2015阿里&看雪移动安全挑战赛-第二题



Scoronepion ● 于 2016-05-07 21:26:26 发布 ● 2808 ☆ 收藏 2
分类专栏: 安卓逆向 文章标签: 安卓逆向 移动安全 移动安全挑战赛 writeup
版权声明:本文为博主原创文章,遵循 CC 4.0 BY-SA 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/scoronepion/article/details/51340137
版权



安卓逆向 专栏收录该内容

2篇文章0订阅

订阅专栏

2015阿里&看雪移动安全挑战赛-第二题

题目传送门: AliCrackme

网上已经有很多writeup,我也是按照乌云上的2015移动安全挑战赛(阿里&看雪主办)全程回顾的基本思路来想的。但作为一个新手,就算照着教程来做也会踩到很多坑。所以我想把自己解题过程中遇到的一些细节问题跟大家分享一下。

文章中提到的环境配置是按照看雪论坛非虫的《Android软件安全与逆向分析》配置的,这本书很棒,讲的很详细。在这里安利 一波。

0x01

app安装完后长这样:





输入错误的密码会提示校验码错误。我们先看看这次日志会不会有什么输出。

com.yaotong.crackme	yaotong	SecurityCheck Started
com.yaotong.crackme	yaotong	SecurityCheck Started
com.yaotong.crackme	Choreographer	Skipped 186 frames! The application may be doing too much work on its main t \square
		hread.

可以发现,每次输入密码提交后,日志总会输出 SecurityCheck Started...,结合上一题的经历,我们能够从代码中发现日志输出语句。

那么我们就来看看从jar文件中能够发现什么。

0x02

```
public void onClick(View paramAnonymousView)
{
   String str = <u>MainActivity</u>.this.inputCode.getText().toString();
   if (<u>MainActivity</u>.this.securityCheck(str))
   {
     Intent localIntent = new Intent(<u>MainActivity</u>.this, <u>ResultActivity</u>.class);
     <u>MainActivity</u>.this.startActivity(localIntent);
     return;
   }
   Toast.makeText(<u>MainActivity</u>.this.getApplicationContext(), "验证码校验失败", 0).show();
}
```

从代码中可以看到,校验成功与否是由 securityCheck()这一方法的返回值决定的。而在它的Java代码中,我没有找到这一方法,也没有找到日志输出的语句。后来看乌云上的资料,在反编译后的lib文件夹下找到了 libcrackme.so 文件。根据提示,用 IDA打开了这个文件。

(注:某些版本的IDA不支持调试so文件,这里我用的是6.6版本)

text:000011A8	;	= S U B	RO	UΤ	ΙN	Е	
text:000011A8							
text:000011A8							
text:000011A8		EXPORT .	Java_	com	yao	tor	ng_crackme_MainActivity_securityCheck
text:000011A8	Java_com_yaotong	j_crackme	e_Mai	InAct	tivi	ty_	_securityCheck
text:000011A8							
text:000011A8	var_20	= -0x20					
text:000011A8	var_1C	= -0x1C					
text:000011A8							
text:000011A8		STMFD	SP!,	, {R/	4-R7	,R1	11,LR}
text:000011AC		SUB	SP,	SP,	#8		
text:000011B0		MOV	R5,	RØ			
text:000011B4		LDR	RØ,	=(_(LOB	AL_	_OFFSET_TABLE 0x11C8)
text:000011B8		LDR	R6,	=(u	1k_6	29	0 - 0x5FBC)
text:000011BC		MOV	R4,	R2	_		
text:000011C0		ADD	RØ,	PC,	RØ	;	GLOBAL OFFSET TABLE
text:000011C4		ADD	RØ,	R6,	RØ	; ī	Jnk_6290

