Wiki »

Frida Beginner tutorials

Installation

Ubuntu 16.04

1. Download frida using

```
pip install frida
```

2. Check version by using

```
frida --version
```

3. Go to frida github and download the same version of server for android and correct architecture. (Currently x86 server on x86 image is crashing the app)

1. Push frida-server to android device, chmod 775 it and execute it with

```
 adb shell "/path/to/frida-server -D"
```

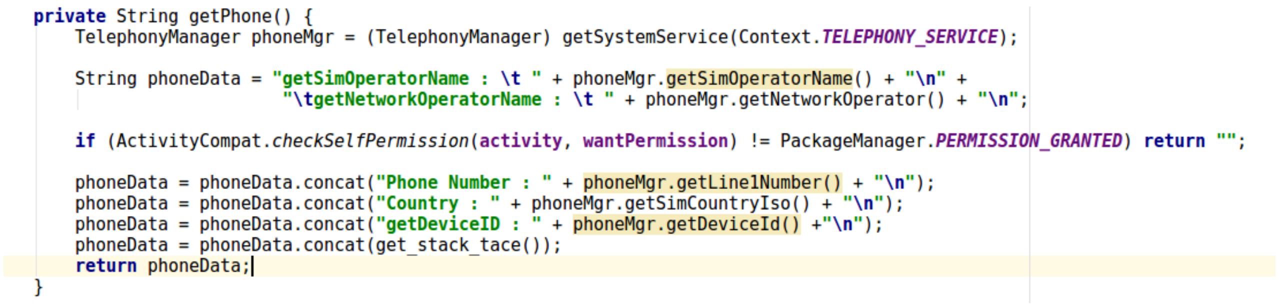
1. On host test frida with

```
frida-ps -U
```

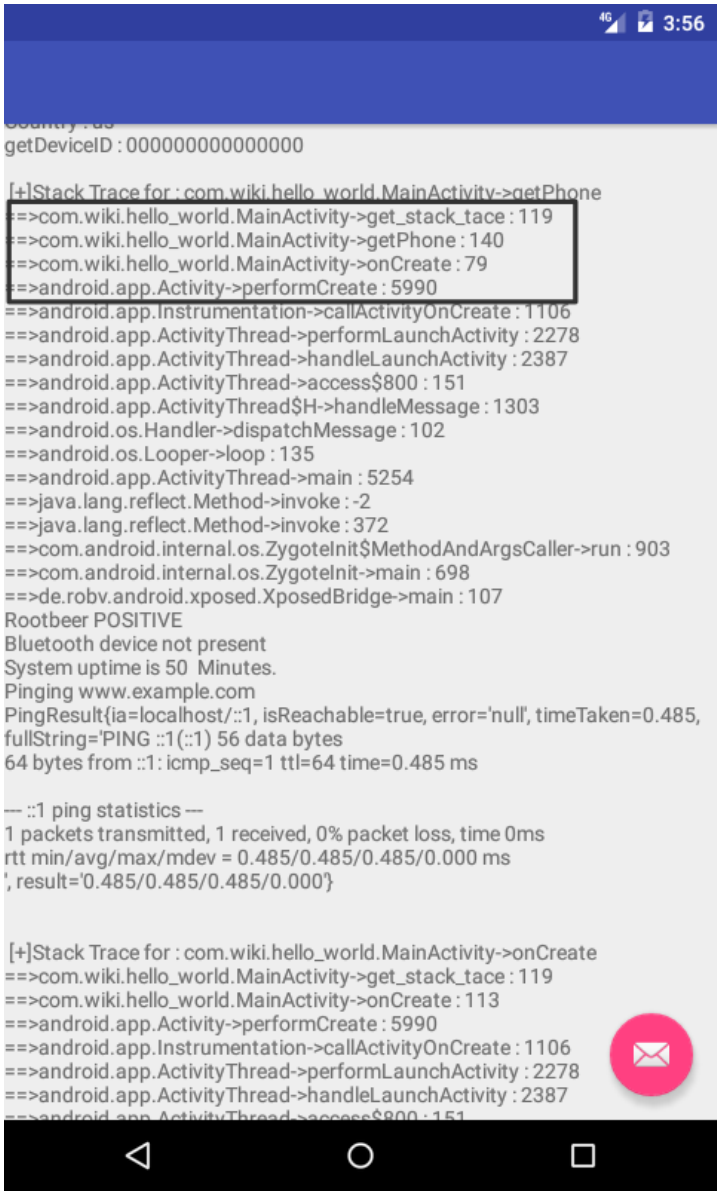
Example 1: Hooking a method

In this example we will run one of our own developed application. This is application is for testing the anti-emulator capabilities of our device. It obtains different information from the system which can be used to detect emulation environment or root. It also displays stack trace of some its methods and that information can be used to detect hooking. This is still underdevelopment.

In this example we will use Frida to hook the getPhone( ) method inside MainActivity. Below is the stack trace and source code of this method.



```
private String getPhone() {
    TelephonyManager phoneMgr = (TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE);

    String phoneData = "getSimOperatorName : \t " + phoneMgr.getSimOperatorName() + "\n" +
                       "\tgetNetworkOperatorName : \t " + phoneMgr.getNetworkOperator() + "\n";

    if (ActivityCompat.checkSelfPermission(activity, wantPermission) != PackageManager.PERMISSION_GRANTED) return "";

    phoneData = phoneData.concat("Phone Number : " + phoneMgr.getLine1Number() + "\n");
    phoneData = phoneData.concat("Country : " + phoneMgr.getSimCountryIso() + "\n");
    phoneData = phoneData.concat("getDeviceID : " + phoneMgr.getDeviceId() +"\n");
    phoneData = phoneData.concat(get_stack_tace());
    return phoneData;
}
```



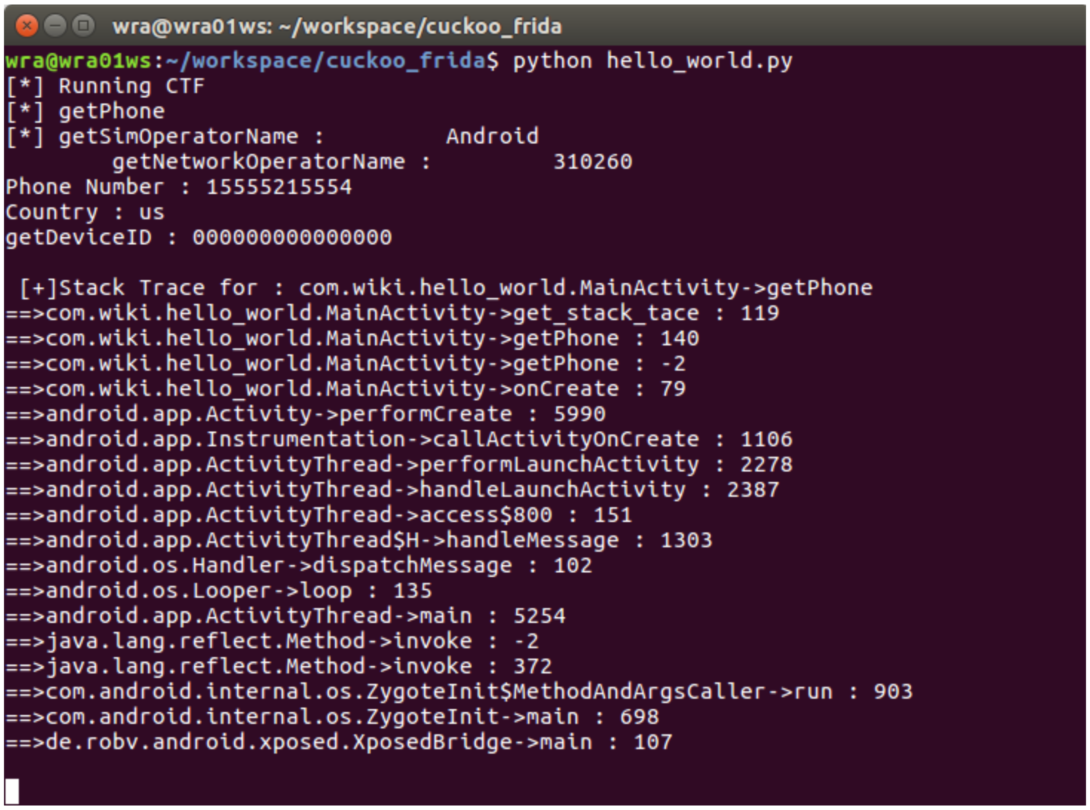
```
getDeviceID : 000000000000000

[+]Stack Trace for : com.wiki.hello_world.MainActivity->getPhone
==>com.wiki.hello_world.MainActivity->get_stack_tace : 119
==>com.wiki.hello_world.MainActivity->getPhone : 140
==>com.wiki.hello_world.MainActivity->onCreate : 79
==>android.app.Activity->performCreate : 5990
==>android.app.Instrumentation->callActivityOnCreate : 1106
==>android.app.ActivityThread->performLaunchActivity : 2278
==>android.app.ActivityThread->handleLaunchActivity : 2387
==>android.app.ActivityThread->access$800 : 151
==>android.app.ActivityThread$H->handleMessage : 1303
==>android.os.Handler->dispatchMessage : 102
==>android.os.Looper->loop : 135
==>android.app.ActivityThread->main : 5254
==>java.lang.reflect.Method->invoke : -2
==>java.lang.reflect.Method->invoke : 372
==>com.android.internal.os.ZygoteInit$MethodAndArgsCaller->run : 903
==>com.android.internal.os.ZygoteInit->main : 698
==>de.robv.android.xposed.XposedBridge->main : 107
Rootbeer POSITIVE
Bluetooth device not present
System uptime is 50  Minutes.
Pinging www.example.com
PingResult{ia=localhost/::1, isReachable=true, error='null', timeTaken=0.485,
fullString='PING ::1(::1) 56 data bytes
64 bytes from ::1: icmp_seq=1 ttl=64 time=0.485 ms

--- ::1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.485/0.485/0.485/0.000 ms
', result='0.485/0.485/0.485/0.000'}


[+]Stack Trace for : com.wiki.hello_world.MainActivity->onCreate
==>com.wiki.hello_world.MainActivity->get_stack_tace : 119
==>com.wiki.hello_world.MainActivity->onCreate : 113
==>android.app.Activity->performCreate : 5990
==>android.app.Instrumentation->callActivityOnCreate : 1106
==>android.app.ActivityThread->performLaunchActivity : 2278
==>android.app.ActivityThread->handleLaunchActivity : 2387
```

