附錄 2

1. Uplive - Live Video Streaming App (com.asiainno.uplive.apk)

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

a) obtain SIM serial number

```
private void initUUID(Context arg11) {
    Object v11 = arg11.getApplicationContext().getSystemService("phone");
    if(PermissionUtil.checkReadPhoneStatePermission()) {
        try {
            String v0_1 = "" + ((TelephonyManager)v11).getDeviceId();
            StringBuilder v3 = new StringBuilder();
            v3.append("");
            v3.append(((TelephonyManager)v11).getSimSerialNumber());
            this.uuid = new UUID(((long)this.androidId.hashCode()), (((long)v0_1.hashCode())) << 32 | (((long)v3.toString().hashCode()))).toString();
        }
        catch(Exception ) {
            this.uuid = "";
            LogUtil.d("Gio.DeviceUUIDFactory", new Object[]{"don\'t have permission android.permission.READ_PHONE_STATE,initUUID failed "});
        }
    }
```

b) obtain SIM status

```
TelephonyManager v2 = this.b.aAQ();
boolean v3 = cou.a(this.b.a);
if(v2 == null || v2.getSimState() != 5) {
    v0_1 = 0;
}
else {
}

if((v3) || v0_1 == 0) {
    v6 = 0;
}
```

c) read phone number

```
PackageInfo v1_1 = this.aAV();
v0.i = v1_1.versionName;
CharSequence v4_3 = this.a.getApplicationInfo().loadLabel(this.daY);
v4_2 = v4_3 != null ? v4_3.toString() : "unknown";
v0.j = v4_2;
TelephonyManager v4_4 = this.daZ;
if(v4_4 != null) {
    int[] v5 = new int[2];
    cou.a(v4_4, v5);
    v0.k = v5[0];
    v0.l = v5[1];
    v0.b = v4_4.getPhoneType();
    String v2_2 = cpa.a(v4_4.getDeviceId(), cpa.a);
    String v3 = cpa.a(v4_4.getSubscriberId(), cpa.b);
    v4_2 = cpa.a(v4_4.getLine1Number(), cpa.c);
    this.k = v2_2;
    v0.c = v2_2;
    v0.d = v3;
    v0.e = v4_2;
}
```

d) connect to the Internet

```
private byte[] a(HttpURLConnection arg5) {
    byte[] v5_2;
    ByteArrayOutputStream v2;
    InputStream v1;
    byte[] v0 = null;
    try {
        v1 = arg5.getInputStream();
    }
    catch(Throwable v5) {
        v1 = ((InputStream)v0);
        v2 = ((ByteArrayOutputStream)v1);
        goto label_44;
    }
    catch(Exception v5_1) {
        v2 = ((ByteArrayOutputStream)v0);
        goto label_41;
    }

    try {
```

e) obtain network information

```
try {
    v6 = this.getWifiInfo(arg6);
    v2 = null;
    if(v6 != null && ("02:00:00:00:00:00".equals(v6.getMacAddress()))) {
        try {
            v6_1 = this.getAdressMacByInterface();
        }
        catch(Exception ) {
            goto label_20;
        }

        try {
            if(TextUtils.isEmpty(((CharSequence)v6_1))) {
                v2 = this.getAddressMacByFile(this.wifiManager);
                goto label_30;
            }
            else {
                goto label_17;
            }
        }
        catch(Exception ) {
            v2 = v6_1;
            goto label_20;
        }
```

f) obtain device information

```
private void initUUID(Context arg11) {
    Object v11 = arg11.getApplicationContext().getSystemService("phone");
    if(PermissionUtil.checkReadPhoneStatePermission()) {
        try {
            String v0_1 = "" + ((TelephonyManager)v11).getDeviceId();
            StringBuilder v3 = new StringBuilder();
            v3.append("");
            v3.append(((TelephonyManager)v11).getSimSerialNumber());
            this.uuid = new UUID(((long)this.androidId.hashCode()), (((long)v0_1.hashC
        }
        catch(Exception ) {
            this.uuid = "";
            LogUtil.d("Gio.DeviceUUIDFactory", new Object[]{"don\'t have permission an
        }
    }
    else {
        LogUtil.d("Gio.DeviceUUIDFactory", new Object[]{"don\'t have permission androi
    }
}
```

