微信反编译密码的主要过程

（1）使用apktool工具反编译apk

（2）使用Eclipse工具把反编译的代码导入，进行全局的关键类搜索（因为里面的代码还是smali规则的）

（2）使用Jadx可视化工具在进行具体的代码分析，根据上一个步骤在这里具体分析代码

1、使用反编译工具apktool 把微信的apk进行反编译，使用的命令为：

java -jar apktool_2.0.1.jar d -d weixin.apk -o weixin_oat

（这里的命令中加上了-d的参数了，是为了反编译之后生成的是java文件，而不是smali文件。便于后面能够导入Eclipse中）

这里必须清楚 apktool的版本问题，

（1）1.5.2版本没效果

```
D:\apktool>java -jar apktool_1.5.2.jar d -d weixin.apk -o weixin_out -f
Apktool v1.5.2 - a tool for reengineering Android apk files
Copyright 2010 Ryszard Wi?niewski <brut.alll@gmail.com>
with smali v1.4.1, and baksmali v1.4.1
Updated by @iBotPeaches <connor.tumbleson@gmail.com>
Apache License 2.0 (http://www.apache.org/licenses/LICENSE-2.0)

Usage: apktool [-q|--quiet OR -v|--verbose] COMMAND [...]

COMMANDs are:

    d[ecode] [OPTS] <file.apk> [<dir>]
        Decode <file.apk> to <dir>.

        OPTS:

        -s, --no-src
            Do not decode sources.
        -r, --no-res
            Do not decode resources.
        -d, --debug
            Decode in debug mode. Check project page for more info.
        -b, --no-debug-info
            Baksmali -- don't write out debug info (.local, .param, .line, etc.)

        -f, --force
            Force delete destination directory.
        -t <tag>, --frame-tag <tag>
            Try to use framework files tagged by <tag>.
        --frame-path <dir>
            Use the specified directory for framework files
        --keep-broken-res
            Use if there was an error and some resources were dropped, e.g.:
            "Invalid config flags detected. Dropping resources", but you
            want to decode them anyway, even with errors. You will have to
            fix them manually before building.

    b[uild] [OPTS] [<app_path>] [<out_file>]
        Build an apk from already decoded application located in <app_path>.

        It will automatically detect, whether files was changed and perform
        needed steps only.

        If you omit <app_path> then current directory will be used.
        If you omit <out_file> then <app_path>/dist/<name_of_original.apk>
        will be used.

        OPTS:

        -f, --force-all
            Skip changes detection and build all files.
        -d, --debug
            Build in debug mode. Check project page for more info.
        -a, --aapt
            Loads aapt from specified location.

    if|install-framework <framework.apk> [<tag>] --frame-path [<location>]
```

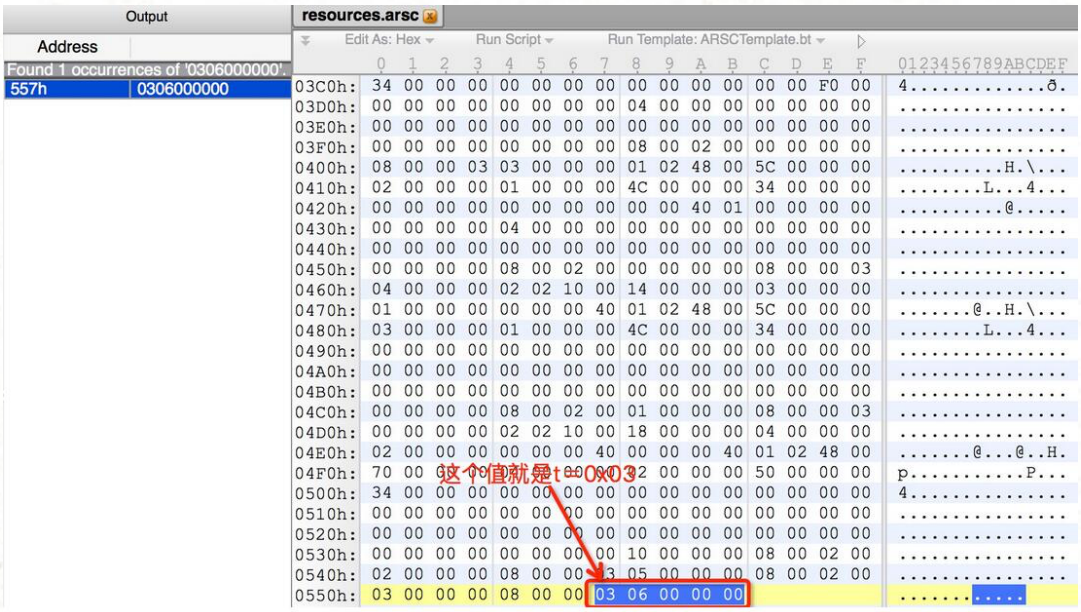
（2）2.3.1版本又把smaliDebugging移除了，也没效果！

```
D:\apktool>java -jar apktool_2.3.1.jar d -d weixin.apk -o weixin_out -f
SmaliDebugging has been removed in 2.1.0 onward. Please see: https://github.com/
iBotPeaches/Apktool/issues/1061
```

