Frida.Android.Practice (ssl unpinning)

瘦蛟舞 / 2018-05-13 16:20:00 / 浏览数 5633 技术文章 技术文章 顶(1) 踩(0)

> Author:瘦蛟舞
> Create:20180124

安卓证书锁定解除的工具
对之前发布工具的文章补充,后续还会写一篇证书锁定方案的文章.

目录:

0x00 功能介绍竞品对比

官方主页

github

Inject JavaScript to explore native apps on Windows, Mac, Linux, iOS and Android.

- Hooking Functions
- Modifying Function Arguments
- Calling Functions
- Sending messages from a target process
- Handling runtime errors from JavaScript
- Receiving messages in a target process
- Blocking receives in the target process

相对于xposed或cydia

优势:

- 更改脚本不用重启设备(有些xposed插件也可以做到)
- 对native hook支持较好
- 开发更便捷(简单的模块确实如此)
- 兼容性更好,支持设备和系统版本更广
- 不用单独处理multidex(classLoader问题).

劣势:

- 不适合写过于复杂的项目,影响app性能比较明显
- 需要自己注意脚本加载时机
- 相对容易被检测到,都这样吧.
- app启动后进行attach.可以使用-f参数frida来生成已经注入的进程(先注入Zygote为耗时操作),通常配合--no-pause使用.
- PY JS脚本混杂排错困难(-l 选项直接写js脚本,新版本错误提示已经非常人性化了.)
- E4A这种中文的代码直接GG.
- 不能全局hook也就是不能一次性hook所有app.只能指定进程hook.

0x01 基础入门设置

PC端设置

python环境

```
$ pip install -U frida
```

可选:源码编译

```
$ git clone git://github.com/frida/frida.git
$ cd frida
$ make
```

Android设备设置

首先下载android版frida-server,尽量保证与fridaServer与pc上的frida版本号一致.

```
» frida --version
```

```
10.6.55
```

完整frida-server release地址

https://github.com/frida/frida/releases

```
# getprop ro.product.cpu.abi
```

```
x86
```

下一步部署到android设备上:

```
#!bash
$ adb push frida-server /data/local/tmp/
```

跑起来

设备上运行frida-server:

```
root@android:/ # chmod 700 frida-server
root@android:/ # /data/local/tmp/frida-server -t 0 (■■■root■■■)
root@android:/ # /data/local/tmp/frida-server
```

电脑上运行adb forward tcp转发:

```
adb forward tcp:27042 tcp:27042
   adb forward tcp:27043 tcp:27043
```

27042端口用于与frida-server通信,之后的每个端口对应每个注入的进程.

运行如下命令验证是否成功安装:

```
#!bash
$ frida-ps -R
```

正常情况应该输出进程列表如下:

```
PID NAME
1590 com.facebook.katana
13194 com.facebook.katana:providers
12326 com.facebook.orca
13282 com.twitter.android
...
```

0x02 免root使用frida

针对无壳app,有壳app需要先脱壳.

手动完成frida gadget注入和调用.

1.apktool反编译apk

```
$ apktool d test.apk -o test
```

2.将对应版本的gadget拷贝到/lib没有了下.例如arm32的设备路径如下.

/lib/armeabi/libfrida-gadget.so

下载地址:

https://github.com/frida/frida/releases/

3.smali注入加载library,选择application类或者Activity入口.

```
const-string v0, "frida-gadget" invoke-static {v0}, Ljava/lang/System;->loadLibrary(Ljava/lang/String;)V
```

4.如果apk没有网络权限需要在配置清单中加入如下权限申明

```
<uses-permission android:name="android.permission.INTERNET" />
```

5.回编译apk

```
$ apktool b -o newtest.apk test/
```

6.重新签名安装运行.成功后启动app会有如下日志

```
Frida: Listening on TCP port 27042
```

使用objection自动完成frida gadget注入到apk中.

