

HUNTING ANDROID MALWARE

A novel runtime technique for identifying malicious applications



WHOAMI



HEROKU

@brompwnie

THANK YOU

- SensePost
- Heroku

OUTLINE

The...

- Problem
- Question
- Idea
- PoC
- Results
- Conclusion

THE PROBLEM

Android malware has and is a constant threat in the
Android Ecosystem

ars TECHNICA UK [BIZ](#) [GIG](#) [TEAM](#) [SECURE](#) [POLICY](#) [LAW](#) [CAPTION](#) [CULTURE](#)

BIZ & IT — **2017**

Malicious apps with >1 million downloads slip past Google defenses twice

Malware scanners fail to detect 50 apps that charged for fake services.

DAN GOODIN (US) - 14/9/2017, 16:50

Found: New Android malware with never-before-seen spying capabilities

Skygofree is among the most powerful spy platforms ever created for Android.

2018

DAN GOODIN - 1/16/2018, 5:45 PM

Currency-mining Android malware is so aggressive it can physically harm phones

This is your phone on mining software. Any questions?

2017

DAN GOODIN - 12/19/2017, 8:40 PM

THE QUESTION

A graphic consisting of the text "www.kweta20" rendered in a green, dashed-line font. The text is centered horizontally and occupies a significant portion of the width of the slide. The background of this section is black.

July 2017 at a conference...in Las Vegas...

THE QUESTION

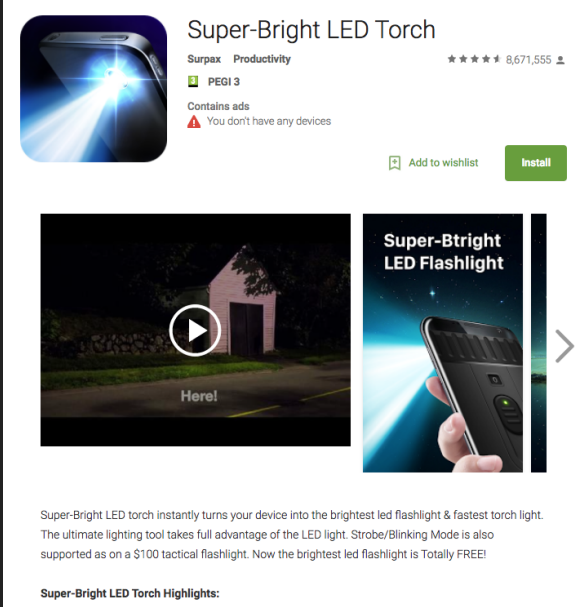
"How do I protect myself from these kind of attacks?"

- We have to look at the APK
 - Statically
 - In a sandbox

CURRENT TECHNIQUES

We look at malware in a few ways

- Hashes
- Code Signatures
- Permissions
- Reputation
- Behavior



The screenshot shows the Google Play Store listing for the app "Super-Bright LED Torch". The app is developed by "Surpax Productivity" and has a rating of 5 stars from 8,671,555 reviews. It is categorized as "Productivity" and has a PEGI 3 rating. The listing includes a green "Install" button and a note that the app "Contains ads". Below the app title, there is a video player showing a flashlight beam illuminating a building at night, with a play button icon and the word "Here!" overlaid. To the right of the video is a smaller image of the app's interface, showing a hand holding a smartphone with the flashlight app open. Below the video and images, there is a short description of the app's features and a section for highlights.

Super-Bright LED Torch

Surpax Productivity ★★★★★ 8,671,555

PEGI 3

Contains ads
⚠️ You don't have any devices

Add to wishlist **Install**

Super-Bright LED Flashlight

Super-Bright LED torch instantly turns your device into the brightest led flashlight & fastest torch light. The ultimate lighting tool takes full advantage of the LED light. Strobe/Blinking Mode is also supported as on a \$100 tactical flashlight. Now the brightest led flashlight is Totally FREE!

Super-Bright LED Torch Highlights:

CURRENT DEFENCES

- Google Play Protect
- Google Playstore
- Third-party Software
 - Anti-virus
 - MDM,MAM,MOM,MMM,...

FRUSTRATIONS

- Static Analysis is hard
- Can't run Cuckoo on my phone
- Scalability
- What if the application isn't on the App Store
- Bypassing AV is too easy

THE FRUSTRATION

No reliable way to detect malware on devices



THE IDEA

But there's HEAPS of data....

- Android apps make use of Objects
- Import statements are useful BUT
- You can import but not instantiate
- If it's instantiated, it's probably being used
- Instantiated objects have data(some)

/PROC//MAPS?

/PROC//MEM?

DUMP.HPROF?



