers with some basic android knowledge is able to find out the logic and the the right place to add malicious code if needed, will take one functioni as example for analysis.



**The methods used to crack it:**

**1) removing the useless stuff**

There are many many way to remove the thousands of useless classes, methods, for example: python, ruby, even C

I just use some simple regular expressions to match them which can be easily performed in notepad++:

a)

^[ ]{1,}new [0-9a-zA-Z]{1,}\(\)\.[0-9a-zA-Z]{1,}\(\)\;$

To remove:

new O1O1t0ti().a63516fed();

new O1O1t0ti().a794a2c9a();

b)

remove blank lines:

^[ \t]\*$\r?\n

c)

^[ ]{1,}public (int|boolean|void|String) [0-9a-zA-Z]{1,}\(\)[ ]{0,}\{\r?\n[ ]{1,}return[ ]{1,}[0-9a-zA-Z]{1,}\;\r?\n[ ]{1,}\}\r?\n

To remove trash in the following pattern:

public boolean OO0il10O() {

return oiLiOo0l;

}

public int OlO0iiLo() {

return L1L0ILi0;

}

d)

^[ ]{0,}public void [0-9a-zA-Z]{1,}[ ]?\(final[ ]{1,}(String|boolean|int)[ ]{1,}[0-9a-zA-Z]{1,}[ ]{0,}\)[ ]{0,}\{\r?\n[ ]{1,}[0-9a-zA-Z]{1,}[ ]{0,}=[ ]{0,}[0-9a-zA-Z]{1,}[ ]{0,}\;\r?\n[ ]{0,}\}\r?\n

To remove trash in the following pattern:

public void I00oliLi(final String ol111ll1) {

Ol111ll1 = ol111ll1;

}

public void L0Ool0L0(final String l011i010) {

L011i010 = l011i010;

}

public void L1OlILLL(final boolean o1OOLliI) {

O1OOLliI = o1OOLliI;

}

e)

^[ ]{0,}public boolean [0-9a-zA-Z]{1,}\(\)[ ]{0,}\{\r?\n[ ]{0,}return obj\.x \+ obj\.y \=\= 3\;\r?\n[ ]{0,}\}\r?\n

To remove trash in the following pattern:

public boolean a01b1a84a() {

return obj.x + obj.y == 3;

}

public boolean a191d875a() {

return obj.x + obj.y == 3;

}

public boolean a2ab64b63() {

return obj.x + obj.y == 3;

}

**2) The class/method rename**

No way to fix since the original messages has been removed, but can change the long name into shorter ones to fasilitate analysis, this can be done by source analysing tools or even manually replace.

**3) they mirrored all the classes, means there are two exact same classes and both of them are invoked, so the whole footprint are doubled finally with no new functionality.**

No need to fix it, just be aware of this is enough.

aa.java == bb.java

cc.java == dd.java

...

...

and all the synthetic (the class whose name contains $) class can be classified into 2 versions

cc$1 == dd$1

the rest are equal

**take one function as example for the functionality analysis:**

public class aa extends AsyncTask<String, Void, String>

{

…

protected /\* bridge \*/ Object doInBackground(final Object[] array) {

return this.doInBackground((String[])array);

}

protected String doInBackground(final String... array) { //the actual function to do the job

Label\_0334: {

if (loOoioOO == "opening") { // this will never be true, it is just a trap

while (true) {

while (true) { //double endless loop

...

...

}

}

}

else {

if (loOoioOO == "booking") { // for the booking page

try {

final JSONArray jsonArray4 = new JSONArray(array[0]);

for (int k = 0; k < jsonArray4.length(); ++k) {

final JSONObject jsonObject = jsonArray4.getJSONObject(k);

final pp ll001IoL2 = new pp();

final xx llIIooIL2 = new xx();

ll001IoL2.Il0li01o(llIIooIL2);

ll001IoL2.IO0oO11I(jsonObject.getString("lab\_id")); //get the lab id

llIIooIL2.setStart\_time(jsonObject.getString("start\_time")); //get start time

llIIooIL2.setEnd\_time(jsonObject.getString("end\_time")); //get end time

llIIooIL2.setDescription(jsonObject.getString("description")); //get description

llIIooIL2.setCreated\_on(jsonObject.getString("created\_on")); //get create on

IiiIO001.add(ll001IoL2); //add these info into array list

}

break Label\_0334;