兼容性较差,不是很推荐.

```
» pip3 install -U objection
» objection patchapk -s yourapp.apk
```

0x03 JAVA hook 实战 SSL Pinning bypass

实战如何使用Frida,就较常见的证书锁定来做演练.要想绕过证书锁定抓明文包就得先知道app是如何进行锁定操作的.然后再针对其操作进行注入解锁.

客户端关于证书处理的逻辑按照安全等级我做了如下分类:

安全等级	策略	信任范围	破解方法
Level 0	完全兼容策略	信任所有证书包括自签发证书	无需特殊操作
1	系统/浏览器默认策略	信任系统或浏览内置CA证书以及用户安装证书(android 7.0开始默认不信任用户导入的证书)	设备安装代理证书
2	CA Pinning Root (intermediate) certificate pinning	信任指定CA颁发的证书	hook注入等方式篡改锁定逻辑
3	Leaf Certificate pinning	信任指定站点证书	hook注入等方式篡改锁定逻辑 如遇双向锁定需将app自带证书导入代理软件

文章要对抗的是最后两种锁定的情况(预告:关于证书锁定方案细节另有文章待发布).

注意这里要区分开攻击场景,证书锁定是用于对抗中间人攻击的而非客户端注入,不要混淆.

工具已经开源 : https://github.com/WooyunDota/DroidSSLUnpinning

HttpsURLConnection with a PinningTrustManager

apache http client 因为从api23起被android抛弃,使用率太低就先不管了.

使用传统的HttpURLConnection类封装请求,客户端锁定操作需要实现X509TrustManager接口的checkServerTrusted方法,通过对比预埋证书信息与请求网站的的证书来判

https://github.com/moxie0/AndroidPinning/blob/master/src/org/thoughtcrime/ssl/pinning/PinningTrustManager.java

```
public void checkServerTrusted(X509Certificate[] chain, String authType)
     throws CertificateException
 {
   if (cache.contains(chain[0])) {
     return;
   }

   // Note: We do this so that we'll never be doing worse than the default
   // system validation.  It's duplicate work, however, and can be factored
   // out if we make the verification below more complete.
   checkSystemTrust(chain, authType);
   checkPinTrust(chain);
   cache.add(chain[0]);
 }
```

知道锁定方法就可以hook解锁了,注入SSLContext的init方法替换信任所有证书的TrustManger

```
// Get a handle on the init() on the SSLContext class
var SSLContext_init = SSLContext.init.overload(
   '[Ljavax.net.ssl.KeyManager;', '[Ljavax.net.ssl.TrustManager;', 'java.security.SecureRandom');
```

```
// Override the init method, specifying our new TrustManager
SSLContext_init.implementation = function (keyManager, trustManager, secureRandom) {

   quiet_send('Overriding SSLContext.init() with the custom TrustManager');

   SSLContext_init.call(this, null, TrustManagers, null);
};
```

okhttp ssl pinning

okhttp将锁定操作封装的更人性化,你只要在client build时加入域名和证书hash即可.

okhttp3.x 锁定证书示例代码

```
String hostname = "yourdomain.com";
CertificatePinner certificatePinner = new CertificatePinner.Builder()
    .add(hostname, "sha256/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=")
    .build();
OkHttpClient client = OkHttpClient.Builder()
    .certificatePinner(certificatePinner)
    .build();

Request request = new Request.Builder()
    .url("https://" + hostname)
    .build();
client.newCall(request).execute();
```