THE IDEA

Instrumentation

- Objects on the HEAP
- Trace calls and behaviour
- Recent Developments made it easier(for me)
- Great way to gain insight into applications
- Extremely Powerful
- Runtime is the best action

FRIDA

<https://www.frida.re/>

THE IDEA

Wouldn't it be cool if at runtime I could see:

- Which objects are instantiated
- What are the values for these objects

THE IDEA

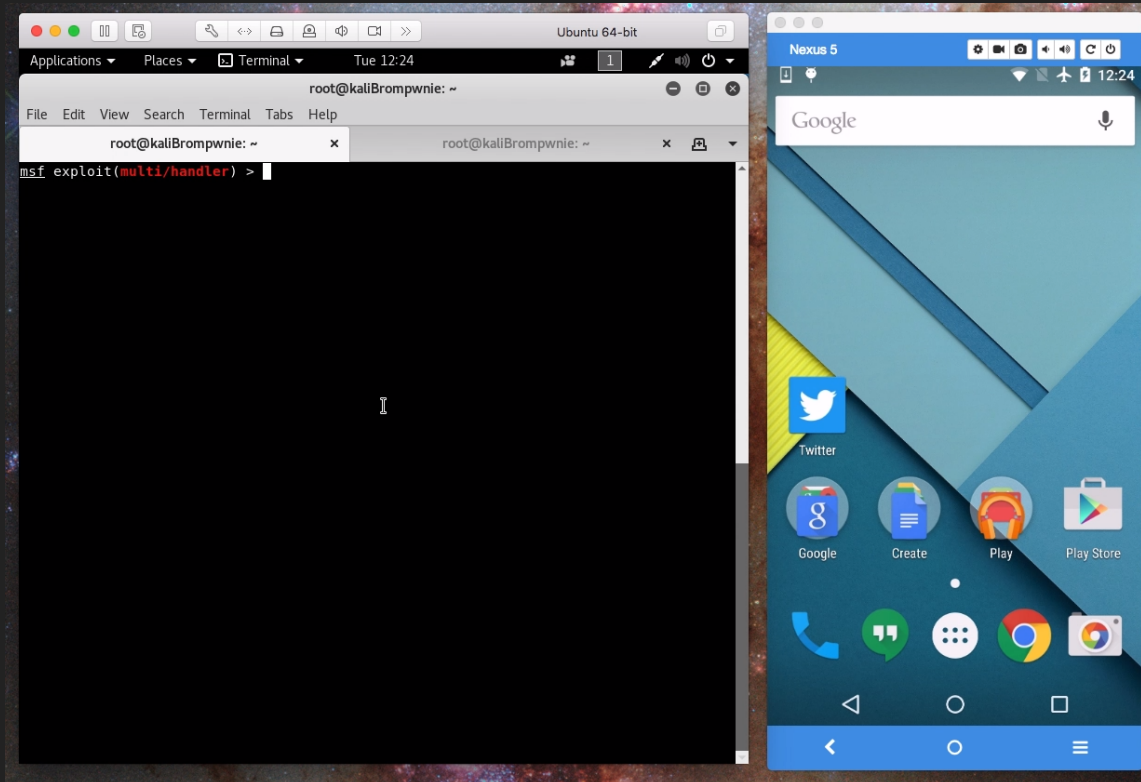
This would give me an idea as to WHAT an app is doing
and HOW

THE IDEA

For example, analysing an app with a meterpreter backdoor:

- Experience tells me to look for:
 - DexClassLoader
 - TCP Connection
- Which tells me that this app is
 - Injecting code at runtime
 - Communicating remotely

DEMO: BASIC MALWARE INFECTION



DEMO: BASIC RUNTIME MALWARE ANALYSIS

The image shows a development environment with two main components: a code editor on the left and a mobile emulator on the right.

Code Editor (Left): The editor displays a JavaScript file named `Demo1_MalwareObjects.js`. The code is designed to monitor various system objects at runtime. It defines an array `objectsToLookFor` containing `"java.net.Socket"`, `"dalvik.system.DexClassLoader"`, `"java.net.URLConnection"`, `"java.net.URL"`, and `"java.security"`. A `for` loop iterates over these objects, and a `Java.perform` function is used to hook the `newInstance` method of each. The hooked functions log detailed information to the console, including the process name, the class being instantiated, and the communication details (protocol, loader, or socket).

```
1 var objectsToLookFor = ["java.net.Socket", "dalvik.system.DexClassLoader", "java.net.URLConnection", "java.net.URL", "java.security"];
2 for (var i in objectsToLookFor) {
3   Java.perform(function () {
4     Java.choose(objectsToLookFor[i], {
5       "onWatch": function (instance) {
6         if (objectsToLookFor[i] === "java.net.URL" && instance.getProtocol() !== "file") {
7           console.log("\n[+] Process has instantiated instance of: " + objectsToLookFor[i]);
8           console.log("[+] Process is communicating via " + instance.getProtocol());
9           console.log("[+] Communication Details: " + instance.toString());
10        }
11        if (objectsToLookFor[i] === "dalvik.system.DexClassLoader") {
12          console.log("\n[+] Process has instantiated instance of: " + objectsToLookFor[i]);
13          console.log("[+] Process is making of DexClassLoader");
14          console.log("[+] Loader Details: " + instance.toString());
15        }
16        if (objectsToLookFor[i] === "java.net.Socket") {
17          console.log("\n[+] Process has instantiated instance of: " + objectsToLookFor[i]);
18          console.log("[+] Process is making of a Socket Connection");
19          console.log("[+] Socket Details: " + instance.toString());
20        }
21        if (objectsToLookFor[i] === "java.net.URLConnection") {
22          console.log("\n[+] Process has instantiated instance of: " + objectsToLookFor[i]);
23          console.log("\n[+] Process is making of a IRI Connection");
24        }
25      }
26    });
27  });
28 }
```