2. Weather forecast (com.tohsoft.appweb.weather.apk)

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

a) obtain SIM operator name, obtain SIM serial number, read phone number, obtain device information

```
@SuppressLint(value={"HardwareIds"}) public static String getPhoneStatus() {
    Object v0 = Utils.getApp().getSystemService("phone");
    String v1 = "" + "DeviceId(IMEI) = " + ((TelephonyManager)v0).getDeviceId() + "\n";
    v1 = v1 + "DeviceSoftwareVersion = " + ((TelephonyManager)v0).getDeviceSoftwareVersion() + "\n";
    v1 = v1 + "Line1Number = " + ((TelephonyManager)v0).getLine1Number() + "\n";
    v1 = v1 + "NetworkCountryIso = " + ((TelephonyManager)v0).getNetworkCountryIso() + "\n";
    v1 = v1 + "NetworkOperator = " + ((TelephonyManager)v0).getNetworkOperator() + "\n";
    v1 = v1 + "NetworkOperatorName = " + ((TelephonyManager)v0).getNetworkOperatorName() + "\n";
    v1 = v1 + "NetworkType = " + ((TelephonyManager)v0).getNetworkType() + "\n";
    v1 = v1 + "PhoneType = " + ((TelephonyManager)v0).getPhoneType() + "\n";
    v1 = v1 + "SimCountryIso = " + ((TelephonyManager)v0).getSimCountryIso() + "\n";
    v1 = v1 + "SimOperator = " + ((TelephonyManager)v0).getSimOperator() + "\n";
    v1 = v1 + "SimOperatorName = " + ((TelephonyManager)v0).getSimOperatorName() + "\n";
    v1 = v1 + "SimSerialNumber = " + ((TelephonyManager)v0).getSimSerialNumber() + "\n";
    v1 = v1 + "SimState = " + ((TelephonyManager)v0).getSimState() + "\n";
    v1 = v1 + "SubscriberId(IMSI) = " + ((TelephonyManager)v0).getSubscriberId() + "\n";
    return v1 + "VoiceMailNumber = " + ((TelephonyManager)v0).getVoiceMailNumber() + "\n";
}
```

b) obtain network information

```
public static String getMacAddress() {
    return DeviceUtils.getMacAddress(null);
}
```

3. Polysphere (com.playgendary.polyspherecoolgame.apk)

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

a) obtain SIM operator name

```
        this.[illegible] = true;
        Object v0_1 = v1.getSystemService("phone");
        this.[illegible] = ((TelephonyManager)v0_1).getSimOperatorName();
        this.[illegible] = ((TelephonyManager)v0_1).getNetworkOperatorName();
        this.[illegible] = Integer.valueOf(((TelephonyManager)v0_1).getPhoneType());
        this.[illegible] = p.[illegible]();
        this.[illegible] = p.[illegible](v1);
    }
    catch(Exception v0) {
        o.[illegible](v0);
    }
```

b) obtain SIM status

```
Object v1_2 = arg10.getSystemService("phone");
if(v1_2 != null && ((TelephonyManager)v1_2).getPhoneType() != 0 && ((TelephonyManager)v1_2).getSimState() == 5) {
    try {
        v2 = ((TelephonyManager)v1_2).getSimOperator();
        v4 = v2;
    }
    catch(Exception v3) {
        a.a(Chartboost.class, "Unable to retrieve sim operator information", v3);
        v4 = v2;
    }
```

c) read phone number

```
static String createStacktrace() {
    StackTraceElement[] v6 = Thread.currentThread().getStackTrace();
    StringBuilder v5 = new StringBuilder("Assembly trace:");
    StackTraceElement[] v0 = v6;
    int v3 = v0.length;
    int v2;
    for(v2 = 0; v2 < v3; ++v2) {
        StackTraceElement v1 = v0[v2];
        String v4 = v1.toString();
        if((OnSubscribeOnAssembly.fullStackTrace) || v1.getLineNumber() > 1 &&
            v5.append("\n at ").append(v4);
        }
    }
```

d) connect to the Internet

```
public void run() {
    String v0_3;
    String v1_3;
    URLConnection v1;
    URLConnection v0_2;
    String v2;
    try {
        v2 = this.a;
        if(!this.e.a.b()) {
            goto label_51;
        }
    }
    catch(Exception v0) {
        goto label_39;
    }

    try {
        v0_2 = new URL(this.a).openConnection();
        v1 = null;
        goto label_11;
    }
    catch(Throwable v0_1) {
    }
    catch(Exception v0) {
```

4. Fruit Slice (com.slice.cut10122018.apk)

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

a) obtain SIM operator name

```
public static String getTelcoName(Context arg3) {
    String v1;
    try {
        v1 = arg3.getSystemService("phone").getSimOperatorName();
        if(v1 == null) {
            v1 = "";
        }
    }
    catch(Exception v0) {
        v0.printStackTrace();
        return "";
    }
```

b) obtain SIM status

```
private void fillSimDetails(TelephonyManager arg3) {
    if(arg3.getSimState() == 5) {
        this.setIsp(arg3.getSimOperator());
        this.setIspName(arg3.getSimOperatorName());
    }
}
```

c) read phone number

```
v1.append(arg6.getClassName());
v1.append(".");
v1.append(arg6.getMethodName());
v0 += CodedOutputStream.computeBytesSize(2, ByteString.copyFromUtf8(v1.toString()));
if(arg6.getFileName() != null) {
    v0 += CodedOutputStream.computeBytesSize(3, ByteString.copyFromUtf8(arg6.getFileName()));
}

if(!arg6.isNativeMethod() && arg6.getLineNumber() > 0) {
    v0 += CodedOutputStream.computeUInt64Size(4, ((long)arg6.getLineNumber()));
}
```

d) obtain network information

```
v1.append(arg6.getClassName());
v1.append(".");
v1.append(arg6.getMethodName());
v0 += CodedOutputStream.computeBytesSize(2, ByteString.copyFromUtf8(v1.toString()));
if(arg6.getFileName() != null) {
    v0 += CodedOutputStream.computeBytesSize(3, ByteString.copyFromUtf8(arg6.getFileName()));
}

if(!arg6.isNativeMethod() && arg6.getLineNumber() > 0) {
    v0 += CodedOutputStream.computeUInt64Size(4, ((long)arg6.getLineNumber()));
}
```