frida Unpinning script for okhttp

```
setTimeout(function(){
       Java.perform(function () {
           //okttp3.x unpinning
           try {
               var CertificatePinner = Java.use("okhttp3.CertificatePinner");
               CertificatePinner.check.overload('java.lang.String', '[Ljava.security.cert.Certificate;').implementation = func
                   // do nothing
                   console.log("Called! [Certificate]");
                   return;
               };
               CertificatePinner.check.overload('java.lang.String', 'java.util.List').implementation = function(p0, p1){
                   // do nothing
                   console.log("Called! [List]");
                   return;
               };
           } catch (e) {
            console.log("okhttp3 not found");
           }
           //okhttp unpinning
           try {
               var OkHttpClient = Java.use("com.squareup.okhttp.OkHttpClient");
               OkHttpClient.setCertificatePinner.implementation = function(certificatePinner){
                   // do nothing
                   console.log("Called!");
                   return this;
               };

               // Invalidate the certificate pinnet checks (if "setCertificatePinner" was called before the previous invalidat
               var CertificatePinner = Java.use("com.squareup.okhttp.CertificatePinner");
               CertificatePinner.check.overload('java.lang.String', '[Ljava.security.cert.Certificate;').implementation = func
                   // do nothing
                   console.log("Called! [Certificate]");
                   return;
               };
               CertificatePinner.check.overload('java.lang.String', 'java.util.List').implementation = function(p0, p1){
                   // do nothing
                   console.log("Called! [List]");
                   return;
               };
           } catch (e) {
            console.log("okhttp not found");
           }


       });

},0);
```

webview ssl pinning

这种场景比很少见,本文拿一个开源项目举例.
https://github.com/menjoo/Android-SSL-Pinning-WebViews
例子中的网站 https://www.infosupport.com/
证书已经更新过一次,代码中的证书info是2015年的,而线上证书已于2017年更换,所以导致pinning失效,直接使用pinning无法访问网站.

这个开源项目的锁定操作本质是拦截webview的请求后自己用httpUrlConnection复现请求再锁定证书.貌似和之前一样,但是这里的关键不是注入点而是注入时机!

这个例子和上文注入点一样hook
SSLcontext即可Unpinning,关键在于hook时机,如果用xposed来hook就没有问题,但是用frida来hook在app启动后附加便会失去hook到init方法的时机.因为pinning操作在
onCreate时调用而我们附加是在onCreate之后执行.需要解决能像xposed一样启动前就注入或者启动时第一时间注入.

```
private void prepareSslPinning() {
       // Create keystore
       KeyStore keyStore = initKeyStore();

       // Setup trustmanager factory
       String algorithm = TrustManagerFactory.getDefaultAlgorithm();
       TrustManagerFactory tmf = null;
       try {
           tmf = TrustManagerFactory.getInstance(algorithm);
           tmf.init(keyStore);

           // Set SSL context
           sslContext = SSLContext.getInstance("TLS");
           sslContext.init(null, tmf.getTrustManagers(), null);
       } catch (NoSuchAlgorithmException e) {
           e.printStackTrace();
       } catch (KeyStoreException e) {
           e.printStackTrace();
       } catch (KeyManagementException e) {
           e.printStackTrace();
       }
   }
```

首选想到是spawn,但是spawn后并没有将脚本自动load..(
LD_PRELOAD 条件苛刻不考虑),也就是使用-f参数的时候-l参数并未生效.

```
frida -U -f com.example.mennomorsink.webviewtest2 --no-pause -l sharecode/objectionUnpinning.js
```

改由python 来完成spawn注入

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import frida, sys, re, sys, os
from subprocess import Popen, PIPE, STDOUT
import codecs, time

if (len(sys.argv) > 1):
   APP_NAME = str(sys.argv[1])
else:
   APP_NAME = "sg.vantagepoint.uncrackable3"

def sbyte2ubyte(byte):
   return (byte % 256)

def print_result(message):
   print ("[!] Received: [%s]" %(message))

def on_message(message, data):
   if 'payload' in message:
       data = message['payload']
       if type(data) is str:
           print_result(data)
       elif type(data) is list:
           a = data[0]
           if type(a) is int:
               hexstr = "".join([("%02X" % (sbyte2ubyte(a))) for a in data])
               print_result(hexstr)
               print_result(hexstr.decode('hex'))
           else:
               print_result(data)
               print_result(hexstr.decode('hex'))
       else:
           print_result(data)
   else:
       if message['type'] == 'error':
           print (message['stack'])
       else:
           print_result(message)


def kill_process():
   cmd = "adb shell pm clear {} 1> /dev/null".format(APP_NAME)
   os.system(cmd)

kill_process()