The terminal window at the bottom shows the command used to run the script on an Android emulator:

```
cleroy-ltasScripts cleroy$ frida -U -l Demo1_MalwareObjects.js com.twitter.android
```

Mobile Emulator (Right): The emulator displays a Twitter login screen on a Nexus 5 device. The screen shows the "Log in to Twitter" header, a "Sign up" link, and input fields for "Phone, email or username" (containing "Brompwnie") and "Password" (containing "apassword"). A "Log in" button is visible at the bottom right of the form.

HOW DOES IT WORK WITH OTHER APPS?

```
[+] Process has Instantiated instance of: java.net.Socket  
[*] Process is making of a Socket Connection  
[+] Socket Details: Socket[unconnected]
```

```
[+] Process has Instantiated instance of: java.net.Socket  
[*] Process is making of a Socket Connection  
[+] Socket Details: Socket[unconnected]
```

```
[+] Process has Instantiated instance of: java.net.Socket  
[*] Process is making of a Socket Connection  
[+] Socket Details: Socket[address=/216.58.210.42,port=443,localPort=35472]
```

```
[+] Process has Instantiated instance of: java.net.Socket  
[*] Process is making of a Socket Connection  
[+] Socket Details: Socket[address=/216.58.204.78,port=443,localPort=54449]
```

```
[+] Process has Instantiated instance of: java.net.Socket  
[*] Process is making of a Socket Connection  
[+] Socket Details: Socket[unconnected]
```

```
[+] Process has Instantiated instance of: java.net.Socket  
[*] Process is making of a Socket Connection  
[+] Socket Details: Socket[unconnected]
```

```
[+] Process has Instantiated instance of: java.net.Socket  
[*] Process is making of a Socket Connection  
[+] Socket Details: Socket[unconnected]
```

```
[+] Process has Instantiated instance of: java.net.Socket  
[*] Process is making of a Socket Connection  
[+] Socket Details: Socket[address=/172.217.23.14,port=443,localPort=36214]
```

```
[+] Process has Instantiated instance of: java.net.Socket  
[*] Process is making of a Socket Connection  
[+] Socket Details: Socket[address=/216.58.206.78,port=443,localPort=58169]
```

```
[+] Process has Instantiated instance of: java.net.Socket  
[*] Process is making of a Socket Connection  
[+] Socket Details: Socket[unconnected]
```

```
[+] Process has Instantiated instance of: dalvik.system.DexClassLoader  
[*] Process is making of DexClassLoader  
[+] Loader Details: dalvik.system.DexClassLoader[DexPathList[[zip file "/data/data/com.google.android.youtube/cache/ads877282444.jar"],nativeLibraryDirectories=[/vendor/lib, /system/lib]]]  
[LG Nexus 5 - com.google.android.youtube] <
```

STATIC VS RUNTIME

What static analysis won't show

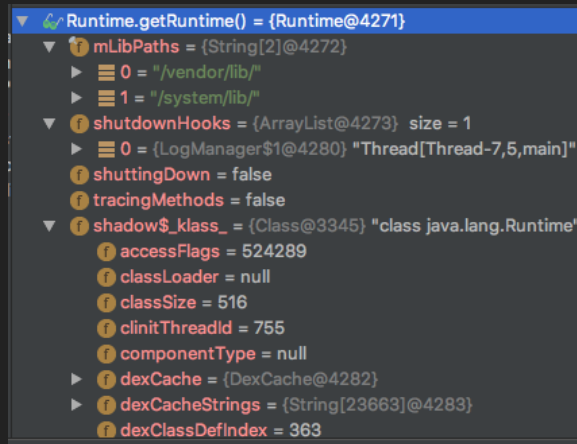
- Runtime Injection
 - Class Loaders
 - What if you don't have the injected JAR/APK
 - `/data/data/com.app.sandbox`
- Java.Lang
 - `Runtime.exec("/bin/sh")`
 - No Import Statements
 - Instantiated but kinda immutable