In order to hook this method follow these steps:

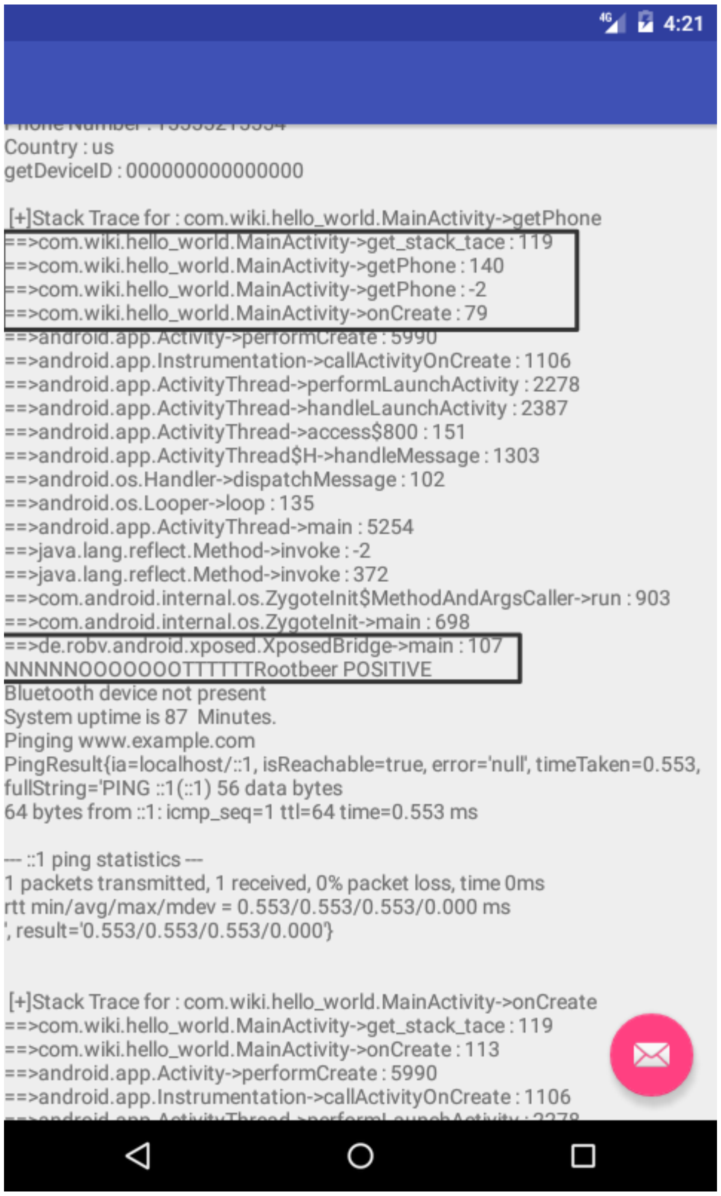
1. Execute frida-server on guest, if it not running already.
2. copy below python script:

```
import frida, sys, time

def on_message(message, data):
    if message['type'] == 'send':
        print("[*] {0}".format(message['payload']))
    else:
        print(message)

jscode = """
Java.perform(function () {
    // Function to hook is defined here
    var MainActivity = Java.use('com.wiki.hello_world.MainActivity');

    // Whenever button is clicked
    MainActivity.getPhone.implementation = function () {
        // Show a message to know that the function got called
        send('getPhone');

        // Call the original onClick handler
        var output = this.getPhone();
        send(output);
        return output+"NNNOOOOTTTT"; // Modify return value of getPhone method
    };
});
"""

device = frida.get_usb_device()
pid = device.spawn(['com.wiki.hello_world'])
process = device.attach(pid)
script = process.create_script(jscode)
script.on('message', on_message)
print('[*] Running CTF')
script.load()
device.resume(pid)
sys.stdin.read()
```

1. The above Python script is doing following stuff:
    1. (Start reading from line 35) It connects with adb,
    2. Opens the application process (its paused and need to be resumed)
    3. Attach it to process, adds the JavaScript code store in jscode variable
    4. Registers the callback function to receive messages from guest (send using send("message")
    5. Load the script and resumes the start executing the application opened in 2.
    6. Add stdin to stop application from exiting
2. In JavaScript code we are doing following:
    1. We open MainActivity using Java.use
    2. We hook the getPhone method, send its original return value and modify the return value. Below images who the original return value displayed on terminal, and modified returned value in app window. Also notice the stack trace and compare it with the previous one in which no hooking was involved.



```
wra@wra01ws:~/workspace/cuckoo_frida$ python hello_world.py
[*] Running CTF
[*] getPhone
[*] getSimOperatorName :         Android
        getNetworkOperatorName :         310260
Phone Number : 15555215554
Country : us
getDeviceID : 000000000000000

 [+]Stack Trace for : com.wiki.hello_world.MainActivity->getPhone
==>com.wiki.hello_world.MainActivity->get_stack_tace : 119
==>com.wiki.hello_world.MainActivity->getPhone : 140
==>com.wiki.hello_world.MainActivity->getPhone : -2
==>com.wiki.hello_world.MainActivity->onCreate : 79
==>android.app.Activity->performCreate : 5990
==>android.app.Instrumentation->callActivityOnCreate : 1106
==>android.app.ActivityThread->performLaunchActivity : 2278
==>android.app.ActivityThread->handleLaunchActivity : 2387
==>android.app.ActivityThread->access$800 : 151
==>android.app.ActivityThread$H->handleMessage : 1303
==>android.os.Handler->dispatchMessage : 102
==>android.os.Looper->loop : 135
==>android.app.ActivityThread->main : 5254
==>java.lang.reflect.Method->invoke : -2
==>java.lang.reflect.Method->invoke : 372
==>com.android.internal.os.ZygoteInit$MethodAndArgsCaller->run : 903
==>com.android.internal.os.ZygoteInit->main : 698
==>de.robv.android.xposed.XposedBridge->main : 107
```



