SHA256: 159a7af39c0d6c2334df77088fe2d545a96d591dbf2b85c373a4a45377f492c4

Analysis: The following analysis is just a quick run through while practicing on APK files.

Running the file through Detect-it-Easy shows that the file is an APK file protected by DexGuard.

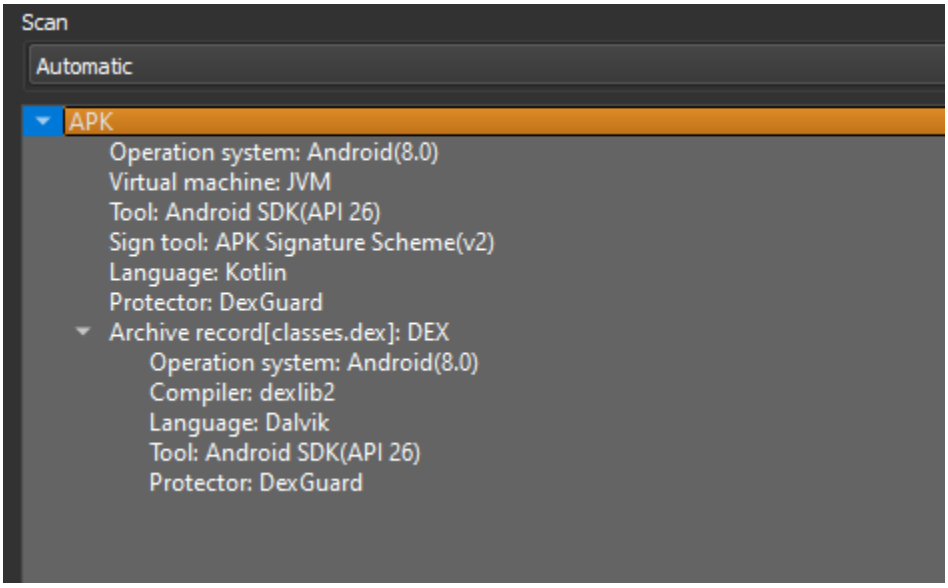

Figure 1. D-i-E Analysis on File Type

The file also appears to be querying some kind of operating system files related to Chinese manufacturers such as Oppo, Vivo, Xiaomi, and Huawei. It is currently unclear what querying these information is used for at this moment.


```
  "myself":true,
  "enable": true,
  "action":"home",
  "ids":[],
  "pgk":[
    "com.android.settings",
    "com.vivo.permissionmanager",
    "com.huawei.appmarket",
    "com.xiaomi.market",
    "com.bbk.appstore",
    "com.huawei.systemmanager",
    "com.miui.securitycenter",
    "com.coloros.securitypermission",
    "com.iqoo.secure",
    "com.coloros.oppoguardelf",
    "com.samsung.accessibility",
    "com.huawei.android.launcher",
    "com.oplus.battery",
    "com.google.android.permissioncontroller",
    "com.oplus.safecenter",
    "com.oplus.trafficmonitor",
    "com.google.android.apps.wellbeing",
    "com.vivo.abe",
    "com.tct.onetouchbooster",
    "com.samsung.android.lool"
  ],
  "clz":[],
  "text":[],
  "noUse":[],
  "note":"001"
},{
  "myself":true,
  "enable": true,
  "action":"home",
  "ids":[],
  "pgk":[],
  "clz":[
    "com.android.settings.subsettings",
    "com.android.settings.cleansubsettings",
    "com.vivo.settings.vivosubsettings",
    "com.samsung.accessibility.core.winset.activity.subsettings"
  ],
  "text":[],
  "noUse":[],
  "note":"002"
},{
  "myself":true
```


