170918 逆向-jeb2动态调试(ciscn-crack.apk)

原创

<u>奈沙夜影</u> ● 于 2017-09-19 01:25:29 发布 ● 9396 ☆ 收藏 2
 分类专栏: <u>CTF</u>
 版权声明:本文为博主原创文章,遵循 <u>CC 4.0 BY-SA</u>版权协议,转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/whklhhhh/article/details/78025563

版权

CTF 专栏收录该内容

163 篇文章 4 订阅 订阅专栏 1625-5 王子昂 总结《2017年9月18日》 【连续第351天总结】 A. jeb2动态调试Android B.

之前在国赛的时候有一个APKCRACK,当时逆的过程中发现是调用了一个jni方法,so逆出来发现是很复杂的多行处理,最后返回到DEX中的

由于不怎么熟悉so和jni,遂放弃

后来几天看到了WriteUp:

可以看到,程序在取得了用户输入的字符串后,会调用wick.show方法,这个方法会调用jni中的对应函数,该jni函数会开启反调试并给静态 变量A、B赋值success和failed。随后会进入simple.check方法开启验证。 这个验证函数非常长,笔者也没看懂。Simple类中有两个字节数组,一个用于存储输入,把它命名为input;另一个数组初始为空,把它命 名为empty。 使用jeb2的动态调试功能,把断点下到00000A7A函数的返回指令处,在手机中输入随意字符并点击确定,程序会断在返回指令处。 此时查看empty数组的值,发现疑似ASCII码的数字,转换过来就是flag

虽然模拟器+IDA勉强也可以远程调试,但是由于IDA只能处理so文件,所以整个流程并不完整 这次安卓真机终于到手了,正好看到jeb2对动态调试的能力很强,就拿上次没做完的、简单的CRACK入手吧

jeb2的Debug按钮很好找,只不过窗口里并没有像教程那样显示出来IP、端口和进程来Attch

搜了很久的jeb2的动态调试进程,除了i春秋有付费视频没看到内容以外,其他基本都是介绍静态反编译或是一笔带过动态调试 的,没有一个有完整教程

首先要运行adb进行连接 日常报错:

- daemon not running. starting it now on port 5037 * CreateProcess failure, error 2
- failed to start daemon *
 error: cannot connect to daemon

检查5037端口没被占用、重启adb都无效

最后在http://blog.csdn.net/xishuluoye/article/details/17880663找到了解决方法:

解决方法: 1.将本机C:\WINDOWS\System32下的adb.exe文件复制到C:\Windows\SysWOW64下。 2.将本机C:\WINDOWS\System32下的AdbWinApi.dll文件复制到C:\Windows\SysWOW64下。

其中第二条我没找到dll文件所以没做,不过adb也终于连接成功了

然后先是准备将apk文件下载到手机端再安装,然而找了半天也没发现有可写权限的目录 最后同学提醒/sdcard目录一般都可写,另外在adb shell中可以用su获得root权限 传输成功以后又安装失败了┐(╯_╰) ╭ 提示invalid apk file

查了一下似乎跟权限有关, chmod 777了也不好使 查询过程中想起来可以直接install 本地目录的APK, 远程安装

仍然失败,提示

Failure [INSTALL_FAILED_OLDER_SDK]

查了一下,是安卓版本小于APK要求版本导致的......(╯、□')╯ ⌒**┴──┴**你一个逆向题要这么高版本干啥啊!弱鸡二手机受不起啊!

打开源码目录下的AndroidManifest.xml文件, 然后注释掉或者删除掉这行: <uses-sdk android:minSdkVersion="19" />

用APK改之理(AndroidKiller应该也可以)删去apktool.yml下的apkinfo中的version信息后重新编译生成APK即可

再用jeb2重新打开新生成的APK检查确认minSdkVersion不在了方可 刚开始由于APK改之理修改完了以后还需要手动点击保存,以及生成APK的时候似乎不会自动覆盖同名APK,导致重复了好几 次

再adb install就成功了 手机上打开,然后在jeb2中进行attch:

🖫 Project Explorer 🛛 🙀 🗖	🗖 🌻 Bytecode	/ 🌻 Bytecode/ 🛛	🛛 Manifes	t/F 🔛 o	ert <unb< th=""><th> 🔳 arm<unb< th=""><th> 🖸 simj</th><th>ole/So</th><th>🗟 elf<unbo< th=""><th>₱ Process/B</th><th>🅸 Process/L</th><th>. 🏇 Process</th></unbo<></th></unb<></th></unb<>	🔳 arm <unb< th=""><th> 🖸 simj</th><th>ole/So</th><th>🗟 elf<unbo< th=""><th>₱ Process/B</th><th>🅸 Process/L</th><th>. 🏇 Process</th></unbo<></th></unb<>	🖸 simj	ole/So	🗟 elf <unbo< th=""><th>₱ Process/B</th><th>🅸 Process/L</th><th>. 🏇 Process</th></unbo<>	₱ Process/B	🅸 Process/L	. 🏇 Process
✓												
 ApkIDE crack.apk 	00000A54	add-int/lit8	v7, v3, (×01								
✓ I ApkIDE crack.apk	00000A58	aget-byte	v7, v2, v	7								
Manifest	: A60	IT-gu	v/, vii,	AOC								
🔛 Certificate	00000A60	add-int/lit8	v7, v3, 6	×01								
v 🌻 Bytecode	00000A64	aget-byte	v7, v2, v	7								
v 🥔 decompiler	00000A68	1†-ge	v7, v9,	A42								
P ichunqiu	00000A6C	int-to-byte	v0, v6									~
1) simple	00000A6E	const/4	v6, 0	Attac	h							×
v 🗁 Resources	00000A70	goto/16	:6D4	Machin	es / Devi	res						
> 🗁 layout	00000A74	if-eqz	v0, :A70		,							
> 🗁 menu	:A78			Filter: 1	type "Ent	er" to validate						
> 🗁 mipmap-hdpi-v4	00000A78	const/4	v7, 1	Index	Name		Location	Flags	Information			
> 🗁 mipmap-mdpi-v4	00000474	return	v7	0	192.168	3.123.115:5555		Online	U9180			
> 🗁 mipmap-xhdpi-v4	: A7C											
> 🗁 mipmap-xxhdpi-v4	00000A7C	const/4	v7, 0									
> 🗁 mipmap-xxxhdpi-v4	00000A7E	goto	:A/A Processes									
> 🗁 values	00000A80	00000A80 .sparse-switch com.exam				hunqiu_2						
> 🗁 values-v11		0x0 -> :A74		Index	ID	Name					Flags	^
> 🗁 values-v14		0x2F -> :A42 0x41 > :83A		242	14062	hworker/0:1					Tiags	
> 🗁 values-w820dp-v13		0x42 -> :9D0		242	14902	kworker/0.1						
🗸 🗁 Libraries		0x43 -> :9BC		245	14970	kworker/0.0H						_
> 🗁 armeabi		0x44 -> :8AA		244	14975	kworker/u:2						_
✓ 参 VM		0x45 -> :8E2		245	15048	kworker/0:1H						_
梦 Process		0x47 -> :91A		246	15099	/system/bin/dh	срса					
✓ 莎 VM	<			247	15145	kworker/u:5						_
≫ Process	Description	Hex Dump Disassem	bly String	248	15381	com.example.id	hunqiu_2				Dj	-
~ ≫ VM				249	15665	ps						
Structure Process	Switch to i	t by issuing the "u	- 9" comm									~
>	Native libr	aries were found: a	native co	Ontions								
> 3% VM	Unit "Process" (dbug_apk_native) was cre											
> 珍 VM	A command i	A command interpreter for unit "Process" Ususpend all threads										
	Switch to i	t by issuing the "u	se 10" com	Attack	1	关闭						
	Unit "VM" (dbug ank) was create	od bod				_	_				

按WP上的地方下了断点以后,在手机上随便输入内容然后点击Check按钮 jeb2就成功断住了

下面进入另一个大坑,由于jeb2根本没有debug教程,WP上说的也很简略,只是说查看Simple的empty数组的值就知道了

然而怎么查看那玩意儿啊(╯`□′)╯⌒┻━━━

刚开始以为要在console里按gdb的方式来查看,然而无论输入什么VM都只会回显

VM> info -1: An error occurred ("info")

明明info指令是help回显出来的说……

最后终于在窗口中找到VM>locals的查看窗口 打开后两个醒目的数组吸引我打开它:

Name	Type	Value	Extra	
✓ this	Lcom/example/ichungiu	id=830029158872		
~ • A	[B	id=830029158896		
	byte	49	31h	
	byte	50	32h	
	byte	51	33h	
	byte	0	0h	
	byte	0	0h	
	byte	0	0h	
	byte	0	Ob	

	~,~~	~	
	byte	0	0h
	byte	0	0h
	byte	0	0h
	byte	0	Oh
	byte	0	0h
	byte	0	Oh
	byte	0	Oh
	byte	0	Oh
	byte	0	0h
	hvte	0	Oh
Description Log	Threads Breakpoints	Locals	http://blog.csdn.net/whklhhhh

	Name	Туре	Value	Extra			
	 this Lcom/example/ichunqiu 		id=830029158872				
	> • A [B		id=830029158896				
	∽ • B	[B	id=830029158992				
		byte	99	63h			
		byte	108	6Ch			
		byte	111	6Fh			
		byte	53	35h			
		byte	101	65h			
		byte	114	72h			
		byte	0	Oh			
		byte	0	Oh			
		byte	0	Oh			
		byte	0	0h			
		byte	0	0h			
		byte	0	0h			
		byte	0	0h			
		byte	0	0h			
		byte	0	0h			
		byte	0	0h			
		byte	0	0h			
		byte	0	0h			
		byte	0	0h			
		byte	0	0h			
		hvte	0	0h			
Description Log Threads Breakpoints Locals							
E	🖳 Logger 🕼 Console 🛿 http://blog.csdn.net/whklhhhh						

this->A很明显是我输入测试的123的ASCI

那么this->B应该就是flag的ASCII了,拖下来手动转换一下就能得到flag了

另外,调试断下来以后在源码界面也可以得到更清晰直接的信息:

```
blic class simple {
  private byte[] A;
  private byte[] B;
  public simple(String arg5) {
     int v3 = 50;
     super();
     this.A = new byte[v3];
     this.B = arg5.getBytes();
     int v0;
      for(v0 = 0; v0 < this.B.length; ++v0) {</pre>
         this.A[v0] = this.B[v0];
      }
                             B - [B
      this.B = new byte[v3]; array@830029158992 (type=[B)
  }
                             [99(63h), 108(6Ch), 111(6Fh), 53(35h), 101(65h), 114(72h), 0, 0, 0,
  public boolean check() {
      byte v11 = 57;
     byte v9 = 48;
     byte[] v2 = new byte[20
     v2[0] = 47;
     v2[1] = v9;
     v2[2] = 81;
     v2[3] = 82;
      v2[4] = 79;
                              <
                                                                                                            >
     v2[5] = 54;
     v2[6] = 77:
```

很明显,simple方法对this.A和this.B进行了循环比较,这个时候鼠标放上去就能看到B的内容了

至此,很简单的动态调试、查看本地变量过程就完成了~

C.明日计划 国赛WP