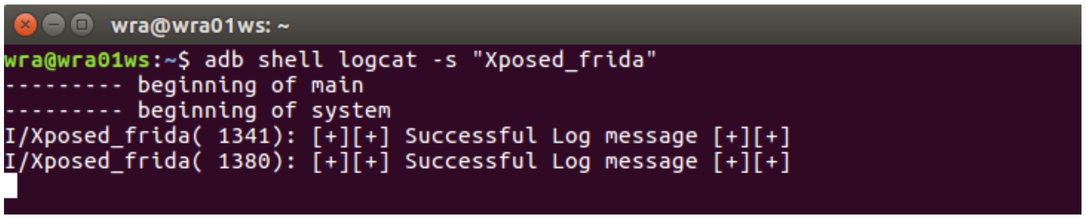
```
Country : us
getDeviceID : 000000000000000

 [+]Stack Trace for : com.wiki.hello_world.MainActivity->getPhone
==>com.wiki.hello_world.MainActivity->get_stack_tace : 119
==>com.wiki.hello_world.MainActivity->getPhone : 140
==>com.wiki.hello_world.MainActivity->getPhone : -2
==>com.wiki.hello_world.MainActivity->onCreate : 79
==>android.app.Activity->performCreate : 5990
==>android.app.Instrumentation->callActivityOnCreate : 1106
==>android.app.ActivityThread->performLaunchActivity : 2278
==>android.app.ActivityThread->handleLaunchActivity : 2387
==>android.app.ActivityThread->access$800 : 151
==>android.app.ActivityThread$H->handleMessage : 1303
==>android.os.Handler->dispatchMessage : 102
==>android.os.Looper->loop : 135
==>android.app.ActivityThread->main : 5254
==>java.lang.reflect.Method->invoke : -2
==>java.lang.reflect.Method->invoke : 372
==>com.android.internal.os.ZygoteInit$MethodAndArgsCaller->run : 903
==>com.android.internal.os.ZygoteInit->main : 698
==>de.robv.android.xposed.XposedBridge->main : 107
NNNNNOOOOOOOTTTTTTRootbeer POSITIVE
Bluetooth device not present
System uptime is 87  Minutes.
Pinging www.example.com
PingResult{ia=localhost/::1, isReachable=true, error='null', timeTaken=0.553,
fullString='PING ::1(::1) 56 data bytes
64 bytes from ::1: icmp_seq=1 ttl=64 time=0.553 ms

--- ::1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.553/0.553/0.553/0.000 ms
', result='0.553/0.553/0.553/0.000'}


 [+]Stack Trace for : com.wiki.hello_world.MainActivity->onCreate
==>com.wiki.hello_world.MainActivity->get_stack_tace : 119
==>com.wiki.hello_world.MainActivity->onCreate : 113
==>android.app.Activity->performCreate : 5990
==>android.app.Instrumentation->callActivityOnCreate : 1106
```

Example 2: Getting Javascript wrapper for Java classes (Logcat example):

Python code for this example is below. In this code it can be seen that we use Java.use method provided by JavaScript API of frida to get a JavaScript wrapper for class 'android.util.Log'. We then check for log.i method and if it is present, we call it.

```
import frida, sys, time

def on_message(message, data):
    if message['type'] == 'send':
        print("[*] {0}".format(message['payload']))
    else:
        print(message)

jscode = """
Java.perform(function () {
    // Function to hook is defined here
    var MainActivity = Java.use('com.wiki.hello_world.MainActivity');
    var Log = Java.use('android.util.Log');
    if (Log.i){
        send("Sending message");
        Log.i("Xposed_frida", "[+][+] Successful Log message [+][+]");
    }
});
"""

device = frida.get_usb_device()
pid = device.spawn(['com.wiki.hello_world'])
process = device.attach(pid)
script = process.create_script(jscode)
script.on('message', on_message)
print('[*] Running CTF')
script.load()
device.resume(pid)

sys.stdin.read()
```

Now when you run this script and the result can be seen below:



```
wra@wra01ws:~$ adb shell logcat -s "Xposed_frida"
--------- beginning of main
--------- beginning of system
I/Xposed_frida( 1341): [+][+] Successful Log message [+][+]
I/Xposed_frida( 1380): [+][+] Successful Log message [+][+]
```

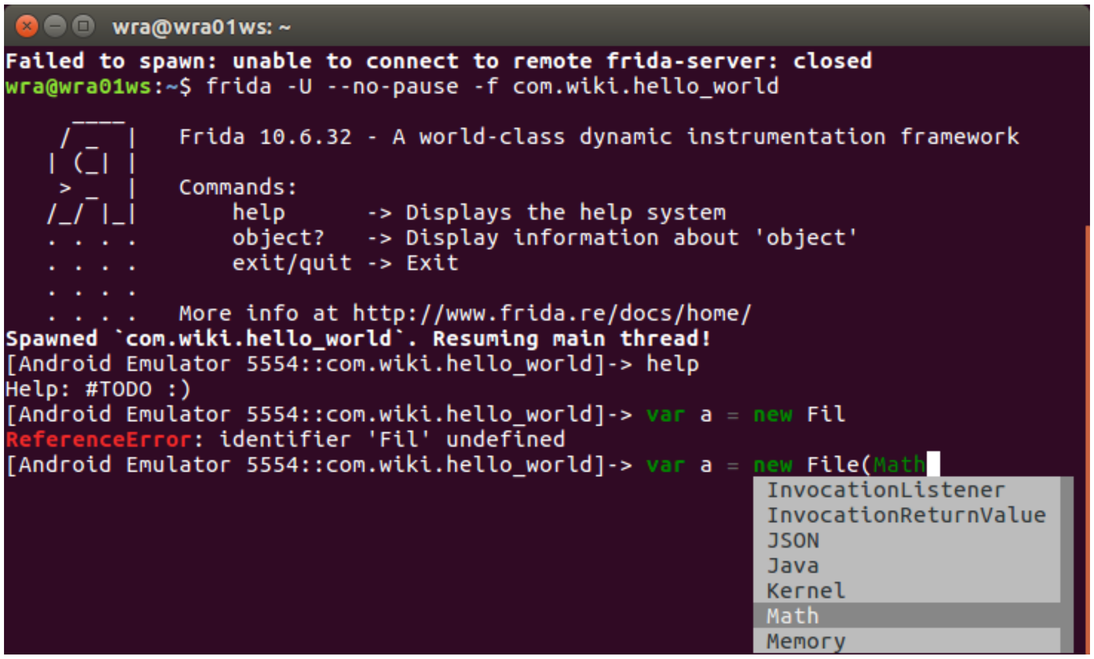
This example demonstrate the power of frida, we can use all the Java functionality inside our script.

Example 3: Interactive Frida (Frida-CLI):

One can get an interactive frida shell by entering the following command:

```
frida -U --no-pause -f com.package.name
```

This command will launch the app and gives an interactive shell where you can type Javascript to do your stuff. If you want to write your script before the app launches, omit the --no-pause parameter.



```
Failed to spawn: unable to connect to remote frida-server: closed
wra@wra01ws:~$ frida -U --no-pause -f com.wiki.hello_world
     ____
    / _  |   Frida 10.6.32 - A world-class dynamic instrumentation framework
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at http://www.frida.re/docs/home/
Spawned `com.wiki.hello_world`. Resuming main thread!
[Android Emulator 5554::com.wiki.hello_world]-> help
Help: #TODO :)
[Android Emulator 5554::com.wiki.hello_world]-> var a = new Fil
ReferenceError: identifier 'Fil' undefined
[Android Emulator 5554::com.wiki.hello_world]-> var a = new File(Math
                                                      InvocationListener
                                                      InvocationReturnValue
                                                      JSON
                                                      Java
                                                      Kernel
                                                      Math
                                                      Memory
```

Example 4: JSON objects:

In the exploit script we can use JSON class to handle JSON data. In the below example we will send some JSON data(hooks) in the form of string to guest where exploit script will receive it and will convert it to a JSON object. Then we will access different elements of this data using JavaScript. Below are the scripts:

see python script Host

```
import frida, sys, time, os

def on_message(message, data):
    if message['type'] == 'send':
        print("[*] {0}".format(message['payload']))
        if message['payload'] == 'get_hooks':
            with open(os.path.join(os.path.dirname(__file__),"hooks.json")) as hooks:
                script.post(hooks.read())
    else:
        print(message)

device = frida.get_usb_device()
pid = device.spawn(['com.wiki.hello_world'])
device.resume(pid)
time.sleep(1)
process = device.attach(pid)
script = process.create_script(open(os.path.join(os.path.dirname(__file__),"recv_hooks.js"))

script.on('message', on_message)
print('[*] Running CTF')
script.load()
sys.stdin.read()
```