发现了 securityCheck() 方法,继续往下找,我们在0x1284处发现了日志输出函数 android_log_print

.text:00001284	BL	android log print
.text:00001288	LDR	R 0, [R5]
.text:0000128C	MOV	R1, R4
.text:00001290	MOV	R2, #0
.text:00001294	LDR	R3, [R0,#0x2A4]
.text:00001298	MOV	R0, R5
.text:0000129C	BLX	R3
.text:000012A0	LDR	R1, =(off 628C - 0x5FBC)
.text:000012A4	LDR	R2, [R1,R7] ; off 628C
.text:000012A8		
.text:000012A8 10	c 12A8	; CODE XREF: Java com yaotong crackme MainActivity securityCheck+120_
.text:000012A8	LDRB	R3, [R2]

0x03

那么, securityCheck 的执行流程是怎样的?从乌云上的文章解释得很详细:

在securityCheck这个方法调用前,在init_array段和JNI_Onload函数里程序都做了些处理,而在securityCheck方法的最后有一个判断,将用 户输入和wojiushidaan做比较。尝试直接输入wojiushidaan,发现密码错误,因此可以猜测前面一大段逻辑的作用就是会把这个最终的字符 串改掉。此时的思路是只需知道最终判断时候这个wojiushidaan地址上的变换后的值就行了。尝试使用IDA调试发现一旦attach上去,整个程 序就退出,想必一定是在之前的代码中有反调试的代码。

既然我们要得到变换后的值,我们就可以借助日志输出函数将我们想要得到的值输出出来就好了,因此,我们需要对so文件进行一下修改。

- 将从0x1284到0x129C处都用NOP改写(NOP为空操作)
- 在0x12AC处调用 android_log_print 函数
- 为了不影响R1的值,把0x12A0处的R1改成R3
- 将0x12A4处的R1改成R3
- 将0x12A8处指令改成 MOV R0,#4

因为so为二进制文件,所以我们修改的也是二进制。

- NOP对应的二进制值为00 00 A0 E1
- android_log_print 对应的二进制值为88 FF FF EB
- 0x12A0处的值应改为60 30 9F E5
- 0x12A4处的值应改为07 20 93 E7
- MOV R0,#4 的值为04 00 A0 E3

在IDA中,选中某一行,进入Hex View选项卡就可以看到该行的二进制数值。按下F2键可以进行编辑,编辑好后再按下F2键可以进行保存。此时回到IDA View界面就可以看到程序已经改好了。但注意,IDA中的修改只是保存在了它自己的数据库里,并没有保存到so文件里。所以为了能够直接编辑so文件,我们可以使用UltraEdit来进行编辑。在IDA中确认无误后可将修改内容通过UE保存。

改之前代码:

	IDA View-A 🛛 🛛	🔼 Structures		Enums	×	2	Imports	×]	Exports	×	
	.text:0000126C loc_126C	:		; CODE XRE	F: Java_c	om_yaoto	ng_crack	ne_MainAc	tivity	securityChec	:k+80îj	
→•	.text:0000126C	LDR R	0, =(_GLOBAL_	OFFSET_TABLE	- 0x127	3)						
•	.text:00001270	ADD R	7, PC, R0 ; _	GLOBAL_OFFSE	T_TABLE_							
•	.text:00001274	ADD R	0, R6, R7	_								
•	.text:00001278	ADD R	1, R0, #0x74	1	Hex View-1						8	
•	.text:0000127C	ADD R	2, R0, #0xDC	E								
•	.text:00001280	MOV R	0, #4		00001244	00 00 8	F E0 00 2	20 81 E0	AC 10	9F E5 00 70	86 E0 🔺	
•	.text: <mark>00001284</mark>	BL _	_android_log_	print 🛛	00001254	00 30 8	1 E0 DC 0	00 87 E2	19 10	A0 E3 A3 04	00 EB	
•	.text:00001288	LDR R	0, [R5]		00001264	01 00 A	0 E3 CA 0	00 C7 E5	90 00	9F E5 00 70	8F E0	
•	.text:0000128C	MOV R	1, R4		00001274	07 00 8	6 E0 74 1	0 80 E2	DC 20	80 E2 04 00	A0 E3	
•	.text:00001290	MOV R	2, #0		00001284	92 FF F	F EB 00 0	00 95 E5	04 10	A0 E1 00 20	AØ E3	
•	.text:00001294	LDR R	3, [R0,#0x2A4]	00001294	A4 32 9	0 E5 05 0	00 A0 E1	33 FF	2F E1 60 10	9F E5	
•	.text:00001298	MOV R	0, R5		00001284 00	00000000	001284 .Ta	va com vao	tong cr	ackma MainActiv	vity sec -	
	.text:0000129C	BLX R	3					va_oom_yaa		doxinc_ndrinkoor	101_000 +	
	.text:000012A0	LDR R	1, =(off_628C	- 0x5FBC)	•						P	
•	.text:000012A4	LDR R	2, [R1,R7] ;	off_628C 👘								
	.text:000012A8											
	.text:000012A8 loc_12A8			; CODE XRE	F: Java_c	om_yaoto	ng_crack	ne_MainAc	tivity_	_securityChec	k+120jj	
	.text:000012A8	LDRB R	3, [R2]									
	.text:000012AC	LDRB R	1, [R0]									
	.text:000012B0	CMP R	3, R1									
	.text:000012B4	BNE 1	0C_12D0									
+	00001284 000000000001284:	Java_com_yaotong_cr	ackme_MainActiv	vity_securityCh	neck+DC							