HEAPS OF LOVE

- Don't have to trawl code
- Identify specific anomalies

HEAPS OF FRUSTRATION

- java.lang.Runtime
- Kind of immutable?
- exec("/system/bin/ps")
 - Does not have much of a footprint



```
Runtime.getRuntime() = {Runtime@4271}
└─ mLibPaths = {String[2]@4272}
   └─ 0 = "/vendor/lib/"
   └─ 1 = "/system/lib/"
└─ shutdownHooks = {ArrayList@4273} size = 1
   └─ 0 = {LogManager$1@4280} "Thread[Thread-7,5,main]"
      └─ shuttingDown = false
      └─ tracingMethods = false
└─ shadow$_klass_ = {Class@3345} "class java.lang.Runtime"
   └─ accessFlags = 524289
   └─ classLoader = null
   └─ classSize = 516
   └─ clinitThreadId = 755
   └─ componentType = null
   └─ dexCache = {DexCache@4282}
   └─ dexCacheStrings = {String[23663]@4283}
   └─ dexClassDefIndex = 363
```



I love it when a plan comes together!

HEAPS OF FRUSTRATION

Whats the plan?

- What is fundamental to objects?
- We can hook methods
 - i.e `Runtime.getRuntime().exec("/bin/sh")`
- How?
 - Overload with Frida
- Not just looking at object state
 - Object behaviour

HEAPS OF FRUSTRATION

Whats the plan?

```
exec(String command)  
Executes the specified string command in a separate process.  
exec(String[] cmdarray)  
Executes the specified command and arguments in a separate process.  
exec(String[] cmdarray, String[] envp)  
Executes the specified command and arguments in a separate process with the specified environment.  
exec(String[] cmdarray, String[] envp, File dir)  
Executes the specified command and arguments in a separate process with the specified environment and working directory.  
exec(String command, String[] envp)  
Executes the specified string command in a separate process with the specified environment.  
exec(String command, String[] envp, File dir)  
Executes the specified string command in a separate process with the specified environment and working directory.
```

```
Java.perform(function () {  
    var targetClass = Java.use("java.lang.Runtime");  
  
    targetClass.exec.overload('java.lang.String').implementation = function (x) {  
        console.log("[*] exec() got called!: "+x);  
        return this.exec(x);  
    };  
  
    targetClass.exec.overload('[Ljava.lang.String;').implementation = function (x) {  
        console.log("[*] exec() got called!: "+x);  
        return this.exec(x);  
    };  
  
    targetClass.exec.overload('java.lang.String', '[Ljava.lang.String;').implementation = function (x,y) {  
        console.log("[*] exec() got called X= "+x);  
        console.log("[*] exec() got called Y= "+y);  
        return this.exec(x,y);  
    };  
});
```

DEMO: OVERLOAD 'DEM METHODS

```
public static void runShellCommand() {
    try {
        System.out.println("RUNTIME TO STRING: " + Runtime.getRuntime().toString());
        Process process = Runtime.getRuntime().exec("command: /system/bin/ps");
        InputStreamReader reader = new InputStreamReader(process.getInputStream());
        BufferedReader bufferedReader = new BufferedReader(reader);
        int numRead;
        char[] buffer = new char[5000];
        StringBuffer commandOutput = new StringBuffer();
        while ((numRead = bufferedReader.read(buffer)) > 0) {
            commandOutput.append(buffer, 0, numRead);
        }
        bufferedReader.close();
        process.waitFor();
        System.out.println("EXECUTING SHELL COMMAND:\n" + commandOutput.toString());
    } catch (IOException e) {}
}

public void loadDexClasses() throws NoSuchFieldException, ClassNotFoundException, IllegalAccessException, NoSuchMethodException, InvocationTargetException {
    File[] files = new File(getAssets().getDataDir()).listFiles();
    Log.v(tag, "loadDexClasses", "msg: \"Preparing to loadDexClasses!\");

    for (File file : files) {
        File tmpDir = new File(String.valueOf(this.getFileDir()));
        DexClassLoader classLoader = new DexClassLoader(file.getAbsolutePath(), tmpDir.getAbsolutePath(), librarySearchPath, classLoader.getSystemClassLoader());
        Log.v(tag, "loadDexClasses", "msg: \"ATTEMPTING TO LOAD CLASSES!\");
        ClassToLoad = (ClassToLoad) classLoader.loadClass("runtime.brompwnie.com.theRuntimeApp.LoadMe");
        Method makeBeard = classToLoad.getMethod("makeBeard");
        makeBeard.invoke(@ null);
        Method runShellCommand = classToLoad.getMethod("runShellCommand");
        runShellCommand.invoke(@ null);
        Method urlConnection = classToLoad.getMethod("urlConnection");
        urlConnection.invoke(@ null);
        Log.v(tag, "loadDexClasses", "msg: \"We are done running new functionality!\");
    }

    public void runStuff(View view) throws IOException, InterruptedException {
        try {
            loadDexClasses();
        } catch (Exception e) {}
    }
}
```