Exploit

```
'use strict';

Java.perform(function () {
    var Log = Java.use('android.util.Log');
    var MainActivity = Java.use('com.wiki.hello_world.MainActivity');
    var received_json = "";

    send("Sending message");
    send("get_hooks");
    recv( function (a){
        //send(a);
        received_json = a;
        }).wait();

    send("Just received string \n");
    //send(received_json);
    var hooks = JSON.parse(received_json);
    send("Total number of methods to be hooked is : ");
    send(hooks.hookConfigs.length);
    for (var method in hooks.hookConfigs){
        send(hooks.hookConfigs[method].class_name + "->" + hooks.hookConfigs[method].method
/*
        if (Java.use(hooks.hookConfigs[method].class_name)[hooks.hookConfigs[method].method]
            for (overload in Java.use(hooks.hookConfigs[method].class_name)[hooks.hookConfig
                send(Java.use(hooks.hookConfigs[method].class_name)[hooks.hookConfigs[method
            }
        }
*/
    }

    var tm = Java.use("android.telephony.TelephonyManager");
    send(tm.getDeviceId.overloads[0]);
});
```

see hooks.json contents

```
{
    "hookConfigs": [
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getDeviceId",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getSubscriberId",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getLine1Number",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getNetworkOperator",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getNetworkOperatorName",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getSimOperatorName",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.net.wifi.WifiInfo",
            "method": "getMacAddress",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getSimCountryIso",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getSimSerialNumber",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getNetworkCountryIso",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getDeviceSoftwareVersion",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.os.Debug",
            "method": "isDebuggerConnected",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.app.SharedPreferencesImpl$EditorImpl",
            "method": "putString",
            "thisObject": false,
            "type": "globals"
        },
        {
            "class_name": "android.app.SharedPreferencesImpl$EditorImpl",
            "method": "putBoolean",
            "thisObject": false,
            "type": "globals"
        },
        {
            "class_name": "android.app.SharedPreferencesImpl$EditorImpl",
            "method": "putInt",
            "thisObject": false,
            "type": "globals"
        },
        {
            "class_name": "android.app.SharedPreferencesImpl$EditorImpl",
            "method": "putLong",
            "thisObject": false,
            "type": "globals"
        },
        {
            "class_name": "android.app.SharedPreferencesImpl$EditorImpl",
            "method": "putFloat",
            "thisObject": false,
            "type": "globals"
        },
        {
            "class_name": "android.content.ContentValues",
            "method": "put",
            "thisObject": false,
            "type": "globals"
        },
        {
            "class_name": "java.net.URL",
            "method": "openConnection",
            "thisObject": true,
            "type": "network"
        },
        {
            "class_name": "org.apache.http.impl.client.AbstractHttpClient",
            "method": "execute",
            "thisObject": false,
            "type": "network"
        }
    ],
    "trace": false
```

```
}
```

The output of these scripts is below (Output in python terminal):

```
/usr/bin/python2.7 /home/wra/workspace/cuckoo_frida/sending_hooks_json/send_hooks.py
[*] Running CTF
[*] Sending message
[*] get_hooks
[*] Just received string

[*] Total number of methods to be hooked is :
[*] 20
[*] android.telephony.TelephonyManager->getDeviceId 2
[*] android.telephony.TelephonyManager->getSubscriberId 2
[*] android.telephony.TelephonyManager->getLine1Number 1
[*] android.telephony.TelephonyManager->getNetworkOperator 1
[*] android.telephony.TelephonyManager->getNetworkOperatorName 2
[*] android.telephony.TelephonyManager->getSimOperatorName 1
[*] android.net.wifi.WifiInfo->getMacAddress 1
[*] android.telephony.TelephonyManager->getSimCountryIso 2
[*] android.telephony.TelephonyManager->getSimSerialNumber 2
[*] android.telephony.TelephonyManager->getNetworkCountryIso 1
[*] android.telephony.TelephonyManager->getDeviceSoftwareVersion 2
[*] android.os.Debug->isDebuggerConnected 1
[*] android.app.SharedPreferencesImpl$EditorImpl->putString 1
[*] android.app.SharedPreferencesImpl$EditorImpl->putBoolean 1
[*] android.app.SharedPreferencesImpl$EditorImpl->putInt 1
[*] android.app.SharedPreferencesImpl$EditorImpl->putLong 1
[*] android.app.SharedPreferencesImpl$EditorImpl->putFloat 1
[*] android.content.ContentValues->put 9
[*] java.net.URL->openConnection 2
[*] org.apache.http.impl.client.AbstractHttpClient->execute 8
```

In the output we see count the number the methods in hooks.json file, the display the class names, method names and number of method overloads.

Example 5: Compile a Frida script comprised of one or more Node.js modules (File I/O test case):

Its not much useful to code in simple JavaScript in our use case scenario. Our use case scenario is same as server side programming using JavaScript and its important to be able to use NodeJS. Frida provides a tool for that called "Frida-Compile". This tool compile a Frida script comprised of one or more Node.js modules. In layman terms, its takes a NodeJS projects and create a single file payload/agent that we can load using the "load_script" method that comes with frida.

First we need to install frida-compile . To be able to use it in command line anywhere, we can install it with "-g" tag to make it available globally. Sudo is required for its global installation.

```
sudo npm install frida-compile -g
```

To verify the installation and see help:

```
frida-compile --help

  Usage: frida-compile [options] <module>

  Options:

    -V, --version             output the version number
    -o, --output <file>       set output <file>
    -w, --watch               watch for changes and recompile
    -b, --bytecode            output bytecode
    -x, --no-babelify         skip Babel transforms
    -S, --no-sourcemap        omit sourcemap
    -c, --compress            compress using UglifyJS2
    -a, --use-absolute-paths  use absolute source paths
    -h, --help                output usage information
```

Now the command to compile a NodeJS project/code is below. Here app.js the NodeJS main file and payload.js is the output file that we will load in python binding using load_script.

```
frida-compile app.js -o payload.js
```

Below are scripts that I used to open a text file located on the device and send its content to host.

Make sure that you install the required packages (frida-fs)using npm install.

JavaScript

```
'use strict';

const fs = require("frida-fs");

Java.perform(function () {

    var readStream = fs.createReadStream("/data/local/tmp/hooks.json");
    var text = "";

    readStream
        .on('readable', function () {
            var chunk;
            while (null !== (chunk = readStream.read())) {
                text = text.concat(chunk);
            }
        })
        .on('end', function () {
            send(text);
        });
});
```

In this python code we first compile the NodeJS code using frida-compile and then use the output script as a normal JavaScript payload shown in every other example.

Python

```
import frida, sys, time, os, subprocess

def on_message(message, data):
    try:
        if message['type'] == 'send':
            print("[*] {0}".format(message['payload']))
        else:
            print(message)
    except:
        print(message)

frida_ready = False
device = frida.get_usb_device()
while not frida_ready:
    try:
        time.sleep(5)
        pid = device.spawn(['com.wiki.hello_world'])
        frida_ready = True
    except Exception as e:
        print("Frida not ready yet " + str(e))

time.sleep(1)
```

```
cwd = os.path.dirname(__file__)
print("Working Directory is : "+ cwd)

print ("Compilation result is: \n")
print(subprocess.Popen("frida-compile index.js -o compiled.js", shell=True).communicate())

try:
    process = device.attach(pid)
    script = process.create_script(
        open("/home/wra/workspace/cuckoodroid/frida-scripts/file_io/compiled.js").read())
    script.on('message', on_message)
    print('[*] Running Frida')
    script.load()
    device.resume(pid) # At end Important, otherwise app will start and you will miss some e
except Exception as e:
    print("Frida python binding code error :" + str(e))

sys.stdin.read(1)
```

Below the output, you can see that the contents of hooks.json file was sent by guest and we displayed it on host.

Output

```
Working Directory is :
Compilation result is:

(None, None)
[*] Running Frida
[*] {
    "hookConfigs": [
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getDeviceId",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getSubscriberId",
            "thisObject": false,
            "type": "fingerprint"
        }
    ],
    "trace": false
}
```

That's the end of this example. I hope it was useful.

Example 6: Reading a file (Synchronously and asynchronously)

By now it would be clear enough that you would be able to compile a nodejs script so forward we will not post the python and terminal part unless necessary.