（3）最终选定了2.0.1版本，但是这个版本反编译也不顺利 出现了问题

```
D:\apktool>java -jar apktool_2.0.1.jar d -d weixin.apk -o weixin_out -f
I: Using Apktool 2.0.1 on weixin.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: C:\Users\Administrator\apktool\framework\1.
apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
Exception in thread "main" brut.androlib.err.UndefinedResObject: resource spec:
0x010104e2
        at brut.androlib.res.data.ResPackage.getResSpec(ResPackage.java:59)
        at brut.androlib.res.data.ResTable.getResSpec(ResTable.java:65)
        at brut.androlib.res.data.ResTable.getResSpec(ResTable.java:61)
        at brut.androlib.res.data.value.ResReferenceValue.getReferent(ResReferen
ceValue.java:57)
        at brut.androlib.res.data.value.ResReferenceValue.encodeAsResXml(ResRefe
renceValue.java:47)
        at brut.androlib.res.data.value.ResScalarValue.encodeAsResXmlValue(ResSc
alarValue.java:58)
        at brut.androlib.res.data.value.ResStyleValue.serializeToResValuesXml(Re
sStyleValue.java:71)
        at brut.androlib.res.AndrolibResources.generateValuesFile(AndrolibResour
ces.java:496)
        at brut.androlib.res.AndrolibResources.decode(AndrolibResources.java:252
)
        at brut.androlib.Androlib.decodeResourcesFull(Androlib.java:134)
        at brut.androlib.ApkDecoder.decode(ApkDecoder.java:104)
        at brut.apktool.Main.cmdDecode(Main.java:165)
        at brut.apktool.Main.main(Main.java:81)
```