| Time | pid | tid | name | args | cpu | mem | other |
|--------------------|------|------|--------------------------------|--------------------|-----|-----|-------|
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 95 | 2 | 0 |
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 94 | 2 | 0 |
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 95 | 2 | 0 |
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 96 | 2 | 0 |
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 97 | 2 | 0 |
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 98 | 2 | 0 |
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 99 | 2 | 0 |
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 100 | 2 | 0 |
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 101 | 2 | 0 |
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 102 | 2 | 0 |
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 104 | 2 | 0 |
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 105 | 2 | 0 |
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 106 | 2 | 0 |
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 107 | 2 | 0 |
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 108 | 2 | 0 |
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 109 | 2 | 0 |
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 110 | 2 | 0 |
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 111 | 2 | 0 |
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 112 | 2 | 0 |
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 113 | 2 | 0 |
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 114 | 2 | 0 |
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 115 | 2 | 0 |
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 116 | 2 | 0 |
| 03-05 16:55:44.268 | 9261 | 9261 | victim.brompwnie.com.victimapp | I/System.out: root | 117 | 2 | 0 |

```
Java.perform(function () {
    var targetClass = Java.use("java.lang.Runtime");

    targetClass.exec.overload("java.lang.String").implementation = function (x) {
        console.log("[*] exec() got called: " + x);
        return this.exec(x);
    };

    targetClass.exec.overload("[Ljava.lang.String;").implementation = function (x) {
        console.log("[*] exec() got called: " + x);
        return this.exec(x);
    };

    targetClass.exec.overload("java.lang.String", "[Ljava.lang.String;").implementation = function (x, y) {
        console.log("[*] exec() got called X= " + x);
        console.log("[*] exec() got called Y= " + y);
        return this.exec(x, y);
    };

    targetClass.exec.overload("[Ljava.lang.String;", "[Ljava.lang.String;").implementation = function (x, y) {
        console.log("[*] exec() got called X= " + x);
        console.log("[*] exec() got called Y= " + y);
        console.log("[*] exec() got called Z= " + z);
        return this.exec(x, y);
    };

    targetClass.exec.overload("[Ljava.lang.String; ", "[Ljava.lang.String; ", "java.io.File").implementation = function (x, y, z) {
        console.log("[*] exec() got called X= " + x);
        console.log("[*] exec() got called Y= " + y);
        console.log("[*] exec() got called Z= " + z);
        return this.exec(x, y, z);
    };

    targetClass.exec.overload("java.lang.String", "[Ljava.lang.String; ", "java.io.File").implementation = function (x, y, z) {
        console.log("[*] exec() got called X= " + x);
        console.log("[*] exec() got called Y= " + y);
        console.log("[*] exec() got called Z= " + z);
        return this.exec(x, y, z);
    };
});
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
cleroy-ltm:Scripts cleroy$ frida -U -l Demo3_HuntingJavaDotLang.js victim.brompwnie.com.victimapp
```

SNAPSHOT

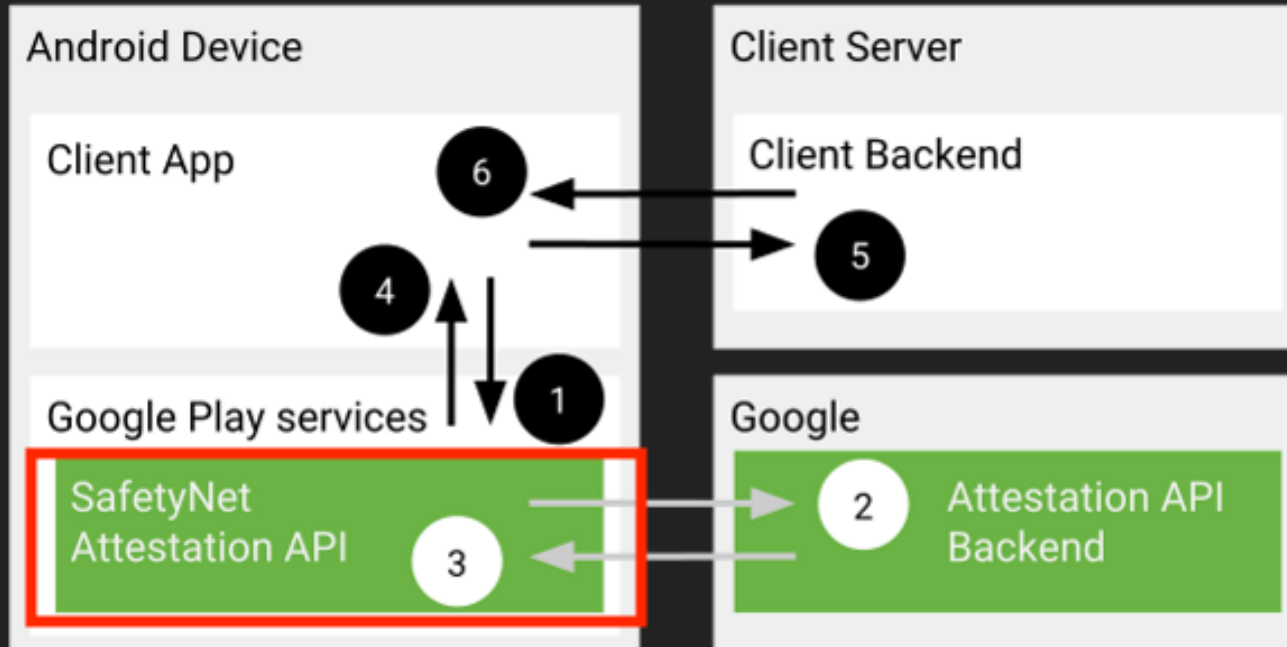
We have the ability to:

- Analyse objects on the HEAP
- Hook methods for certain objects
- Perform this at runtime on a device
- See more than static analysis
- Perform the above from a workstation

A SOLUTION: SAFETYNET ATTESTATION API

The SafetyNet Attestation API helps you assess the security and compatibility of the Android environments in which your apps run. You can use this API to analyze devices that have installed your app.

A SOLUTION: SAFETYNET ATTESTATION API



A SOLUTION: UITKYK

- You can use this API to analyze applications that are installed on a Android device
- Custom Android Frida Library
 - DBUS over TCP
- Frida Server Integration
- Can run all the previously demo'd tests
 - And more!

A SOLUTION: UITKYK API

Hey Frida, give me running processes

```
$ frida-ps -U
```

| PID | Name |
|-------|------------------------------------|
| 207 | adbd |
| 27599 | android.process.acore |
| 1566 | android.process.media |
| 16850 | app_process |
| 16894 | app_process |
| 17005 | app_process |
| 17145 | app_process |
| 17163 | app_process |
| 17190 | app_process |
| 17327 | app_process |
| 17384 | app_process |
| 17430 | app_process |
| 26767 | app_process32 |
| 194 | bridgemgrd |
| 27556 | com.android.defcontainer |
| 27799 | com.android.keychain |
| 27622 | com.android.musicfx |
| 1799 | com.android.nfc |
| 1855 | com.android.phone |
| 1544 | com.android.systemui |
| 27643 | com.android.vending |
| 27882 | com.google.android.apps.cloudprint |

A SOLUTION: UITKYK API

Hey Android, give me running processes

```
@Override
protected String doInBackground(String... params) {
    UitkykUtils uitkykUtils= new UitkykUtils();
    return uitkykUtils.fridaPS(fridaHost,fridaPort);
}
```


A SOLUTION: UITKYK API

Hey Android, tell me if this application looks malicious

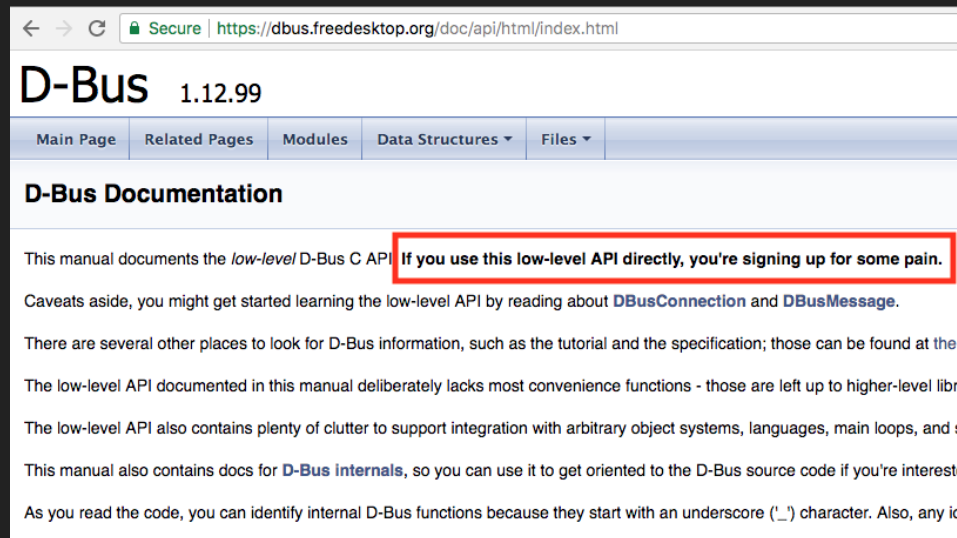
```
protected String doInBackground(Object... objects) {  
    UitkykUtils uitkykUtils = new UitkykUtils(fridaHost, fridaPort);  
    return uitkykUtils.analyzeProcess(this.pid);  
}
```

A SOLUTION: UITKYK API



WHY UITKYK API?