JavaScript code: Read file Asynchronously

```
'use strict';

const fs = require("frida-fs");
const events = require("events");

Java.perform(function () {
    var writeStream = fs.createWriteStream("/data/local/tmp/text")
    var of = fs.createReadStream("/data/local/tmp/hooks.json").pipe(writeStream);
    send("Reading file");
    var readStream = fs.createReadStream("/data/local/tmp/text")
    var text = "";
    readStream
        .on('readable', function () {
            var chunk;
            while (null !== (chunk = readStream.read())) {
                text = text.concat(chunk);
            }
        })
        .on('end', function () {
            send(text);
        });
});
```

JavaScript code: Read file Synchronously

```
/*
Example demonstrates how to read a file synchronously using frida on target machine.
use frida-compile to compile this code
*/

'use strict';

const fs = require("frida-fs");
Java.perform(function () {
    send("reading file.txt");
    // Some decoding may be required on host to dispaly it in string format.
    send(fs.readFileSync("path/to/file.txt"));
    send("Reading file.txt finished");
});
```

Example 7: Writing to a file

Writing to a file

```
'use strict';

const fs = require("frida-fs");

Java.perform(function () {

    var writeStream = fs.createWriteStream("/data/local/tmp/text");
    send("Writing to file");
    var text = "Hello to frida\n";

    /*
    writeStream
        .on('error', function (err) {
            send("Error detected");
            send(err);
        })
        .on('finish', function () {
            send("Finish event detected");
            writeStream.end();
        });
    */
    writeStream.write(text+text);

});
```

Make sure that the file exists as currently we are only able write to application memory which is located at /data/data/package.name/files/

directory. Below is the code snippet to make a file there and then read it again.

```
'use strict';

const fs = require("frida-fs");

Java.perform(function () {

    var file = new File("/data/data/org.wikipedia/files/file_new.txt","w");
    file.write("Hi Hola Hello");
    file.close();

    send("sending file contents");
    send(JSON.stringify(fs.readFileSync("/data/data/org.wikipedia/files/file_new.txt")));

});
```

Example 8: Hooking methods from json string:

Javascript

```
'use strict';

Java.perform(function () {
    //var classes_dict = {}; // Dictionary for keeping track of classes that we already used
    var Log = Java.use('android.util.Log');
    var MainActivity = Java.use('com.wiki.hello_world.MainActivity');
    var received_json = "";

    send("Sending message");
    send("get_hooks");
    recv( function (a){
        received_json = a;
        }).wait();

    send("Just received string \n");
    //send(received_json);
    var hooks = JSON.parse(received_json);
    var overloads;
    send("Total number of methods to be hooked is : ");
    send(hooks.hookConfigs.length);
    hookMethods(hooks);

    /*
        MY CUSTOM FUNCTIONS
        TODO: FIND A WAY TO MOVE THEM TO ANOTHER FILE
    */

    function hookMethods(hooks_json){
        for (var index in hooks_json.hookConfigs){
            send("Hooking :" + hooks.hookConfigs[index].class_name + "->" + hooks.hookConfig
            hookMethod(hooks.hookConfigs[index].class_name, hooks.hookConfigs[index].method)
        }
    }

    function hookMethod(class_name, method_name){
        var class_method_obj_ = Java.use(class_name)[method_name];
        if (class_method_obj_.overloads.length > 1)
            hookOverloadedMethod(class_method_obj_, method_name);
        else{
            try{
                // Code for hooking method
                var argTypes = class_method_obj_.argumentTypes.map(function(a){return a.clas
                //var methodName = class_method_obj_.methodName;
                var methodName = method_name;
                send("[*]"+methodName+"("+argTypes+")");

                class_method_obj_.implementation = function () {
                        var args = Array.prototype.slice.call(arguments);
                        var retVal = this[methodName].apply(this, args);
                        // Injection
                        send("called " + methodName + "(" + args + ");");
                        return retVal;
                }

                //send("Hooked Method");
            } catch (err){
                send("Error message");
            }
        }
    }

    function hookOverloadedMethod(class_method_obj_, method_name){
        try{
            var overloadMethods = class_method_obj_.overloads;
            for (var i in overloadMethods){
                var argTypes = overloadMethods[i].argumentTypes.map(function(a){return a.cla
                send("    ["+ i +"] " + method_name +"(" + argTypes +");");
                try{
                    class_method_obj_.overload.apply(this, argTypes).implementation = functi
                        var args = Array.prototype.slice.call(arguments);
                        var retVal = this[method_name].apply(this, args);
                        send("Called overloaded "+ method_name + "(" + args+");");
                        return retVal;
                    }
                } catch(err){
                    send("Error in hooking overloaded method" + method_name +"(" + argTypes
                }
            }
            //send("Hooked Method Overloads");
        } catch(err){
            send("Error message overloading" + err);
        }
    }

});
```

Python

```python
import frida, sys, time, os

def on_message(message, data):
    if message['type'] == 'send':
        print("[*] {0}".format(message['payload']))
        if message['payload'] == 'get_hooks':
            with open(os.path.join(os.path.dirname(__file__),"hooks.json")) as hooks:
                script.post(hooks.read())
    else:
        print(message)

device = frida.get_usb_device()
pid = device.spawn(['com.wiki.hello_world'])
time.sleep(1)
process = device.attach(pid)
script = process.create_script(open(os.path.join(os.path.dirname(__file__),"recv_hooks.js"))
script.on('message', on_message)
print('[*] Running CTF')
```

```
script.load()
device.resume(pid) # Important, otherwise app will start and you will miss some early code e

sys.stdin.read()
```

Output on python console

```
/usr/bin/python2.7 /home/wra/workspace/cuckoo_frida/sending_hooks_json/send_hooks.py
[*] Running CTF
[*] Sending message
[*] get_hooks
[*] Just received string

[*] Total number of methods to be hooked is :
[*] 20
[*] Hooking :android.telephony.TelephonyManager->getDeviceId
[*]     [0] getDeviceId();
[*]     [1] getDeviceId(int);
[*] Hooking :android.telephony.TelephonyManager->getSubscriberId
[*]     [0] getSubscriberId();
[*]     [1] getSubscriberId(int);
[*] Hooking :android.telephony.TelephonyManager->getLine1Number
[*] [*]getLine1Number()
[*] Hooking :android.telephony.TelephonyManager->getNetworkOperator
[*] [*]getNetworkOperator()
[*] Hooking :android.telephony.TelephonyManager->getNetworkOperatorName
[*]     [0] getNetworkOperatorName();
[*]     [1] getNetworkOperatorName(int);
[*] Hooking :android.telephony.TelephonyManager->getSimOperatorName
[*] [*]getSimOperatorName()
[*] Hooking :android.net.wifi.WifiInfo->getMacAddress
[*] [*]getMacAddress()
[*] Hooking :android.telephony.TelephonyManager->getSimCountryIso
[*]     [0] getSimCountryIso();
[*]     [1] getSimCountryIso(int);
[*] Hooking :android.telephony.TelephonyManager->getSimSerialNumber
[*]     [0] getSimSerialNumber();
[*]     [1] getSimSerialNumber(int);
[*] Hooking :android.telephony.TelephonyManager->getNetworkCountryIso
[*] [*]getNetworkCountryIso()
[*] Hooking :android.telephony.TelephonyManager->getDeviceSoftwareVersion
[*]     [0] getDeviceSoftwareVersion();
[*]     [1] getDeviceSoftwareVersion(int);
[*] Hooking :android.os.Debug->isDebuggerConnected
[*] [*]isDebuggerConnected()
[*] Hooking :android.app.SharedPreferencesImpl$EditorImpl->putString
[*] [*]putString(java.lang.String,java.lang.String)
[*] Hooking :android.app.SharedPreferencesImpl$EditorImpl->putBoolean
[*] [*]putBoolean(java.lang.String,boolean)
[*] Hooking :android.app.SharedPreferencesImpl$EditorImpl->putInt
[*] [*]putInt(java.lang.String,int)
[*] Hooking :android.app.SharedPreferencesImpl$EditorImpl->putLong
[*] [*]putLong(java.lang.String,long)
[*] Hooking :android.app.SharedPreferencesImpl$EditorImpl->putFloat
[*] [*]putFloat(java.lang.String,float)
[*] Hooking :android.content.ContentValues->put
[*]     [0] put(java.lang.String,java.lang.Boolean);
[*]     [1] put(java.lang.String,java.lang.Byte);
[*]     [2] put(java.lang.String,java.lang.Double);
[*]     [3] put(java.lang.String,java.lang.Float);
[*]     [4] put(java.lang.String,java.lang.Integer);
[*]     [5] put(java.lang.String,java.lang.Long);
[*]     [6] put(java.lang.String,java.lang.Short);
[*]     [7] put(java.lang.String,java.lang.String);
[*]     [8] put(java.lang.String,[B);
[*] Hooking :java.net.URL->openConnection
[*]     [0] openConnection();
[*]     [1] openConnection(java.net.Proxy);
[*] Hooking :org.apache.http.impl.client.AbstractHttpClient->execute
[*]     [0] execute(org.apache.http.HttpHost,org.apache.http.HttpRequest,org.apache.http.cli
[*]     [1] execute(org.apache.http.HttpHost,org.apache.http.HttpRequest,org.apache.http.cli
[*]     [2] execute(org.apache.http.client.methods.HttpUriRequest,org.apache.http.client.Res
[*]     [3] execute(org.apache.http.client.methods.HttpUriRequest,org.apache.http.client.Res
[*]     [4] execute(org.apache.http.HttpHost,org.apache.http.HttpRequest);
[*]     [5] execute(org.apache.http.HttpHost,org.apache.http.HttpRequest,org.apache.http.pro
[*]     [6] execute(org.apache.http.client.methods.HttpUriRequest);
[*]     [7] execute(org.apache.http.client.methods.HttpUriRequest,org.apache.http.protocol.H
[*] called getSimOperatorName();
[*] called getNetworkOperator();
[*] called getLine1Number();
[*] Called overloaded getSimCountryIso();
[*] Called overloaded getDeviceId();
```

