附錄 2

1. JOOX (com.tencent.ibg.joox.apk)

This app has some malicious activities, such as obtain SIM operator name, obtain SIM serial number, obtain SIM status, read phone number, connect to the Internet, obtain network information and obtain device information.

a) obtain SIM operator name

```
private static String getOperationName() {
    String v0_1;
    String v1 = "";
    try {
        v0_1 = ApnManager.mAppContext.getSystemService("phone").getSimOperatorName();
    }
    catch(Throwable v0) {
        v0.printStackTrace();
        v0_1 = v1;
    }

    return v0_1;
}
```

b) obtain SIM serial number

```
public static String getSimSerialNumber(Context arg1) {
    String v0;
    if(arg1 == null) {
        v0 = "";
    }
    else {
        Object v0_1 = arg1.getSystemService("phone");
        v0 = v0_1 == null ? "" : ((TelephonyManager)v0_1).getSimSerialNumber();
    }

    return v0;
}
```

c) obtain SIM status

```
label_1404:
    if(a.q != null) {
        a.a(a.ck, Integer.valueOf(a.q.getSimState()));
    }
    else {
        a.a(a.ck, "ERROR");
    }

    if(!a.n) {
        a.a(a.aJ, "NO_PERMISSION");
    }
    else if(a.r == null) {
        a.a(a.aJ, "ERROR");
    }
```

d) read phone number

```
public class md {
private static String a(String arg3) {
    StackTraceElement[] v0 = Thread.currentThread().getStackTrace();
    if(v0.length >= 4) {
        arg3 = new StringBuilder(String.valueOf(arg3).length() + 13).append(arg3).append(" @").append(v0[3].getLineNumber()).toString();
    }

    return arg3;
}
```

e) connect to the Internet

```
public final void a(Context arg4, String arg5, boolean arg6, HttpURLConnection arg7) {
    arg7.setConnectTimeout(60000);
    arg7.setInstanceFollowRedirects(false);
    arg7.setReadTimeout(60000);
    arg7.setRequestProperty("User-Agent", this.b(arg4, arg5));
    arg7.setUseCaches(false);
}
```

f) obtain network information

```
v0_2 = v2;
try {
label_6:
    if(v0_2 != null) {
        v0_3 = v0_2.getMacAddress();
    }
    else {
        goto label_19;
    }

    goto label_8;
}
catch(Exception v0) {
label_22:
    v0_3 = v1;
    goto label_14;
}
```

g) obtain device information

```
else {
label_41:
    Config.loadConfigs();
    if(!new BigInteger(PhoneUtil.getDeviceId(Magnifier.sApp), 16).mod(new BigInteger(String.valueOf(((int)(1f / Config
        v0 = Magnifier.ILOGUTIL;
        v3_1 = new String[v8];
        v3_1[0] = Magnifier.TAG;
        v3_1[1] = "Not chosen to get info.";
        v0.i(v3_1);
    }
```

2. Clear Phone (com.clear.phone.clearphone.apk)

This app has some malicious activities, such as obtain SIM operator name, obtain SIM status, read phone number, connect to the Internet, obtain network information and obtain device information.

a) obtain SIM operator name

```
public static String h(Context arg4) {
    String v1 = null;
    try {
        Object v0_1 = arg4.getSystemService("phone");
        if(((TelephonyManager)v0_1).getSimState() == 5 && !((TelephonyManager)v0_1).getSimOperatorName().isEmpty()) {
            String v0_2 = ((TelephonyManager)v0_1).getSimOperatorName();
            return v0_2;
        }

        if(((TelephonyManager)v0_1).getNetworkOperatorName().isEmpty()) {
            return v1;
        }

        return ((TelephonyManager)v0_1).getNetworkOperatorName();
    }
    catch(Throwable v0) {
        return v1;
    }

    return v1;
}
```

b) obtain SIM status

```
try {
label_47:
    if(((TelephonyManager)v1).getSimState() != 5) {
        goto label_56;
    }

    goto label_50;
}
catch(Exception ) {
    goto label_56;
}
```

c) read phone number

```
label_24:
    if(v1 != null && v2 != null) {
        String v3_1 = v1.getClassName();
        String v4 = v1.getMethodName();
        int v1_2 = v1.getLineNumber();
        JSONObject v5 = w.a();
        w.a(v5, "timestamp", Long.toString(System.currentTimeMillis()));
        w.a(v5, "message", v2);
        w.a(v5, "sourceFile", v3_1);
        w.b(v5, "lineNumber", v1_2);
        w.a(v5, "methodName", v4);
        w.a(v5, "stackTrace", v0);
        this.d(v5);
        y.b.b("saving to disk...");
        this.c(v5);
        this.h();
    }
```

d) connect to the Internet

```
public Void call() throws Exception {
    URL v0 = new URL(this.val$url.toString());
    URL v1 = null;
    URLConnection v2 = ((URLConnection)v1);
    while(v0 != null) {
        v2 = v0.openConnection();
        boolean v0_1 = v2 instanceof HttpURLConnection;
        if(v0_1) {
            v2.setInstanceFollowRedirects(true);
        }

        v2.setRequestProperty("Prefer-Html-Meta-Tags", "al");
        v2.connect();
```