Figure 2. Querying of OS’es

Figure 3. More Screenshots

```
    "com.android.settings.datausage.AppDataUsageActivity",
    "com.miui.optimizecenter.onekeyclean.MemoryCleanActivity",
    "com.miui.backup.local.LocalHomeActivity",
    "com.miui.backup.pc.PCBackupActivity",
    "com.miui.backup.settings.MoreSettingsActivity",
    "com.miui.backup.settings.LocalBackupManagerActivity",
    "com.miui.cloudservice.ui.MiCloudEntranceActivity",
    "com.xiaomi.market.ui.LocalAppsActivity",
    "com.android.updater.MainActivity",

    "com.huawei.localBackup.InitializeActivity",
    "com.huawei.android.backup.base.activity.OuterMediumSelectionActivity",
    "com.huawei.android.backup.base.activity.AddShareFolderActivity",
    "com.huawei.android.backup.base.activity.BackupToPcActivity",
    "com.huawei.android.hwouc.ui.activities.MainEntranceActivity",
    "com.huawei.systemmanager.power.ui.DetailOfSoftConsumptionActivity",
    "com.huawei.systemmanager.netassistant.traffic.appdetail.AppDetailActivity",
    "com.samsung.android.scloud.bnr.ui.screen.nonspinner.backup.DashboardBackupActivity",
    "com.samsung.android.scloud.bnr.ui.screen.deviceinfo.restore.BackupDeviceListActivity",
    "com.sec.android.easyMover.ui.MainActivity","com.sec.android.easyMover.Agent.DialogActivity",
    "com.android.packageinstaller.permission.ui.ReviewAccessibilityServicesActivity",

    "com.samsung.android.forest.apptimer.ui.applist.AppListActivity",
    "com.android.permissioncontroller.permission.ui.ReviewAccessibilityServicesActivity",
    "com.motorola.ccc.ota.ui.BaseActivity",
    "com.cyin.himgr.clean.view.CleanActivity",
    "com.android.settings.Settings$HighPowerApplicationsActivity",
    "com.android.settings.Settings$AccessibilitySettingsActivity",
    "com.android.settings.applications.specialaccess.deviceadmin.DeviceAdminAdd",
    "com.oplus.settings.feature.security.OplusDeviceAdminAdd",
    "com.coloros.settings.feature.security.ColorDeviceAdminAdd",
    "com.samsung.android.settings.applications.specialaccess.SecDeviceAdminAdd",
    "com.android.settings.Settings$SpecialAccessSettingsActivity",
    "com.android.settings.DeviceAdminAdd"
  ],
  "text": [],
  "noUse": [
    "com.android.packageinstaller.UninstallerActivity",
    "com.android.permissioncontroller.permission.ui.ManagePermissionsActivity",
    "com.google.android.finsky.unauthenticated.activity.UnauthenticatedMainActivity",
    "com.android.settings.Settings$PrivacyDashboardActivity",
    "com.miui.permcenter.settings.PrivacySettingsActivity",
    "com.tct.smartmanager.appoptimise.ui.ApopSettingActivity",
    "com.tct.smartmanager.memory.ui.MemoStateActivity",
    "com.samsung.android.sm.score.ui.ScoreBoardActivity",
    "com.android.settings.Settings$ManageApplicationsActivity",
    "com.android.packageinstaller.permission.ui.ManagePermissionsActivity",
    "com.android.settings.Settings$SecurityDashboardActivity",
    "com.google.android.apps.nbu.files.home.HomeActivity"
  ],
  "note":"005"
```

Figure 4. Bigger List

Here's another example of the strings being queried with an array of strings called "noUse" written in Pinyin. I compiled the translation of the strings along with a screenshot of all instances of this string array below. It might also be related to all of the permissions that the app wants and it wants way more than what a benign app requests for. This app requests almosts all of the available information on a mobile device that is installed including SMS texts, current battery status, screen on, and a lot of other things that no benign app would all need at the same time.

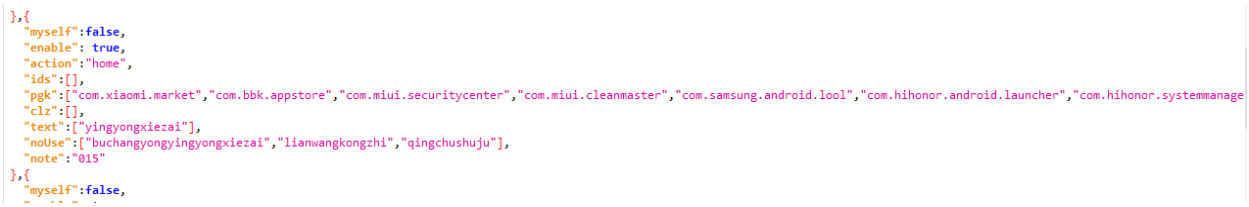

Figure 5. Another Example

Pinyin	Chinese	English
buchangyongyingyongxiezai	补偿应用写载	Compensate Application Write Load
lianwangkongzhi	联网控制	Network Control
qingchushuju	清除数据	Clear Data
qingchushuju	清除数据	Clear Data
quanxianyinsi	权限隐私	Permissions and Privacy
liuliangguanli	流量管理	Data Traffic Management
anquanshijian	安全时间	Security Time
dianchixingneng	电池性能	Battery Performance
qingchushuju	清除数据	Clear Data
wlanliuliangshiyongqingkuan	无线流量使用情况	WLAN Traffic Usage Status
qingchushuju	清除数据	Clear Data
cunchukongjian	存储空间	Storage Space
quanxianguanli	权限管理	Permission Management
yingyongsuo	应用锁	App Lock
quanxianyinsi	权限隐私	Permissions and Privacy
yinsitishen	隐私提醒	Privacy Reminder
dianchi	电池	Battery
shengdianyudianchi	省电与电池	Battery Saver
ziqidong	自启动	Auto Start
qingqiuduxie	请求读写	Request Read/Write
cunchukongjian	存储空间	Storage Space

```
"noUse": [
"noUse":["buchangyongyingyongxiezai","lianwangkongzhi","qingchushuju"],
"noUse":["qingchushuju"],
"noUse": ["quanxianyinsi","liuliangguanli","anquanshijian","dianchixingneng","qingchushuju"],
"noUse":[
"noUse":[],
"noUse":[],
"noUse":[
"noUse":[],
"noUse":[],
"noUse":[],
"noUse":["wlanliuliangshiyongqingkuan","qingchushuju","cunchukongjian","quanxianguanli","yingyongsuo","quanxianyuyinsi", "yinsitishen"
"noUse":[],
"noUse":["qingqiuduxie"],
"noUse":[],
"noUse":[],
"noUse":[],
"noUse":[],
"noUse":["cunchukongjian"],
"noUse":[
"noUse":[],
"noUse":[],
```


