本文是 Piasy 原创，发表于 [https://blog.piasy.com](https://blog.piasy.com/)，请阅读原文支持原创 <https://blog.piasy.com/2017/06/01/frida-android-hook/>

最近的工作需要通过 hook 研究一些目标 APP 的系统 API 调用，很早就了解到了 Frida，这次终于可以体验一把了。花了一整天的时间，才终于把环境搭好，主要是准备手机系统花了时间。[示例代码可以在 GitHub 获取。](https://github.com/Piasy/AndroidPlayground/tree/master/try/FridaDemo)

【2017.6.4 更新】：自动生成 Javascript hook 脚本的工具有了一个可用的版本，[check it out :)](https://github.com/Piasy/FridaAndroidTracer)。

**环境准备**

最终成功的环境：

* macOS 10.12.5 (16F73)
* python 3.6.1
* Frida 10.0.9
* Nexus 5X
* [Android 6.0 factory image MDA89E](https://dl.google.com/dl/android/aosp/bullhead-mda89e-factory-d716b566.zip)
* [卡刷 root](http://www.teamandroid.com/2016/08/04/root-nexus-5x-android-6-0-1-mtc20f-marshmallow-security-update/)

之前失败的手机系统：

* Nexus 5X，自己编译的 AOSP，7.1.1\_r24，由于是工程镜像，所以自带 root，SELinux 可以设置为 Permissive mode；在手机上运行 frida-server 之后，电脑上的所有指令都提示 connection refused；
* 三星 note 3，KingRoot，SELinux 无法设置为 Permissive mode；在手机上运行 frida-server 之后，电脑上可以执行 frida-ps，但 frida-trace 提示 Failed to attach: failed to execute child process “/data/local/tmp/re.frida.server/frida-helper-32” (Permission denied)；

最后分享一个小工具：[frida-server start/stop 脚本](https://github.com/Piasy/frida-push)。

【2017.6.2 更新】：运行得好好的 Frida，今天下午突然又不行了，提示 connection refused，折腾半天发现是电脑不仅连接了 Nexus 5X，还连上了一台 iPhone，拔掉 iPhone 就好了，希望大家不会遇见这样的窘境。

**hook 需求**

MainActivity 的代码如下：

**public** **class** **MainActivity** **extends** **AppCompatActivity** **{**

@Override

**protected** **void** **onCreate(**Bundle savedInstanceState**)** **{**

**super.**onCreate**(**savedInstanceState**);**

setContentView**(**R**.**layout**.**activity\_main**);**

findViewById**(**R**.**id**.**mBtnTest**).**setOnClickListener**(new** View**.**OnClickListener**()** **{**

@Override

**public** **void** **onClick(final** View v**)** **{**

private\_func**();**

private\_func**(**123**);**

private\_func**(**"str"**);**

private\_func**(**"str"**,** **true);**

System**.**out**.**println**(**"func\_with\_ret(4): " **+** func\_with\_ret**(**4**));**

**}**

**});**

**}**

**private** **void** **private\_func()** **{**

System**.**out**.**println**(**"private\_func()"**);**

**}**

**private** **void** **private\_func(int** i**)** **{**

System**.**out**.**println**(**"private\_func(int) " **+** i**);**

**}**

**private** **void** **private\_func(**String s**)** **{**

System**.**out**.**println**(**"private\_func(String) " **+** s**);**

**}**

**private** **void** **private\_func(**String s**,** **boolean** b**)** **{**

System**.**out**.**println**(**"private\_func(String, boolean) " **+** s **+** ", " **+** b**);**

**}**

**private** **int** **func\_with\_ret(int** i**)** **{**

System**.**out**.**println**(**"func\_with\_ret(int) " **+** i**);**

**return** i **\*** i**;**

**}**

**}**

**打日志**

**import** frida, sys

package\_name **=** "com.powersmarttv.www.livestreamp31"

**def** **get\_messages\_from\_js**(message, data):

**print**(message)

**def** **hook\_log\_on\_resume**():

hook\_code **=** """ Java.perform(function () { var Activity = Java.use("android.app.Activity"); Activity.onResume.implementation = function () { send("onResume() " + this); this.onResume(); }; }); """

**return** hook\_code

**def** **main**():

process **=** frida**.**get\_device\_manager()**.**enumerate\_devices()[**-**1]**.**attach(package\_name)

script **=** process**.**create\_script(hook\_log\_on\_resume())

script**.**on('message', get\_messages\_from\_js)

script**.**load()

sys**.**stdin**.**read()

**if** \_\_name\_\_ **==** '\_\_main\_\_':

main()

退后台再切回前台，控制台日志如下：

{'type': 'send', 'payload': 'onResume() com.github.piasy.fridademo.MainActivity@1645013'}

**编写 Javascript 代码**

上面的代码中，Javascript 是字符串形式，编写过程中无法利用任何代码的特性，不好。我们可以编写 Javascript 代码，并通过 Frida CLI 加载 js 代码。

运行 Frida CLI 并 attach 到目标进程：

frida -U com.github.piasy.fridademo -l log\_on\_resume.js

log\_on\_resume.js 内容：

Java.perform(**function** () {

**var** Activity **=** Java.use("android.app.Activity");

Activity.onResume.implementation **=** **function** () {

send("onResume() " **+** **this**);

**this**.onResume();

};

});

onResume 时日志会在 CLI 命令行中打印出来。CLI 模式下还能重新加载 js 脚本，在 CLI 命令行中输入 %reload 指令，回车即可。不过我发现，Frida CLI 还有热加载功能，我们修改脚本保存后，它会自动重新加载脚本。