这个是android apk的保护机制在作怪，所以我们使用010Editor二进制编辑器修改了微信apk包解压出来的resources.arsc文件，

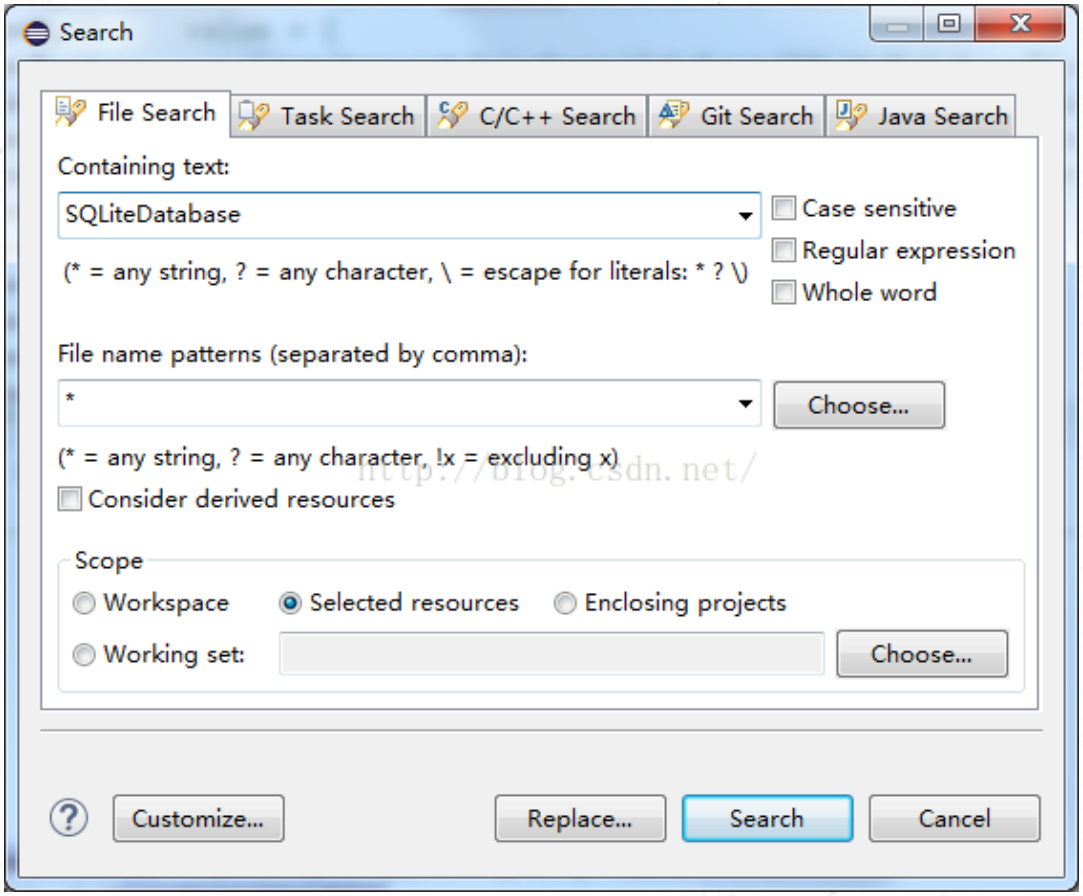


在0306000000位置修改，把03改成02，然后压缩成apk文件之后就可以了。

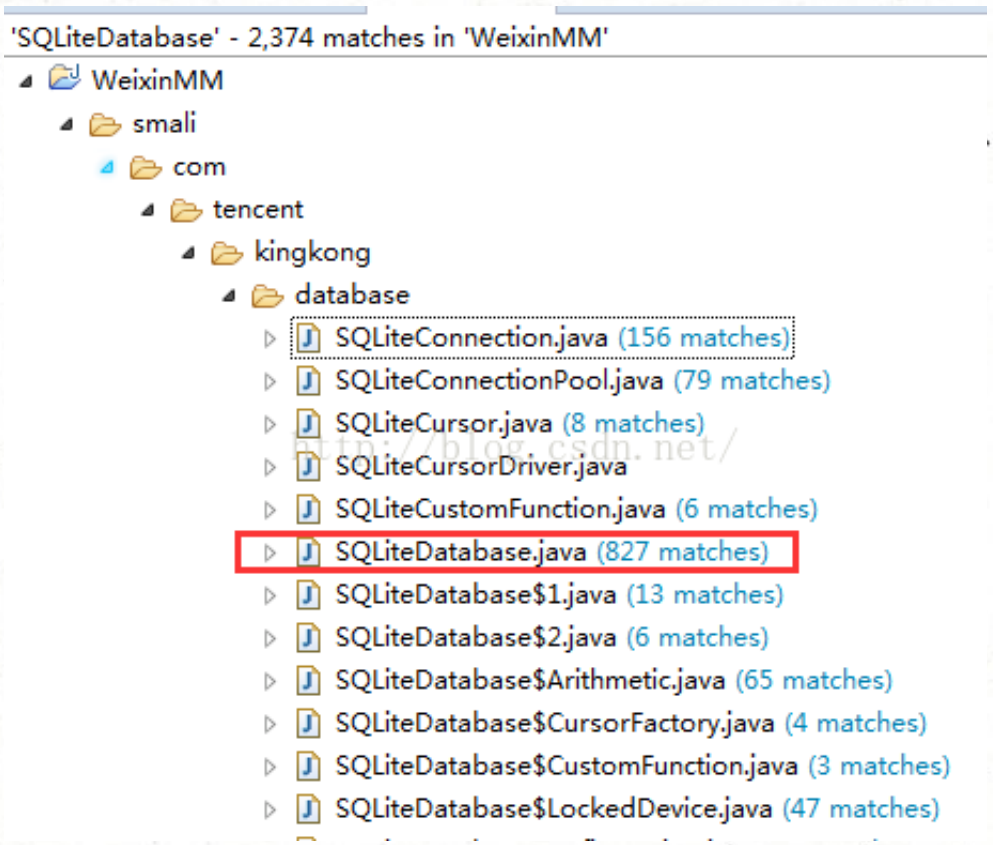
```
D:\apktool>java -jar apktool_2.0.1.jar d -d weixin.apk -o weixin_out -f
I: Using Apktool 2.0.1 on weixin.apk
I: Baksmaling weixin/classes.dex...
I: Baksmaling weixin/classes2.dex...
I: Baksmaling weixin/classes3.dex...
I: Baksmaling weixin/classes4.dex...
I: Baksmaling weixin/classes5.dex...
I: Baksmaling weixin/classes6.dex...
I: Baksmaling weixin/classes7.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
```