e) obtain network information

```
try {
    if(v0.getUpmi() == 1) {
        v1_1.put("imei", CommonDeviceUtil.getIMEI(this.a));
        v1_1.put("mac", CommonDeviceUtil.getMacAddress(this.a));
    }
```

f) obtain device information

```
public boolean onKeyShortcut(int arg5, KeyEvent arg6) {
    Menu v0 = this.getMenu();
    if(v0 != null) {
        int v2 = arg6 != null ? arg6.getDeviceId() : -1;
        boolean v3 = true;
        if(KeyCharacterMap.load(v2).getKeyboardType() != 1) {
        }
        else {
            v3 = false;
        }

        v0.setQwertyMode(v3);
        return v0.performShortcut(arg5, arg6, 0);
    }
```

3. The 1st KPMA (com.kt.kpma.apk)

This app has some malicious activities, such as read phone number, connect to the Internet obtain network information.

a) read phone number

```
while(v5 < v4) {
    if(v3.getClassName().startsWith(arg10[v5])) {
        this.stackLineHash = this.stackLineHash * 31 + (v3.getClassName().hashCode() + v3.getMethodName().hashCode());
        this.lineNumberHash = this.lineNumberHash * 31 + v3.getLineNumber();
    }
    else {
        ++v5;
        continue;
    }

    break;
}
```

b) connect to the Internet

```
if(this.CONF.getHttpConnectionTimeout() > 0) {
    ((HttpURLConnection)v6).setConnectTimeout(this.CONF.getHttpConnectionTimeout());
}

if(this.CONF.getHttpReadTimeout() > 0) {
    ((HttpURLConnection)v6).setReadTimeout(this.CONF.getHttpReadTimeout());
}

((HttpURLConnection)v6).setInstanceFollowRedirects(false);
return ((HttpURLConnection)v6);
```

c) obtain network information

```
if(bC.a == null) {
    try {
        Object v5 = arg5.getSystemService("wifi");
        if(v5 == null) {
            goto label_25;
        }

        WifiInfo v5_1 = ((WifiManager)v5).getConnectionInfo();
        if(v5_1 == null) {
            goto label_25;
        }

        String v5_2 = v5_1.getMacAddress();
        bC.a = v5_2;
        if(v5_2 == null) {
            goto label_25;
        }

        bC.a = bC.a.toUpperCase(Locale.getDefault());
    }
    catch(Exception ) {
        bC.a = v0;
    }
}
```

4. Slime Road (com.rubygames.splashyjump.apk)

This app has some malicious activities, such as obtain SIM operator name, obtain SIM status, read phone number, connect to the Internet and obtain device information.

a) obtain SIM operator name

```
try {
    this.mNetworkOperatorName = ((TelephonyManager)v3).getNetworkOperatorName();
    if(((TelephonyManager)v3).getSimState() != v6) {
        goto label_64;
    }

    this.mSimOperatorName = ((TelephonyManager)v3).getSimOperatorName();
```

b) obtain SIM status

```
private static void loadIMEI(Activity arg4) {
    if(arg4.getSystemService("phone").getSimState() == 5) {
        GALogger.d("loadIMEI(): Sim card present. Trying to load IMEI.");
        if(Build$VERSION.SDK_INT >= 23) {
            GAPlatform.loadIMEI23AndAbove(arg4);
        }
        else {
            GAPlatform.loadIMEI22AndBelow();
        }
    }
```

c) read phone number

```
@VisibleForTesting private static String zzep(String arg3) {
    StackTraceElement[] v0 = Thread.currentThread().getStackTrace();
    if(v0.length >= 4) {
        arg3 = new StringBuilder(String.valueOf(arg3).length() + 13).append(arg3).append(" @").append(v0[3].getLineNumber()).toString();
    }

    return arg3;
}
```

d) connect to the Internet

```
public Void call() throws Exception {
    URL v1 = new URL(this.val$url.toString());
    URLConnection v0 = null;
    while(v1 != null) {
        v0 = v1.openConnection();
        if((v0 instanceof HttpURLConnection)) {
            v0.setInstanceFollowRedirects(true);
        }
```

e) obtain device information

```
try {
    v7.put("action", ((AdUnitMotionEvent)v15).getAction());
    v7.put("isObscured", ((AdUnitMotionEvent)v15).isObscured());
    v7.put("toolType", ((AdUnitMotionEvent)v15).getToolType());
    v7.put("source", ((AdUnitMotionEvent)v15).getSource());
    v7.put("deviceId", ((AdUnitMotionEvent)v15).getDeviceId());
    v7.put("x", ((double)((AdUnitMotionEvent)v15).getX()));
    v7.put("y", ((double)((AdUnitMotionEvent)v15).getY()));
    v7.put("eventTime", ((AdUnitMotionEvent)v15).getEventTime());
    v7.put("pressure", ((double)((AdUnitMotionEvent)v15).getPressure()));
    v7.put("size", ((double)((AdUnitMotionEvent)v15).getSize()));
    v18.put(Integer.toString(v6), v7);
}
```