try:
   with codecs.open("hooks.js", 'r', encoding='utf8') as f:
       jscode  = f.read()
       device  = frida.get_usb_device(timeout=5)
       pid     = device.spawn([APP_NAME])
       session = device.attach(pid)
       script  = session.create_script(jscode)
       device.resume(APP_NAME)
       script.on('message', on_message)
       print ("[*] Intercepting on {} (pid:{})...".format(APP_NAME,pid))
       script.load()
       sys.stdin.read()
except KeyboardInterrupt:
       print ("[!] Killing app...")
       kill_process()
       time.sleep(1)
       kill_process()
```

成功Unpinning .(app启动后需要前后台切换一次才会成功hook到init,猜测是因为pinning初始化是在Activity onCreate时完成的.frida注入onCreate有点问题.https://github.com/frida/frida-java/issues/29)

```
'use strict';
setImmediate(function() {
 send("hooking started");
 Java.perform(function() {
 var X509TrustManager = Java.use('javax.net.ssl.X509TrustManager');
 var SSLContext = Java.use('javax.net.ssl.SSLContext');

 var TrustManager = Java.registerClass({
     name: 'com.sensepost.test.TrustManager',
     implements: [X509TrustManager],
     methods: {
         checkClientTrusted: function (chain, authType) {
         },
         checkServerTrusted: function (chain, authType) {
         },
         getAcceptedIssuers: function () {
             return [];
         }
     }
 });
 // Prepare the TrustManagers array to pass to SSLContext.init()
 var TrustManagers = [TrustManager.$new()];
 send("Custom, Empty TrustManager ready");
 // Override the init method, specifying our new TrustManager
 SSLContext.init.implementation = function (keyManager, trustManager, secureRandom) {
     send("Overriding SSLContext.init() with the custom TrustManager");
     this.init.call(this, keyManager, TrustManagers, secureRandom);
 };
 });
});
```

日志如下

```
» python application.py com.example.mennomorsink.webviewtest2
[*] Intercepting on com.example.mennomorsink.webviewtest2 (pid:1629)...

[!] Received: [hooking started]

[!] Received: [Custom, Empty TrustManager ready]

[!] Received: [Overriding SSLContext.init() with the custom TrustManager]
```

0x04 Native hook

没有合适公开的例子,就拿 https://www.52pojie.cn/thread-611938-1-1.html 帖子中提到的无法 hook ndk 中 getInt 函数问题来做演示.

ndk代码

```
#define  LOGI(...)  __android_log_print(ANDROID_LOG_INFO, "hooktest", __VA_ARGS__)
int getInt(int i)
{
   return i+99;
}

extern "C"   JNIEXPORT jstring   JNICALL Java_mi_ndk4frida_MainActivity_stringFromJNI(
       JNIEnv *env,
       jobject /* this */) {
   LOGI("[+] %d\n", getInt(2));
   return env->NewStringUTF("Hello from C++");
}
```

关键在于对指针和函数入口的理解,例子用了偏移寻址和符号寻址两种方式做对比,偏移和导出符号均可通过IDA静态分析取得,最后效果是一样的.

hook 代码

```
var fctToHookPtr = Module.findBaseAddress("libnative-lib.so").add(0x5A8);

console.log("fctToHookPtr is at " + fctToHookPtr.or(1));

var getIntAddr = Module.findExportByName("libnative-lib.so" , "_Z6getInti");

console.log("getIntAddr is at " + getIntAddr);

var errorAddr = Module.findExportByName("libnative-lib.so","getInt");

var absoluteAddr;
exports = Module.enumerateExportsSync("libnative-lib.so");
for(i=0; i<exports.length; i++){
   console.log("exports func " + i + " " + exports[i].name);
   if (exports[i].name == "_Z6getInti") {
       absoluteAddr = exports[i].address ;
       console.log("_Z6getInti addr = " + exports[i].address);
       var offset = exports[i].address - Module.findBaseAddress("libnative-lib.so") ;
       console.log("offset addr = " + offset.toString(16).toUpperCase() );
   }
   // exports func 0 _Z6getInti
   // exports func 1 Java_mi_ndk4frida_MainActivity_stringFromJNI
   // exports func 2 _ZN7_JNIEnv12NewStringUTFEPKc
}