2、Eclipse导入项目工程

把反编译的工程倒入Eclipse项目中 然后再全局搜索SQLiteDatabase，我们数据库相关的就是这个类了，



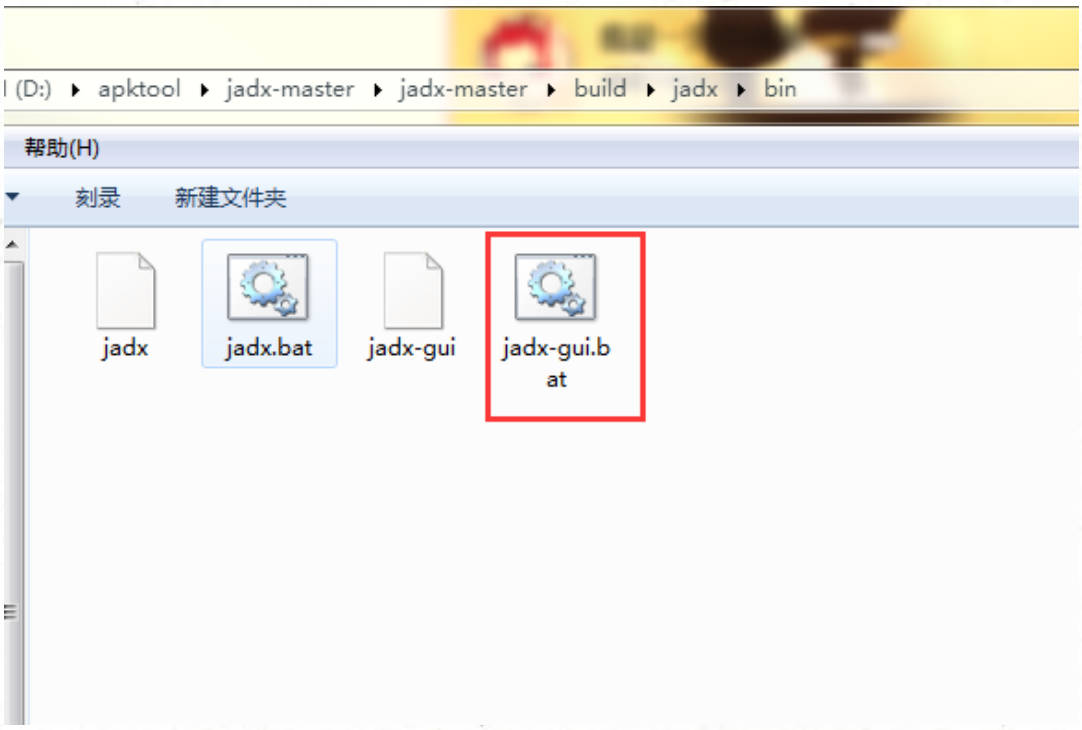
在结果页面中搜索的结果如下



我们记录下这个类的路径，接下来我们要查看这个类的源码。

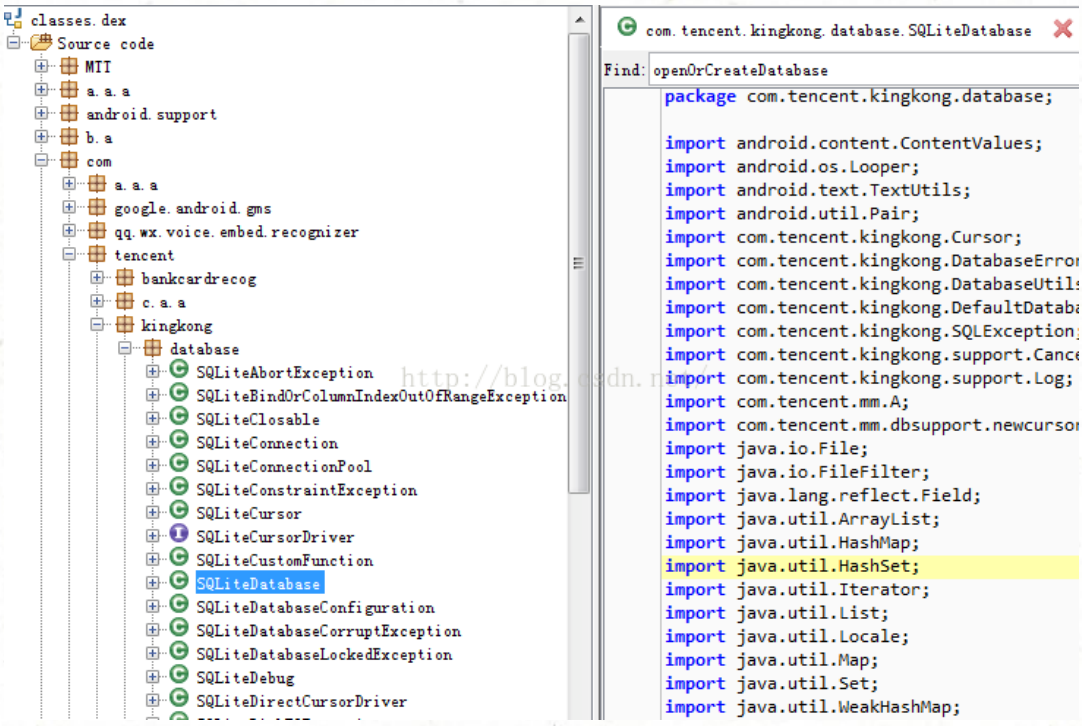
3、Jadx工具可视化

我们使用Jadx工具进行可视化操作，而且他是开源的，而且用的是java编写的，对于混淆代码的跟踪非常有用。他的Github地址：https://github.com/skylot/jadx，下载之后在文件目录下 进行命令行操作（./gradlew dist） 下载完毕之后就可以点击bin目录下的jadx-gui，然后选择我们要反编译的微信apk文件，由于微信的apk太大，我们选择查看apk中的dex文件（微信有多个dex文件）



4、源码分析

恰巧这个SQLieDatabase在第一个classes.dex文件中了，我们可以根据apktool反编译的路径的结果在这里找到要查看的类



这样看起来就方便很多了，我们找到这个类，首先肯定看看他的openDatabase方法，不过这里会发现有很多重载方法，不过最终都要调用的是

这个openDatabase方法：

```
public static SQLiteDatabase openDatabase(String str, LockedDevice lo
    SQLiteDatabase sQLiteDatabase = new SQLiteDatabase(str, i, cursor
    sQLiteDatabase.open(lockedDevice, str2, arithmetic, z, i2);
    return sQLiteDatabase;
}
```

内部接着调用了open方法，继续跟进：

```
private void open(LockedDevice lockedDevice, String str, Arithr
    try {
        openInner(lockedDevice, str, arithmetic, z, i);
    } catch (SQLiteDatabaseCorruptException e) {
        try {
            onCorruption();
            openInner(lockedDevice, str, arithmetic, z, i);
        } catch (SQLiteException e2){
            Log.e(TAG, "Failed to open database '" + getLabel(
            close();
            throw e2;
        }
    }
}
```