hooks.json file

```
{
    "hookConfigs": [
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getDeviceId",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getSubscriberId",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getLine1Number",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getNetworkOperator",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getNetworkOperatorName",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getSimOperatorName",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.net.wifi.WifiInfo",
            "method": "getMacAddress",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getSimCountryIso",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getSimSerialNumber",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getNetworkCountryIso",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getDeviceSoftwareVersion",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.os.Debug",
            "method": "isDebuggerConnected",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.app.SharedPreferencesImpl$EditorImpl",
            "method": "putString",
            "thisObject": false,
            "type": "globals"
        },
        {
            "class_name": "android.app.SharedPreferencesImpl$EditorImpl",
            "method": "putBoolean",
            "thisObject": false,
            "type": "globals"
        },
        {
            "class_name": "android.app.SharedPreferencesImpl$EditorImpl",
            "method": "putInt",
            "thisObject": false,
            "type": "globals"
        },
        {
            "class_name": "android.app.SharedPreferencesImpl$EditorImpl",
            "method": "putLong",
            "thisObject": false,
            "type": "globals"
        },
        {
            "class_name": "android.app.SharedPreferencesImpl$EditorImpl",
            "method": "putFloat",
            "thisObject": false,
            "type": "globals"
        },
        {
            "class_name": "android.content.ContentValues",
            "method": "put",
            "thisObject": false,
            "type": "globals"
        },
        {
            "class_name": "java.net.URL",
            "method": "openConnection",
            "thisObject": true,
            "type": "network"
        },
        {
            "class_name": "org.apache.http.impl.client.AbstractHttpClient",
            "method": "execute",
            "thisObject": false,
            "type": "network"
        }
    ],
    "trace": false
}
```

Example 9: Hooking methods from json file on device:

This is not the final code and is a bit dirty but this will give you the idea. In this example we are using frida-fs, frida-compile which enables to open a file in just one line on of code. Also note the error message in output. Its because the method entry for one of the item is null and it probably means that we need to hook the constructor of that class for that. We will add the logic for that later.

Javascript

```javascript
'use strict';
const fs = require("frida-fs");

var hooks_json =  "/data/local/tmp/hooks.json";

Java.perform(function () {
    //var classes_dict = {}; // Dictionary for keeping track of classes that we already used
    var Log = Java.use('android.util.Log');
    //var Log_i = Log.i;
    var MainActivity = Java.use('com.wiki.hello_world.MainActivity');
    var received_json = "";

    var LOGTAG_SHELL= "Droidmon-shell-";
    var LOGTAG_WORKFLOW = "Droidmon-apimonitor-";
    var LOGTAG_ERROR = "Droidmon-error";
    //public static String LOG_FILE ="/droidmon.log";
    //public static String LOG_FILE_OLD ="/droidmon_old.log";
    var PACKAGENAME = "com.wiki.hello_world";

    var hooks = JSON.parse(fs.readFileSync(hooks_json));

    var overloads;
    send("Total number of methods to be hooked is : "+hooks.hookConfigs.length);
    hookMethods(hooks);

    /*
        MY CUSTOM FUNCTIONS
        TODO: FIND A WAY TO MOVE THEM TO ANOTHER FILE
    */

    // TODO: If method name is null, its probaly constructor, do the appropriate checking fo
    function hookMethods(hooks_json){
        for (var index in hooks_json.hookConfigs){
            send("Hooking :" + hooks.hookConfigs[index].class_name + "->" + hooks.hookConfig
            hookMethod(hooks.hookConfigs[index].class_name, hooks.hookConfigs[index]);
        }
    }
```

```
function hookMethod(class_name, method_json){
    var method_name = method_json.method;
    var class_method_obj_ = Java.use(class_name)[method_json.method];
    if (class_method_obj_.overloads.length > 1)
        hookOverloadedMethod(class_method_obj_, method_json);
    else{
        try{
            // Code for hooking method
            var argTypes = class_method_obj_.argumentTypes.map(function(a){return a.clas
            //var methodName = class_method_obj_.methodName;
            //var methodName = method_json.method;
            //send("[*]"+method_json.method+"("+argTypes+")");

            class_method_obj_.implementation = function () {
                var args = Array.prototype.slice.call(arguments);
                var retVal = this[method_json.method].apply(this, args);
                // Injection start
                //send("called " + method_json.method + "(" + args + ");");
                log(method_json,args, retVal);
                // Injection end
                return retVal;
            }
        } catch (err){
            send("Error message in hookMethod" + err);
        }
    }
}

function hookOverloadedMethod(class_method_obj_, method_json){
    try{
        var overloadMethods = class_method_obj_.overloads;
        for (var i in overloadMethods){
            var argTypes = overloadMethods[i].argumentTypes.map(function(a){return a.cla
            //send("    ["+ i +"] " + method_json.method +"(" + argTypes +");");
            try{
                class_method_obj_.overload.apply(this, argTypes).implementation = functi
                    var args = Array.prototype.slice.call(arguments);
                    var retVal = this[method_json.method].apply(this, args);
                    //send("Called overloaded "+ method_json.method + "(" + args+");");
                    log(method_json, args, retVal);
                    return retVal;
                }
            } catch(err){
                send("Error in hooking overloaded method" + method_json.method +"(" + ar
            }
        }
    } catch(err){
        send("Error message overloading" + err);
    }
}

// Logging methods
function log(method_json, args, retVal) {
    try {
        if (method_json.method.includes("invoke")) logReflectionMethod(method_json, args
        else if (method_json.method.includes("write")) logProcessWriteMethod(method_json
        else if (method_json.method.includes("read")) logProcessReadMethod(method_json, 
        else if (method_json.method.includes("OpenDexFile") || method_json.method.includ
        else send("Droidmon-apimonitor"+ JSON.stringify(logGenericMethod(method_json, ar
    } catch (e){
        send("Log error " + e);
    }
}
```

```
    function logReflectionMethod() {}
    function logProcessWriteMethod() {}
    function logProcessReadMethod() {}
    function logAndDumpFile() {}
    function logGenericMethod(method_json, args, retVal) {
        var hookData = generateHookDataJson(method_json);
        if (args.length>0)
            hookData["args"] = args;
        if (typeof(retVal)!="undefined") //TODO: Check if undefined can be changed, fix it i
            hookData["result"] = retVal;
        //if (method_json.thisObject)
        hookData["this"] = method_json.thisObject; // Fix this one
        return hookData;
    }

    function generateHookDataJson(hookJson){
        // {"class":"android.os.SystemProperties","method":"get","timestamp":1515600745104,"
        return({"class":hookJson.class_name,"method":hookJson.method,"timestamp":Date.now(),

    }

    // https://github.com/idanr1986/droidmon/blob/master/src/com/cuckoodroid/droidmon/utils/

});
```

Python

```
import frida, sys, time, os, subprocess

payload = "/home/wra/workspace/cuckoodroid/frida-scripts/frida-nodes/compiled.js"
script = "/home/wra/workspace/cuckoodroid/frida-scripts/frida-nodes/app.js"

def on_message(message, data):
    try:
        if message['type'] == 'send':
                print("[*] {0}".format(message['payload']))
                #print(message)
        else:
            print(message)
    except Exception as e:
        print(message)
        print(e)

frida_ready = False
device = frida.get_usb_device()
while not frida_ready:
    try:
        time.sleep(5)
        pid = device.spawn(['com.wiki.hello_world'])
        frida_ready = True
    except Exception as e:
        print("Frida not ready yet " + str(e))

time.sleep(1)

cwd = os.path.dirname(__file__)
#print("Working Directory is : "+ cwd)

print ("Compiling NodeJS code\n")
stdout, stderr = subprocess.Popen("frida-compile "+ script+ " -o "+ payload, shell=True, std

if len(stdout) > 0:
    print(stdout)
if len(stderr) > 0:
    print("error :" + stdout)

try:
    process = device.attach(pid)
    script = process.create_script(
        open(payload).read())
    script.on('message', on_message)
    print('[*] Running Frida')
    script.load()
    device.resume(pid) # At end Important, otherwise app will start and you will miss some e
except Exception as e:
    print("Frida python binding code error :" + str(e))

sys.stdin.read(1)
```