}

catch (JSONException ex2) {

System.out.println(ex2.toString());

break Label\_0334;

}

}

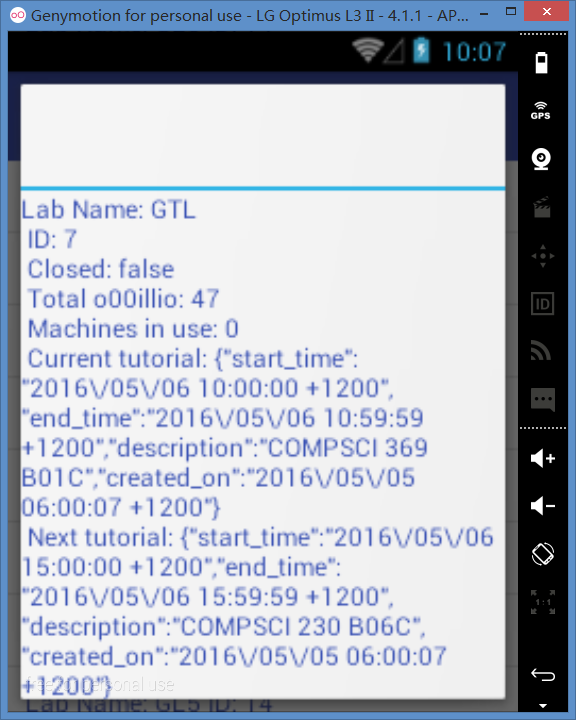
}

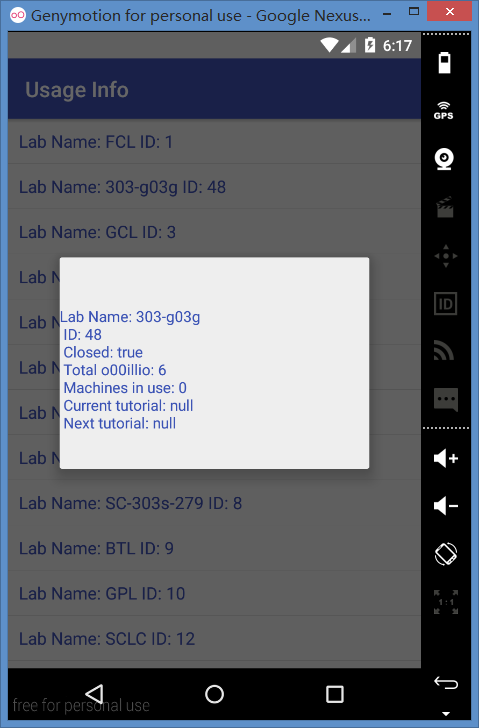
return "Done";

}

**Bug in the APP:**

The obfuscated string showed in the application, it seems like that the obfuscation tools mistakenly replaced some strings that will be used in the activities while doing the renaming.

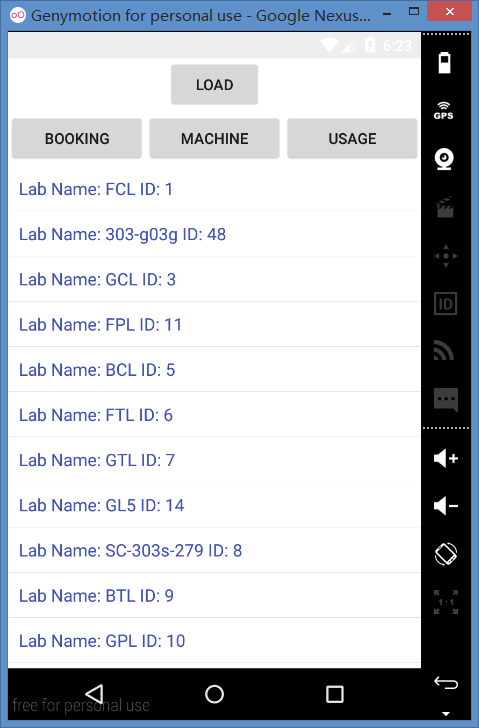




The obfuscated string cannot be properly handled by the application itself, it showed during running.

**Appendix:**

the main activity of the APP:



According to the main activity, it is easy to find out that the ‘opening’ tag is a fake entry, it is just something to confuse the reader, since there is an infinite loop in this branch.