内部又调用了openInner方法，接着跟进：

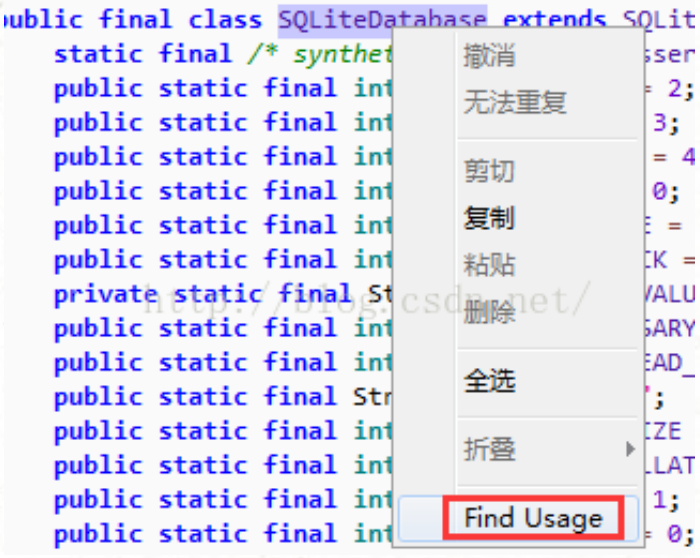
```
private void openInner(LockedDevice lockedDevice, String str, Arithmetic arithmetic, boolean z, int i) {
    synchronized (this.mLock) {
        if ($assertionsDisabled || this.mConnectionPoolLocked == null) {
            this.isEncrypt = !TextUtils.isEmpty(str);
            this.mConnectionPoolLocked = SQLiteConnectionPool.open(this.mConfigurationLocked, lockedDevice,
        } else {
            throw new AssertionError();
        }
    }
    synchronized (sActiveDatabases) {
        sActiveDatabases.put(this, null);
    }
}
```

调用了SQLiteConnectionPool的open方法，再跟进去：

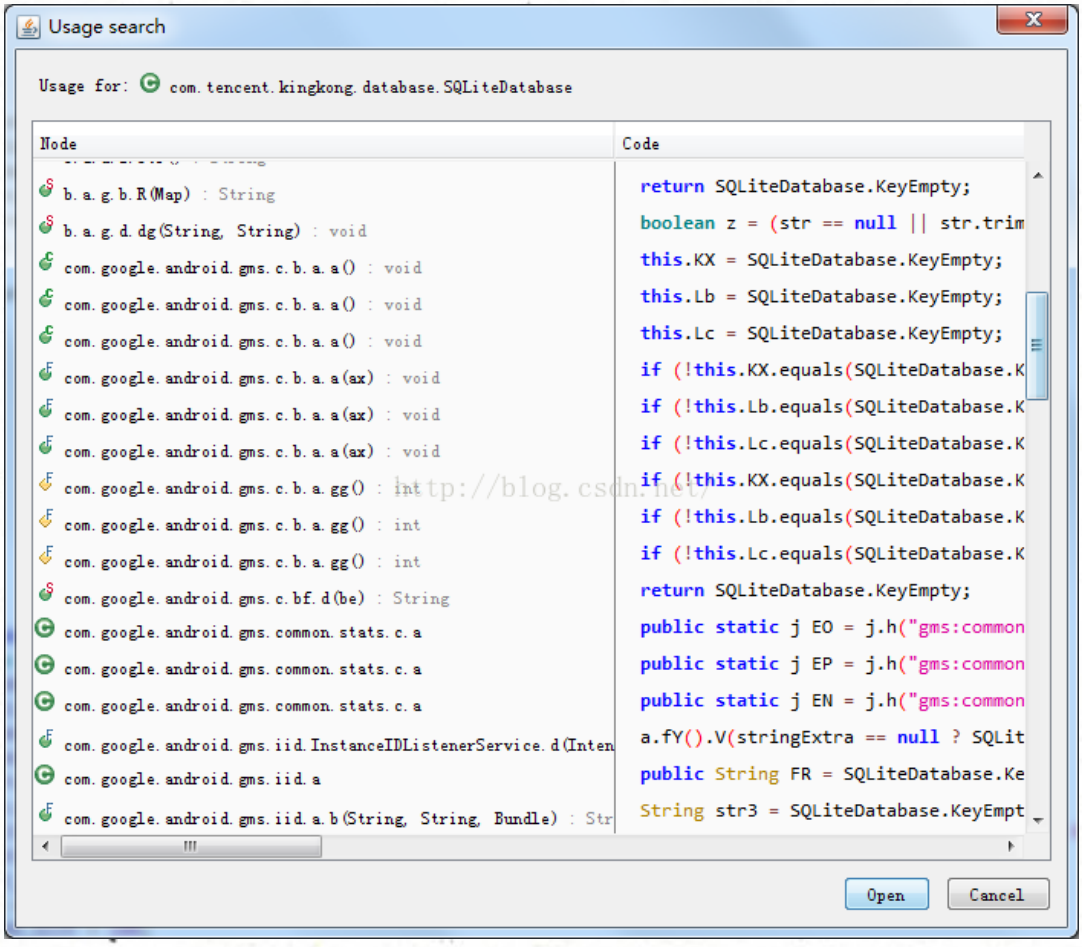
```
public static SQLiteConnectionPool open(SQLiteDatabaseConfiguration sQLite
    if (sQLiteDatabaseConfiguration == null) {
        throw new IllegalArgumentException("configuration must not be null
    }
    SQLiteConnectionPool sQLiteConnectionPool = new SQLiteConnectionPool(s
    sQLiteConnectionPool.mIsInitWaited = z;
    sQLiteConnectionPool.mlockedDevice = lockedDevice.getValue();
    sQLiteConnectionPool.mPassword = str;
    sQLiteConnectionPool.mArithmetic = arithmetic.getValue();
    sQLiteConnectionPool.open();
    return sQLiteConnectionPool;
}
```