Output on python console

```
/usr/bin/python2.7 /home/wra/workspace/cuckoodroid/frida-scripts/frida-nodes/agent_manager.p
Compiling NodeJS code

[*] Running Frida
[*] Total number of methods to be hooked is : 84
[*] Hooking :android.telephony.TelephonyManager->getDeviceIdindex0
[*] Hooking :android.telephony.TelephonyManager->getSubscriberIdindex1
[*] Hooking :android.telephony.TelephonyManager->getLine1Numberindex2
[*] Hooking :android.telephony.TelephonyManager->getNetworkOperatorindex3
[*] Hooking :android.telephony.TelephonyManager->getNetworkOperatorNameindex4
[*] Hooking :android.telephony.TelephonyManager->getSimOperatorNameindex5
[*] Hooking :android.net.wifi.WifiInfo->getMacAddressindex6
[*] Hooking :android.telephony.TelephonyManager->getSimCountryIsoindex7
[*] Hooking :android.telephony.TelephonyManager->getSimSerialNumberindex8
[*] Hooking :android.telephony.TelephonyManager->getNetworkCountryIsoindex9
[*] Hooking :android.telephony.TelephonyManager->getDeviceSoftwareVersionindex10
[*] Hooking :android.os.Debug->isDebuggerConnectedindex11
[*] Hooking :android.app.SharedPreferencesImpl$EditorImpl->putStringindex12
[*] Hooking :android.app.SharedPreferencesImpl$EditorImpl->putBooleanindex13
[*] Hooking :android.app.SharedPreferencesImpl$EditorImpl->putIntindex14
[*] Hooking :android.app.SharedPreferencesImpl$EditorImpl->putLongindex15
[*] Hooking :android.app.SharedPreferencesImpl$EditorImpl->putFloatindex16
[*] Hooking :android.content.ContentValues->putindex17
[*] Hooking :java.net.URL->openConnectionindex18
[*] Hooking :org.apache.http.impl.client.AbstractHttpClient->executeindex19
[*] Hooking :android.app.ContextImpl->registerReceiverindex20
[*] Hooking :android.app.ActivityThread->handleReceiverindex21
[*] Hooking :android.app.Activity->startActivityindex22
[*] Hooking :dalvik.system.BaseDexClassLoader->findResourceindex23
[*] Hooking :dalvik.system.BaseDexClassLoader->findLibraryindex24
[*] Hooking :dalvik.system.DexFile->loadDexindex25
[*] Hooking :dalvik.system.DexClassLoader->nullindex26
{u'columnNumber': 1, u'description': u"TypeError: cannot read property 'overloads' of undefi
[*] Droidmon-apimonitor{"class":"android.telephony.TelephonyManager","method":"getSimOperato
[*] Droidmon-apimonitor{"class":"android.telephony.TelephonyManager","method":"getNetworkOpe
[*] Droidmon-apimonitor{"class":"android.telephony.TelephonyManager","method":"getLine1Numbe
[*] Droidmon-apimonitor{"class":"android.telephony.TelephonyManager","method":"getSimCountry
[*] Droidmon-apimonitor{"class":"android.telephony.TelephonyManager","method":"getDeviceId",
```