//fctToHookPtr.or(1) , getIntAddr , absoluteAddr  are  function hook enter address.
```

```
try {
   var fungetInt = new NativeFunction(fctToHookPtr.or(1), 'int', ['int']);
   console.log("invoke 99 > " + fungetInt(99) );
} catch (e) {
   console.log("invoke getInt failed >>> " + e.message);
} finally {

}



Interceptor.attach(getIntAddr, {
   onEnter: function(args) {
       //args and retval are nativePointer...
       console.log("arg = " + args[0].toInt32());
       // //Error: access violation accessing 0x2
       // console.log(hexdump(Memory.readInt(args[0]), {
       //                          offset: 0,
       //                          length: 32,
       //                          header: true,
       //                          ansi: true
       //                        }));

       args[0] = ptr("0x100");
   },
   onLeave:function(retval){
       console.log("ret = " + retval.toInt32());
       // retval.replace(ptr("0x1"));
       retval.replace(222);
   }
});
```

0x05 tips

获取app context

```
var currentApplication = Dalvik.use("android.app.ActivityThread").currentApplication();
   var context = currentApplication.getApplicationContext();
```

创建对象示例

```
obj.$new();
```

hook 构造方法

```
obj.$init.implementation = function (){
}
```

实现java接口

https://gist.github.com/oleavr/3ca67a173ff7d207c6b8c3b0ca65a9d8

java接口使用参考,其中X509TrustManager是interface类型.TrustManager为其实现类.manager为实例.

我就成功过这一个接口,其他接口比如Runnable , HostNamerVerifier都没成功.

```
'use strict';

var TrustManager;
var manager;

Java.perform(function () {
 var X509TrustManager = Java.use('javax.net.ssl.X509TrustManager');

 TrustManager = Java.registerClass({
   name: 'com.example.TrustManager',
   implements: [X509TrustManager],
   methods: {
     checkClientTrusted: function (chain, authType) {
       console.log('checkClientTrusted');
```

```
      },
      checkServerTrusted: function (chain, authType) {
        console.log('checkServerTrusted');
      },
      getAcceptedIssuers: function () {
        console.log('getAcceptedIssuers');
        return [];
      }
    }
  });
  manager = TrustManager.$new();
});
```

str int指针操作,有点乱
utf8 string写
`Memory.allocUtf8String(str)`
`var stringVar = Memory.allocUtf8String("string");`
utf8 string读
`Memory.readUtf8String(address[, size = -1])`

int写
`var intVar = ptr("0x100");`
`var intVar = ptr("256");`
int读
`toInt32()`: cast this NativePointer to a signed 32-bit integer

二进制读取
`hexdump(target[, options])`: generate a hexdump from the providedArrayBuffer or _NativePointer_ `target`, optionally with `options` for customizing the output.

0x06 推荐工具和阅读

frida api
https://www.frida.re/docs/javascript-api
中文翻译
https://zhuanlan.kanxue.com/article-342.htm
https://zhuanlan.kanxue.com/article-414.htm

工具推荐
appmon : https://github.com/dpnishant/appmon
droidSSLUnpinning : https://github.com/WooyunDota/DroidSSLUnpinning
objection : https://github.com/sensepost/objection

0x07 reference

---

https://github.com/datatheorem/TrustKit-Android
https://github.com/moxie0/AndroidPinning
https://koz.io/using-frida-on-android-without-root/
https://medium.com/@appmattus/android-security-ssl-pinning-1db8acb6621e
https://developer.android.com/training/articles/security-ssl.html#Pinning
https://developer.android.com/training/articles/security-config.html?hl=zh-cn

点击收藏 | 1 关注 | 3
上一篇：php一句话木马检测绕过研究 下一篇：强网杯拟态防御赛ez_upload...
1. 1 条回复

[3t2uggle](#) 2018-10-24 14:32:16

感谢大佬整理知识，无私分享！！

0 回复Ta

[现在登录](#)

热门节点

[技术文章](#)

[社区小黑板](#)

目录

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)