关键的东东看到了，密码！！

终于找到核心的地方了，这里看到果然有一个密码的字段，那么这个值就是SQLiteDatabase中的openDatabase方法的第二个参数，那么现在我们就去分析哪里调用了SQLiteDatabase的openDatabase方法，因为SQLiteDatabase的openDatabase的重载方法太多了，所以一个一个找很费劲，所以可以直接搜SQLiteDatabase被调用的地方，可以直接使用Jadx的查找跟踪功能：

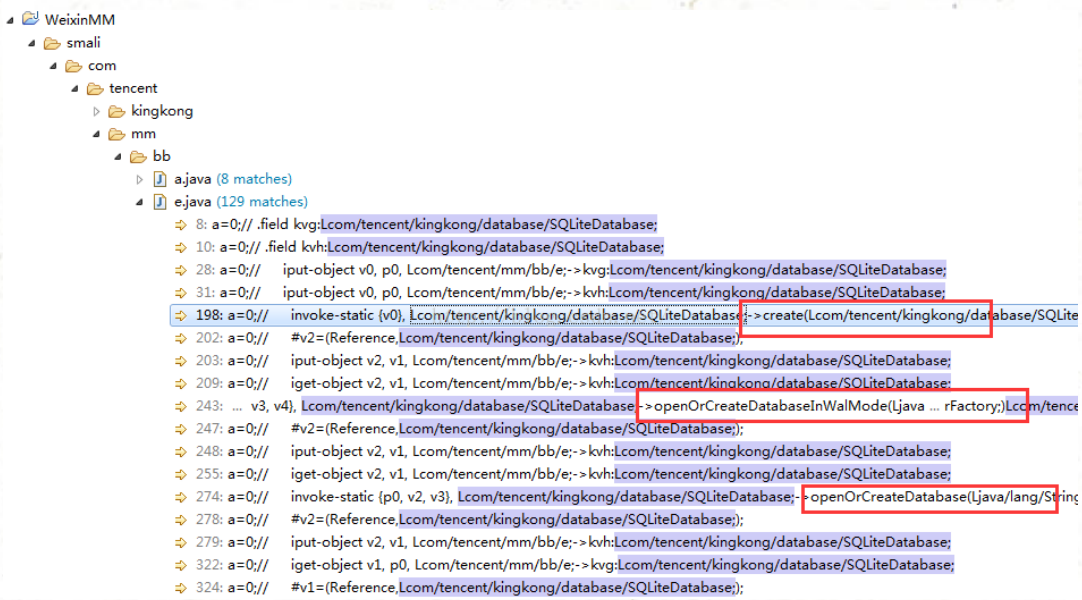


查找结果：

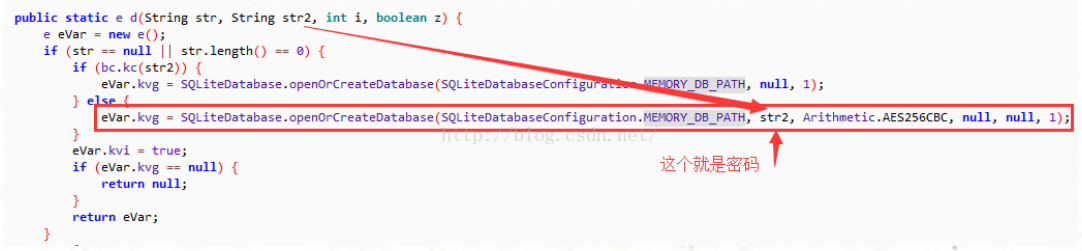


这里会发现，很多地方都调用了，看起来非常麻烦，所以这里得想个办法缩减查找范围，我们刚刚看到SQLiteDatabase类中的open方法都是static的，所以在调用的时候肯定是这么使用的：