hooks.json file

```
{
    "hookConfigs": [
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getDeviceId",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getSubscriberId",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getLine1Number",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getNetworkOperator",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getNetworkOperatorName",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getSimOperatorName",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.net.wifi.WifiInfo",
            "method": "getMacAddress",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getSimCountryIso",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getSimSerialNumber",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getNetworkCountryIso",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "getDeviceSoftwareVersion",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.os.Debug",
            "method": "isDebuggerConnected",
            "thisObject": false,
            "type": "fingerprint"
        },
        {
            "class_name": "android.app.SharedPreferencesImpl$EditorImpl",
            "method": "putString",
            "thisObject": false,
            "type": "globals"
        },
        {
            "class_name": "android.app.SharedPreferencesImpl$EditorImpl",
            "method": "putBoolean",
            "thisObject": false,
            "type": "globals"
        },
        {
            "class_name": "android.app.SharedPreferencesImpl$EditorImpl",
            "method": "putInt",
            "thisObject": false,
            "type": "globals"
        },
        {
            "class_name": "android.app.SharedPreferencesImpl$EditorImpl",
            "method": "putLong",
            "thisObject": false,
            "type": "globals"
        },
        {
            "class_name": "android.app.SharedPreferencesImpl$EditorImpl",
            "method": "putFloat",
            "thisObject": false,
            "type": "globals"
        },
        {
            "class_name": "android.content.ContentValues",
            "method": "put",
            "thisObject": false,
            "type": "globals"
        },
        {
            "class_name": "java.net.URL",
            "method": "openConnection",
            "thisObject": true,
            "type": "network"
        },
        {
            "class_name": "org.apache.http.impl.client.AbstractHttpClient",
            "method": "execute",
            "thisObject": false,
            "type": "network"
        },
        {
            "class_name": "android.app.ContextImpl",
            "method": "registerReceiver",
            "thisObject": false,
            "type": "binder"
        },
        {
            "class_name": "android.app.ActivityThread",
            "method": "handleReceiver",
            "thisObject": false,
            "type": "binder"
        },
        {
            "class_name": "android.app.Activity",
            "method": "startActivity",
            "thisObject": false,
            "type": "binder"
        },
        {
            "class_name": "dalvik.system.BaseDexClassLoader",
            "method": "findResource",
            "thisObject": false,
            "type": "dex"
        },
        {
            "class_name": "dalvik.system.BaseDexClassLoader",
            "method": "findLibrary",
            "thisObject": false,
            "type": "dex"
        },
        {
            "class_name": "dalvik.system.DexFile",
            "method": "loadDex",
            "thisObject": false,
            "type": "dex"
        },
        {
            "class_name": "dalvik.system.DexClassLoader",
            "method": null,
            "thisObject": false,
            "type": "dex"
        },
        {
            "class_name": "dalvik.system.DexClassLoader",
            "method": "load",
            "thisObject": false,
            "type": "dex"
        },
        {
            "class_name": "dalvik.system.BaseDexClassLoader",
            "method": "findResources",
            "thisObject": false,
            "type": "dex"
        },
        {
            "class_name": "dalvik.system.DexFile",
            "method": "loadClass",
            "thisObject": false,
            "type": "dex"
        },
        {
            "class_name": "dalvik.system.DexFile",
            "method": null,
            "thisObject": false,
            "type": "dex"
        },
        {
            "class_name": "dalvik.system.PathClassLoader",
            "method": null,
            "thisObject": false,
            "type": "dex"
        },
        {
            "class_name": "java.lang.reflect.Method",
            "method": "invoke",
            "thisObject": false,
            "type": "reflection"
        },
        {
            "class_name": "javax.crypto.spec.SecretKeySpec",
            "method": null,
            "thisObject": false,
            "type": "crypto"
        },
        {
            "class_name": "javax.crypto.Cipher",
            "method": "doFinal",
            "thisObject": true,
            "type": "crypto"
        },
        {
            "class_name": "javax.crypto.Mac",
            "method": "doFinal",
            "thisObject": false,
            "type": "crypto"
        },
        {
            "class_name": "android.app.ApplicationPackageManager",
            "method": "setComponentEnabledSetting",
            "thisObject": false,
            "type": "generic"
        },
        {
            "class_name": "android.app.NotificationManager",
            "method": "notify",
            "thisObject": false,
            "type": "generic"
        },
        {
            "class_name": "android.util.Base64",
            "method": "decode",
            "thisObject": false,
            "type": "generic"
        },
        {
            "class_name": "android.telephony.TelephonyManager",
            "method": "listen",
            "thisObject": false,
            "type": "generic"
        },
        {
            "class_name": "android.util.Base64",
            "method": "encode",
            "thisObject": false,
            "type": "generic"
        },
        {
            "class_name": "android.util.Base64",
            "method": "encodeToString",
            "thisObject": false,
            "type": "generic"
        },
        {
            "class_name": "android.net.ConnectivityManager",
            "method": "setMobileDataEnabled",
            "thisObject": false,
            "type": "generic"
        },
        {
            "class_name": "android.content.BroadcastReceiver",
            "method": "abortBroadcast",
            "thisObject": false,
            "type": "generic"
        },
        {
            "class_name": "android.telephony.SmsManager",
            "method": "sendTextMessage",
            "thisObject": false,
            "type": "sms"
        },
        {
            "class_name": "android.telephony.SmsManager",
            "method": "sendMultipartTextMessage",
            "thisObject": false,
            "type": "sms"
        },
        {
            "class_name": "java.lang.Runtime",
            "method": "exec",
            "thisObject": false,
            "type": "runtime"
        },
        {
            "class_name": "java.lang.ProcessBuilder",
            "method": "start",
            "thisObject": true,
            "type": "runtime"
        },
        {
            "class_name": "java.io.FileOutputStream",
            "method": "write",
            "thisObject": false,
            "type": "runtime"
        },
        {
            "class_name": "java.io.FileInputStream",
            "method": "read",
            "thisObject": false,
            "type": "runtime"
        },
        {
            "class_name": "android.app.ActivityManager",
            "method": "killBackgroundProcesses",
            "thisObject": false,
            "type": "runtime"
        },
        {
            "class_name": "android.os.Process",
            "method": "killProcess",
            "thisObject": false,
            "type": "runtime"
        },
        {
            "class_name": "android.content.ContentResolver",
            "method": "query",
            "thisObject": false,
            "type": "content"
        },
        {
            "class_name": "android.content.ContentResolver",
            "method": "registerContentObserver",
            "thisObject": false,
            "type": "content"
        },
        {
            "class_name": "android.content.ContentResolver",
            "method": "insert",
            "thisObject": false,
            "type": "content"
        },
        {
            "class_name": "android.accounts.AccountManager",
            "method": "getAccountsByType",
            "thisObject": false,
            "type": "content"
        },
        {
            "class_name": "android.accounts.AccountManager",
            "method": "getAccounts",
            "thisObject": false,
            "type": "content"
        },
        {
            "class_name": "android.location.Location",
            "method": "getLatitude",
            "thisObject": false,
            "type": "content"
        },
        {
            "class_name": "android.location.Location",
            "method": "getLongitude",
            "thisObject": false,
            "type": "content"
        },
        {
            "class_name": "android.content.ContentResolver",
            "method": "delete",
            "thisObject": false,
            "type": "content"
        },
        {
            "class_name": "android.media.AudioRecord",
            "method": "startRecording",
            "thisObject": false,
            "type": "content"
        },
        {
            "class_name": "android.media.MediaRecorder",
            "method": "start",
            "thisObject": false,
            "type": "content"
        },
        {
            "class_name": "android.os.SystemProperties",
            "method": "get",
            "thisObject": false,
            "type": "content"
        },
        {
            "class_name": "android.app.ApplicationPackageManager",
            "method": "getInstalledPackages",
            "thisObject": false,
            "type": "content"
        },
        {
            "class_name": "libcore.io.IoBridge",
            "method": "open",
            "thisObject": false,
            "type": "file"
        },
        {
            "class_name": "libcore.io.IoBridge",
            "method": "close",
            "thisObject": true,
            "type": "file"
        },
        {
            "class_name": "android.app.ActivityManager",
            "method": "getRunningAppProcesses",
            "thisObject": false,
            "type": "binder"
        },
        {
            "class_name": "android.app.ActivityManager",
            "method": "getRunningTasks",
            "thisObject": false,
            "type": "binder"
        },
        {
            "class_name": "dalvik.system.DexFile",
            "method": "openDexFile",
            "thisObject": false,
            "type": "dex"
        },
        {
            "class_name": "java.lang.Runtime",
            "method": "load",
            "thisObject": false,
            "type": "dex"
        },
        {
            "class_name": "android.os.Process",
            "method": "start",
            "thisObject": false,
            "type": "runtime"
        },
        {
            "class_name": "android.content.ContextWrapper",
            "method": "startActivities",
            "thisObject": false,
            "type": "binder"
        },
        {
            "class_name": "android.content.ContextWrapper",
            "method": "startService",
            "thisObject": false,
            "type": "binder"
        },
        {
            "class_name": "android.content.ContextWrapper",
            "method": "startActivity",
            "thisObject": false,
            "type": "binder"
        },
        {
            "class_name": "android.content.ContextWrapper",
            "method": "sendBroadcast",
            "thisObject": false,
            "type": "binder"
        },
        {
            "class_name": "java.net.ProxySelectorImpl",
            "method": "select",
            "thisObject": false,
            "type": "generic"
        },
        {
            "class_name": "org.apache.http.impl.client.DefaultHttpClient",
            "method": "switch_proxy",
            "thisObject": false,
            "type": "generic"
        },
        {
            "class_name": "android.webkit.WebView",
            "method": "addJavascriptInterface",
            "thisObject": false,
            "type": "network"
        },
        {
            "class_name": "android.webkit.WebView",
            "method": "setWebChromeClient",
            "thisObject": false,
            "type": "network"
        },
        {
            "class_name": "android.webkit.WebView",
            "method": "setWebViewClient",
            "thisObject": false,
            "type": "network"
        },
        {
            "class_name": "android.content.ContextWrapper",
            "method": "openFileOutput",
            "thisObject": false,
            "type": "file"
        },
        {
            "class_name": "android.app.AlarmManager",
            "method": "setAlarmClock",
            "thisObject": false,
            "type": "runtime"
        },
        {
            "class_name": "android.app.AlarmManager",
            "method": "set",
            "thisObject": false,
            "type": "file"
        },
        {
            "class_name": "java.io.File",
            "method": "exists",
            "thisObject": true,
            "type": "file"
        }

    ],
    "trace": false
}
```

TIPS and Undocumented features:

1. Getting Arguments and name from method wrapper: Source
   Due to some reason, I can only get the name of overloaded method.

```
                var argTypes = overloadMethods[i].argumentTypes.map(function(a){return a
                var methodName = overloadMethods[i].methodName;
```

Other properties are: implementation, returnType, canInvokeWith, PENDING_CALLS, loader, $new, $alloc, $init, class etc. (Verify that these are correct properties and are not result of misinterpretation of source code)

2. **JavaScript Internals:** JavaScript has a very different structure than other languages. It is designed to be run in the browser and browser needs to have a specific refresh rate to display web-pages and respond to events like click etc. Due to this reason it heavily rely on concurrent programming or it heaving use Asynchronous callbacks. All callbacks are pushed to the Message Queue where they are executed only when call stack is empty. Also the browser rendering has a priority over callbacks waiting in message queue so Browser is rendering the page in between back to back callback calls from message queue. That is why it is recommended to keep callbacks smaller so that it doesn't block rendering. Here is very useful video explaining this in nice graphical way https://www.youtube.com/watch?v=8aGhZQkoFbQ
For internal of NodeJS please visit this link https://nodejs.org/en/docs/guides/event-loop-timers-and-nexttick/

Resources

Opening files
https://codeshare.frida.re/@dzonerzy/filedumper/

https://github.com/AndroidTamer/frida-push

Frida anti root
https://codeshare.frida.re/@dzonerzy/fridantiroot/

File Dumper:
https://codeshare.frida.re/@dzonerzy/filedumper/

Mobile Application hacking Paper:
https://www.exploit-db.com/papers/44145/

TODOs

1. Inject own written java classes (See)

hello_world.apk - This is application is for testing the anti-emulator capabilities of our device. It obtains different information from the system which can be used to detect emulation environment or root. It also displays stack trace of some its methods and that informa (1.98 MB) Waqar Rashid, 2018-01-11 03:53 PM

hello_world_no_hooking.png - Part of output of hello_world.apk without hooking (70.3 KB) Waqar Rashid, 2018-01-11 03:59 PM

getPhone_java_code.png - Java source code of the method we would be hooking using Frida (68.6 KB) Waqar Rashid, 2018-01-11 04:05 PM

hello_world_python_execution.png - Exectution of python script on host and original returned value getPhone() (115 KB) Waqar Rashid, 2018-01-11 04:27 PM

hello_word_after_hooking.png - Application window when getPhone() was hooked (70.1 KB) Waqar Rashid, 2018-01-11 04:27 PM

hello_world_python_execution.png.png (108 KB) Waqar Rashid, 2018-01-11 04:32 PM

logcat_wrapper.png (22.2 KB) Waqar Rashid, 2018-01-15 01:03 PM

interactive_frida.png (69 KB) Waqar Rashid, 2018-01-16 03:38 PM