- No Android Frida Library
- Wanted to use Frida
- Wanted a Client Server Model
- Didn't want pain



← → ↻ Secure | <https://dbus.freedesktop.org/doc/api/html/index.html>

D-Bus 1.12.99

Main Page | Related Pages | Modules | Data Structures ▾ | Files ▾

D-Bus Documentation

This manual documents the *low-level* D-Bus C API. **If you use this low-level API directly, you're signing up for some pain.**

Caveats aside, you might get started learning the low-level API by reading about [DBusConnection](#) and [DBusMessage](#).

There are several other places to look for D-Bus information, such as the tutorial and the specification; those can be found at the [D-Bus website](#).

The low-level API documented in this manual deliberately lacks most convenience functions - those are left up to higher-level libraries.

The low-level API also contains plenty of clutter to support integration with arbitrary object systems, languages, main loops, and so on.

This manual also contains docs for [D-Bus Internals](#), so you can use it to get oriented to the D-Bus source code if you're interested.

As you read the code, you can identify internal D-Bus functions because they start with an underscore ('_') character. Also, any ide

HOW DOES UITKYK, UITKYK?

- TCP Socket to Daemon
- Push and Pull Bytes
 - Sniffed Frida sessions
 - Outlined TCP Flags
 - Identified key bytes (trial and error)
 - Stared at my monitor
- Wash,rinse,repeat

```
byte[] AUTH_Message2 = {  
    (byte) 0x41, (byte) 0x55, (byte) 0x54, (byte) 0x48,  
    (byte) 0x20, (byte) 0x41, (byte) 0x4e, (byte) 0x4f,  
    (byte) 0x4e, (byte) 0x59, (byte) 0x4d, (byte) 0x4f,  
    (byte) 0x55, (byte) 0x53, (byte) 0x20, (byte) 0x34,  
    (byte) 0x37, (byte) 0x34, (byte) 0x34, (byte) 0x34,  
    (byte) 0x32, (byte) 0x37, (byte) 0x35, (byte) 0x37,  
    (byte) 0x33, (byte) 0x32, (byte) 0x30, (byte) 0x33,  
    (byte) 0x30, (byte) 0x32, (byte) 0x65, (byte) 0x33,  
    (byte) 0x31, (byte) 0x0d, (byte) 0x0a};
```