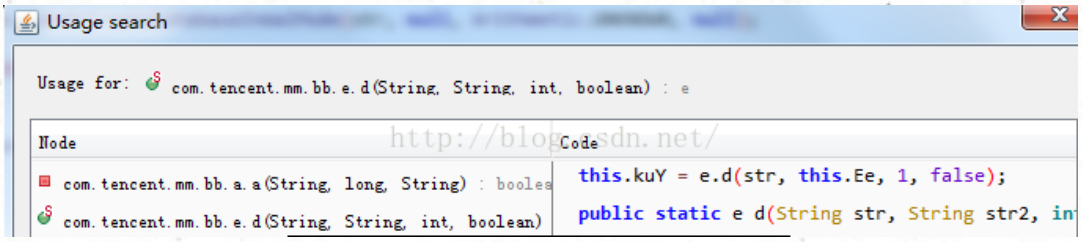
- Lcom/tencent/kingkong/database/SQLiteDatabase; 这个是标准的smali语法调用形式，所以这时候我们在去Eclipse中全局搜索这个字符串内容：



最终看到在com.tencent.mm.bb.e这个类中，有多个地方都调用了，咋们再去看看这个类：



果然在d方法中调用了数据库的open方法，而且传入的str2就是密码，在跟踪d方法在哪里被调用了：



点击进入查看：

```
this.Ee = com.tencent.mm.a.g.j((str2 + j).getBytes()).substring(0, 7);
this.kuY = e.d(str, this.Ee, 1, false);
if (this.kuY == null) {
    gVar = g.gar;
    g.b(181, 0, 1, false);
    this.aps = "Endbinit open failed: [" + j + "] dev: [" + str2 + "]";
    return false;
}
```

这里的this.Ee就是密码，看到他的赋值情况，调用了j方法获取一个字符串，然后取前7个字符

这里的this.Ee就是密码，看他的赋值，是先调用j方法构造一个字符串出来，然后取前7个字符即可，再看看j方法：

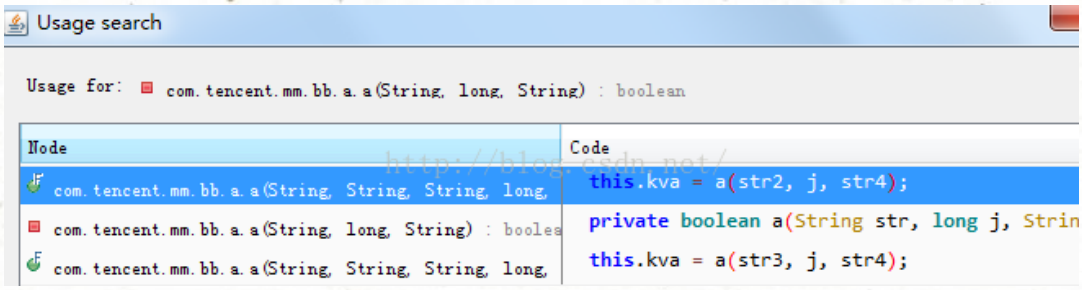
```
public static final String j(byte[] bArr) {
    int i = 0;
    char[] cArr = new char[]{'0', '1', '2', '3', '4', '5', '
    try {
        MessageDigest instance = MessageDigest.getInstance("
        instance.update(bArr);
        byte[] digest = instance.digest();
        int length = digest.length;
        char[] cArr2 = new char[(length * 2)];
        int i2 = 0;
        while (i < length) {
            byte b = digest[i];
            int i3 = i2 + 1;
            cArr2[i2] = cArr[(b >>> 4) & 15];
            i2 = i3 + 1;
            cArr2[i3] = cArr[b & 15];
            i++;
        }
        return new String(cArr2);
    } catch (Exception e) {
        return null;
    }
}
```


这个方法看起来就眼熟了，计算MD5值的

这个方法看起来就眼熟了，计算字符串的MD5值的，这里需要注意的是，MD5的值是小写的，好了，到这里我们就了解了：密码其实是一个字符串的MD5值的前7位字符串，那么接下来的问题在继续跟踪是哪个字符串来计算MD5的：

```
private boolean a(String str, long j, String str2) {
    int Hy = Hy(str);
    Editor edit = PreferenceManage .getDefaul SharedPreferences(z.getContext(
    if (Hy > 2) {
        v.e("MicroMsg.DBInit", "db  rash cou t extends limit ,rename db ");
        e.renameTo(new File(str), ne  File(st r + "err" + bc.Gq()));
        edit.putInt(str, 0).commit()
        g.gar.h(11670, new Object[0]);
        g gVar = g.gar;
        g.b(181, 2, 1, false);
    }
    this.Ee = com.tencent.mm.a.g.j((str2 + j).getBytes()).substring(0, 7);
    this.kuY = e.d(str, this.Ee, 1, false);
    if (this.kuY == null) {
```

在继续跟踪a方法在哪里被调用了：



找到之后点击进入：

```
public final boolean a(String str, String str2, String str3, long j, String str4, HashMap hashMap, boolean z) {
    Object obj;
    Assert.assertTrue("create SqliteDB enDbCachePath == null", !bc.kc(str2));
    this.kvb = str2 + ".errreport";
    if (this.kuY != null) {
        this.kuY.close();
        this.kuY = null;
    }
    boolean aB = e.aB(str2);
    boolean z2 = !aB || Hy(str2) > 2;
    boolean aB2 = e.aB(str);
    boolean aB3 = e.aB(str3);
    Object obj2 = (aB || !aB2) ? null : 1;
    Object obj3 = null;
    this.kva = a(str2, j, str4);
    String str5 = "MicroMsg.DBInit";
```