**hook 重载函数**

Java.perform(function () {

var MainActivity = Java.use("com.github.piasy.fridademo.MainActivity");

MainActivity.private\_func.overload().implementation = function () {

send("private\_func()");

this.private\_func();

};

MainActivity.private\_func.overload("int").implementation = function (i) {

send("private\_func(int): " + i);

this.private\_func(i);

};

MainActivity.private\_func.overload("java.lang.String").implementation = function () {

send("private\_func(String): " + arguments[0]);

this.private\_func(arguments[0]);

};

MainActivity.private\_func.overload("java.lang.String", "boolean").implementation = function (s, b) {

send("private\_func(String,boolean): " + s + ", " + b);

this.private\_func(s, b);

};

});

点击 TEST 按钮，控制台日志如下：

{'type': 'send', 'payload': 'private\_func()'} {'type': 'send', 'payload': 'private\_func(int): 123'} {'type': 'send', 'payload': 'private\_func(String): str'} {'type': 'send', 'payload': 'private\_func(String,boolean): str, **true**'}

要点：

* 目标方法有多个重载版本时，没有参数的版本，也需要调用 overload()；
* 多参数/不同参数，体现在 overload 函数的参数列表上；
* primitive type 对应的 overload 参数名即为类型名，对象则为全引用名；
* 如果 overload 找不到匹配的方法，frida 会给出错误日志，例如：

{'type': 'error', 'description': "Error: private\_func(): argument count of 2 does not match any of: \n\t.overload() \n\t.overload('java.lang.String') \n\t.overload('int')", 'stack': "Error: private\_func(): argument count of 2 does not match any of: \n\t.overload() \n\t.overload('java.lang.String') \n\t.overload('int')\n at throwOverloadError (frida/node\_modules/frida-java/lib/class-factory.js:1449)\n at frida/node\_modules/frida-java/lib/class-factory.js:858\n at [anon] (script1.js:16)\n at frida/node\_modules/frida-java/lib/vm.js:33\n at y (frida/node\_modules/frida-java/index.js:322)\n at frida/node\_modules/frida-java/index.js:296\n at frida/node\_modules/frida-java/lib/vm.js:33\n at java.js:1369\n at script1.js:20", 'fileName': 'frida/node\_modules/frida-java/lib/class-factory.js', 'lineNumber': 1449, 'columnNumber': 1}

* 修改了 Java 代码之后，需要 build 安装，并运行起来，hook 代码才能感知到（其实废话，hook 的是手机，在电脑里面改了代码，手机里运行的代码当然没变化）；
* 调用参数通过 arguments 数组访问，也可以在 implementation 函数中声明对应的形参；

**修改函数调用**

Java.perform(**function** () {

**var** MainActivity **=** Java.use("com.github.piasy.fridademo.MainActivity");

MainActivity.private\_func.overload("java.lang.String", "boolean").implementation **=** **function** (s, b) {

send("private\_func(String,boolean): " **+** s **+** ", " **+** b);

**this**.private\_func("HOOKED!")

};

});

通过 logcat 日志，我们发现成功修改了调用的函数：

***06-01 11:38:33.993 8827-8827/com.github.piasy.fridademo I/System.out: private\_func()***

***06-01 11:38:33.994 8827-8827/com.github.piasy.fridademo I/System.out: private\_func(int) 123***

***06-01 11:38:33.994 8827-8827/com.github.piasy.fridademo I/System.out: private\_func(String) str***

***06-01 11:38:33.998 8827-8827/com.github.piasy.fridademo I/System.out: private\_func(String) HOOKED!***

**返回指定函数值**

**function** **print\_args**() {

**var** str **=** "";

**for** (**var** i **=** 0; i **<** **arguments**.length; i**++**) {

str **+=** **arguments**[i] **+** ", "

}

**return** str;

}

Java.perform(**function** () {

**var** MainActivity **=** Java.use("com.github.piasy.fridademo.MainActivity");

MainActivity.func\_with\_ret.implementation **=** **function**(i) {

send("func\_with\_ret(int): " **+** i);

**return** 100;

};

**var** AudioRecord **=** Java.use("android.media.AudioRecord");

AudioRecord.getMinBufferSize.implementation **=** **function**(sampleRateInHz, channelConfig, audioFormat) {

**var** real **=** **this**.getMinBufferSize(sampleRateInHz, channelConfig, audioFormat)

send("getMinBufferSize: " **+** print\_args(sampleRateInHz, channelConfig, audioFormat) **+** "real ret: " **+** real);

**return** **-**1;

};

});

通过 logcat 日志，我们发现成功修改了返回值：

***06-01 21:30:34.024 5597-5597/com.github.piasy.fridademo I/System.out: func\_with\_ret(4): 100***

***06-01 21:30:34.027 5597-5597/com.github.piasy.fridademo I/System.out: getMinBufferSize(16000, 16, 2): -1***

**小结**

Frida 的基本使用算是搞定了，但如果我想要 hook 目标 APP 对某个类所有方法的调用，怎么实现？手写代码显然太 low，所以我打算写一个工具，从 AOSP 获取目标类的代码，从中解析 public API，然后生成对应的 Javascript hook 脚本，stay tuned :)

【2017.6.4 更新】：自动生成 Javascript hook 脚本的工具有了一个可用的版本，[check it out :)](https://github.com/Piasy/FridaAndroidTracer)。

https://blog.piasy.com/2017/06/01/frida-android-hook/