ORIGINAL PYTHON POC

```
import socket

TCP_IP = '10.42.0.15'
TCP_PORT = 1337
BUFFER_SIZE = 100
# .AUTH
MESSAGE = '\x2e\x41\x55\x54\x48\x0d\x0a'
# MESSAGE = '\x41\x55\x54\x48\x0d\x0a'

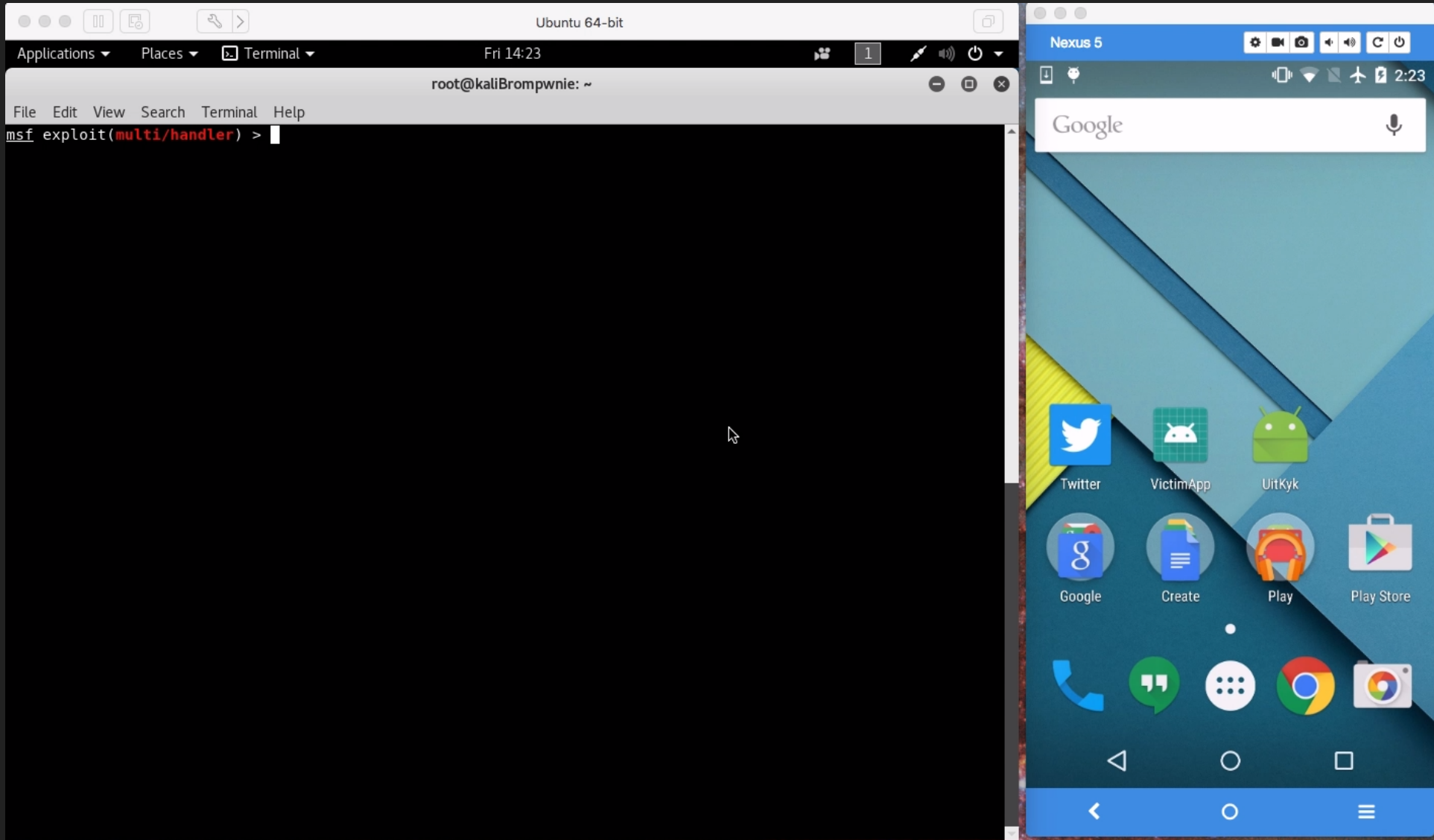
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print "Sending-> .AUTH"
s.connect((TCP_IP, TCP_PORT))
s.send(MESSAGE)
data = s.recv(BUFFER_SIZE)
print "received data:", data
# s.close()

# AUTH ANONYMOUS 474442757320302e31
MESSAGE2="\x41\x55\x54\x48\x20\x41\x4e\x4f\x4e\x59\x4d\x4f\x55\x53\x20\x34\x37\x34\x34\x34\x32\x37\x35\x37\x33\x32\x30\x33\x30\x32\
BUFFER_SIZE2 = 100
# s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print "Sending-> AUTH ANONYMOUS 474442757320302e31"
# s.connect((TCP_IP, TCP_PORT))
s.send(MESSAGE2)
data2 = s.recv(BUFFER_SIZE2)
print "received data:", data2
# s.close()

# #
# #BEGIN
MESSAGE3="\x42\x45\x47\x49\x4e\x0d\x0a"
BUFFER_SIZE3 = 100
# s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print "Sending-> BEGIN"
# s.connect((TCP_IP, TCP_PORT))
s.send(MESSAGE3)
# data3 = s.recv(BUFFER_SIZE3)
# print "received data:", data3
# # s.close()

# 6c 01 00 01 1b 00 00 01 00 00 00 60 00 00 00 08 01 67 00 01 73 00 00 01 01 6f 00 15 00 00 2f 72 65 2f 66 72 69 64 61 2f 48 {
# HostSession....s....GetAll...s....org.freedesktop.DBus.Properties....re.frida.HostSession10.
MESSAGE4="\x6c\x01\x00\x01\x1b\x00\x00\x00\x01\x00\x00\x00\x60\x00\x00\x00\x08\x01\x67\x00\x01\x73\x00\x00\x01\x01\x6f\x00\x15\x00\
BUFFER_SIZE4 = 100
print "Sending-> HostSession"
s.send(MESSAGE4)
data4 = s.recv(BUFFER_SIZE4)
print "received data:", data4
```

DEMO: UITKYK



UITKYK

- Library
 - github.com/brompwnie/uitkyk
- Blogpost/s
 - brompwnie.github.io
- Frida Scripts
 - github.com/brompwnie/uitkyk
- Videos
 - <https://goo.gl/k6BNBq>

SHORTCOMINGS

- Increased Attack Surface
- Abuse(duh)
- We are still struggling to get basic security right



CONCLUSION

- It's a journey
- Uitkyk is a step in the right direction
- No Silver Bullet
- Defense In Depth
- Android OS is key to protecting itself



QUESTIONS?

DANKE!