继续查找这个a方法又在哪里被调用了，这里调用的比较深，所以需要多次进行查找跟踪，耐心点即可：


```
        j = this.cachePath + "MicroMsg.db";
        String str = this.cachePath + "EnMicroMsg.db";
        String str2 = this.cachePath + "EnMicroMsg2.db";
        cy(null);
        this.brX = new g(new com.tencent.mm.bb.g.a(this) {
            final /* synthetic */ c bsn;

            {
                this.bsn = r2;
                if (Boolean.FALSE.booleanValue()) {
                    A.a();
                }
            }

            public final void rQ() {
                if (this.bsn.brG != null) {
                    com.tencent.mm.sdk.platformtools.v.i("MicroMsg.Accou
                    this.bsn.brG.hg(true);
                }
                com.tencent.mm.modelstat.f DL = com.tencent.mm.modelstat
                if (DL != null) {
                    com.tencent.mm.sdk.platformtools.v.i("MicroMsg.Accou
                    long currentTimeMillis = System.currentTimeMillis();
                    DL.bXz.hd(true);
                    com.tencent.mm.sdk.platformtools.v.i("MicroMsg.NetSt
                }
            }

            public final void rR() {
            }
        });
        g gVar = this.brX;
        long j2 = (long) i;
        String mY = p.mY();
        HashMap hashMap = new HashMap();
        hashMap.putAll(aZn);
        hashMap.putAll(ah.tb().uB());
        if (gVar.a(j, str, str2, j2, mY, hashMap, true)) {
            String str3 = this.brX.kVq;
```

最终到了这个类的方法中，我们看到，mY值是通过mY方法获取的，j2值是上面的i值转化过来的：

```
public static String mY() {
    String str = (String) k.mK().get(258);
    if (str == null) {
        str = getDeviceID(z.getContext());
        if (str == null) {
            str = "1234567890ABCDEF";
        }
        k.mK().set(258, str);
    }
    return str;
}
```

获取设备的IMEI值

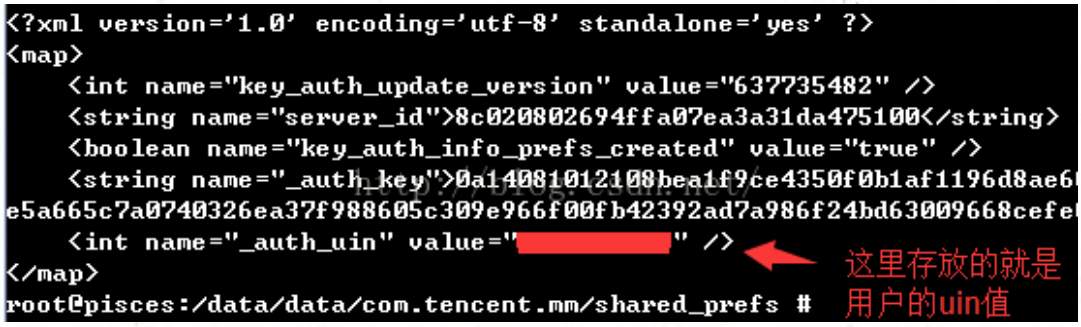
查看mY方法的实现，很简单，获取设备的IMEI值，而i值在前面进行赋值了：

```
final void cZ(int i) {
    if (this.bsm != null) {
        this.bsm.clear();
    }
    SharedPreferences sharedPre
    if (i == 0) {
        if (this.uin != 0) {
            release();
        }
        i = this.uin;
```

看到了，是一个uin值，再看看这个uin值在哪里进行赋值操作的：

```
public final SharedPreferences dP(String str) {
    if (this.uin == 0) {
        return null;
    }
    if (this.brZ.containsKey(str)) {
        return (SharedPreferences) this.brZ.get(str);
    }
```


看到这里就放心了，原来这个uin值存放在SharedPreferences中的，那么就简单了，我们在开始的时候把沙盒数据全部导出来了，可以全局搜一下uin字符串的值：



在这里找到了这个值。

总结：

到这里我们就分析完了微信中数据库加密的密码获取流程了，下面来总结一下：

1、首先我们全局查找SQLiteDatabase类，因为这个类是操作数据库的核心类，他是突破口。

2、找到这个类的定义之后，再次查看他的open系列方法，因为要操作数据库肯定有open之类的方法。

3、再去全局查找SQLiteDatabase的open方法的调用地方，这里调用的地方比较多，所以大家需要耐心的查找，而且为了缩小查找范围，我们可以根据smali语法调用格式的字符串内容来进行查找。

4、找到了这个方法的被盗用的地方，下面就开始一步一步的往下跟踪，到了一个核心的方法中了解到了，密码是一个字符串计算MD5之后取前7个字符串的值。

5、继续跟踪，找到这个被计算MD5的字符串内容，最后跟踪到这个字符串其实是设备的IMEI加上用户的uin值，而这个uin值是保存在SharedPreferences中的。