Figure 6. List of Appearances for “noUse”

Social media stealing functions I assume.

```
public Jfoamfat() {
    super(LoginSocmedActivityVM.class);
    this.d = new ArrayList();
    this.h = "";
    this.i = "";
}

/* JADX WARN: Multi-variable type inference failed */
public final void B(String str) {
    this.j = true;
    boolean z = ar0.a.a;
    if ("otherLogin-fb".equalsIgnoreCase(str)) {
        c.f().n("msisdnTemp");
        this.h = str;
        this.i = "Facebook";
        getViewModel().socialAuth("facebook");
        Adjust.trackEvent(new AdjustEvent(getString(R.string.adjust_login_facebook)));
        return;
    }
    if ("otherLogin-tw".equalsIgnoreCase(str)) {
        c.f().n("msisdnTemp");
        this.h = str;
        this.i = "Twiter";
        getViewModel().socialAuth("twitter");
        Adjust.trackEvent(new AdjustEvent(getString(R.string.adjust_login_twitter)));
        return;
    }
    if ("otherLogin-google".equalsIgnoreCase(str)) {
        c.f().n("msisdnTemp");
        this.h = str;
        this.i = "Google";
        getViewModel().socialAuth("google");
    }
}
```

Figure 7. Functions to Steal Social Media Credentials

Interesting loop that might be used as a key.

```
public static byte[] E() {
    byte[] bArr = new byte[32];
    for (int i = 0; i < 64; i += 2) {
        bArr[i / 2] = (byte) (Character.digit("039061dadae669b5464f107fec424632936eb022e1340414e831183429c18256".charAt(i + 1), 16) + (Character.digit("039061dadae669b5464f107fe
    }
    return bArr;
}
```

Figure 8. 32 Byte Array that loops.

Function using AES.

```
public static void z(File file, File file2, byte[] bArr) {
    y5.j("V3D-EQ-SPOOLER", "encrypt(" + file.toString() + ")");
    FileInputStream fileInputStream = new FileInputStream(file);
    FileOutputStream fileOutputStream = new FileOutputStream(file2);
    SecretKeySpec secretKeySpec = new SecretKeySpec(E(), "AES");
    E();
    y5.j("V3D-EQ-SPOOLER", "key length: 32");
    y5.j("V3D-EQ-SPOOLER", "generate IV(size): " + bArr.length);
    IvParameterSpec ivParameterSpec = new IvParameterSpec(bArr);
    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
    boolean z = true;
    cipher.init(1, secretKeySpec, ivParameterSpec);
    y5.j("V3D-EQ-SPOOLER", "" + cipher.getParameters());
    CipherOutputStream cipherOutputStream = new CipherOutputStream(fileOutputStream, cipher);
    byte[] bArr2 = new byte[8];
    while (true) {
        int read = fileInputStream.read(bArr2);
        if (read != -1) {
            if (z) {
                y5.l("V3D-EQ-SPOOLER", "bytes in:".concat(new String(bArr2, Key.STRING_CHARSET_NAME)));
                z = false;
            }
            cipherOutputStream.write(bArr2, 0, read);
        } else {
            cipherOutputStream.flush();
            cipherOutputStream.close();
            fileInputStream.close();
            return;
        }
    }
}
```

Figure 9.Usage of AES Encryption

Function that I'm assuming converts images into base64 then sends it to the C2.

```
public final class DataUrlLoader<Model, Data> implements ModelLoader<Model, Data> {
    private static final String BASE64_TAG = ";base64";
    private static final String DATA_SCHEME_IMAGE = "data:image";
    private final DataDecoder<Data> dataDecoder;
    public InputStream decode(String str) throws IllegalArgumentException {
        if (str.startsWith(DataUrlLoader.DATA_SCHEME_IMAGE)) {
            int indexOf = str.indexOf(44);
            if (indexOf != -1) {
                if (str.substring(0, indexOf).endsWith(DataUrlLoader.BASE64_TAG)) {
                    return new ByteArrayInputStream(Base64.decode(str.substring(indexOf + 1), 0));
                }
                throw new IllegalArgumentException("Not a base64 image data URL.");
            }
            throw new IllegalArgumentException("Missing comma in data URL.");
        }
        throw new IllegalArgumentException("Not a valid image data URL.");
    }

    @Override // com.bumptech.glide.load.model.DataUrlLoader.DataDecoder
    public Class<InputStream> getDataClass() {
        return InputStream.class;
    }
}
```

Figure 10. Image Stealing

There's still a lot of other functions in the file and I likely should've tried putting it on an available deobfuscator. Since this is my first attempt on an apk and due to time constraints I'll continue this practice on another time.