改之后代码:

	IDA View-A 🛛 🔼	Structur	es 🛛 🗶	Enum s	×	2	Imports	×	P	Exports	×
[<u> </u>	.text:00001268	STRB	R0, [R7,	#0xCA]							
- i - i	.text:0000126C										
	.text:0000126C loc_126C			; CODE XRE	EF: Java_co	m_yaot	ong_crackme	_MainActi	vity_se	curityCheck	+80îj
*•	.text:0000126C	LDR	R0, =(_G	LOBAL_OFFSET_TABLE	E 0x1278)					
•	.text:00001270	ADD	R7, PC,	R0 ; _GLOBAL_OFFSE	T_TABLE_						
•	.text:00001274	ADD	RØ, Ró,	R7							
•	.text:00001278	ADD	R1, R0,	#0x74	Hex Viev	/-1					
•	.text:0000127C	ADD	R2, R0,	#ØxDC							
•	.text:00001280	MOV	RØ, #4		000012	50 00	70 86 E0 0	0 30 81 E) DC 00	0 87 E2 19 1	10 A0 E3 🔺
•	.text:00001284	NOP			000012	60 A3	04 00 EB 0 ⁻	1 00 A0 E3) CA 00	0 C7 E5 90 (30 9F E5 🔤 📄
•	.text:00001288	NOP			000012	70 00	70 8F E0 0	7 00 86 EQ) 74 10	0 80 E2 DC 2	20 80 E2
•	.text:0000128C	NOP			000012	80 04	00 A0 E3 0	0 00 A0 E1	00 00	0 AO E1 OO (30 A0 E1
•	.text:00001290	NOP			000012	<mark>96</mark> 88	00 A0 E1 0	0 00 A0 E1	00 00	0 AO E1 OO (JO AO E1
•	.text:00001294	NOP			000012	A0 60	30 9F E5 0	7 20 93 E7	04 00	0 A0 E3 88 I	F FF EB
•	.text:00001298	NOP									
•	.text:0000129C	NOP			0000125	0 00000	0000001290:	Java_com_y	actong_c	rackme_MainAc	tivity_sec +
	.text:000012A0	LDR	R3, =(of	f_628C - 0x5FBC)		111					•
•	.text:000012A4	LDR	R2, [R3,	R7] ; off_628C		_			_		
	.text:000012A8										
	.text:000012A8 loc_12A8			; CODE XRE	EF: Java_co	m_yaot	ong_crackme	_MainActi	vity_se	curityCheck	+120 j j
	.text:000012A8	MOV	RØ, #4								
	.text:000012AC	BL	androi	d_log_print							
	.text:000012B0	CMP	R3, R1								
L r -	.text:000012B4	BNE	1oc_12D0								
	.text:000012B8	ADD	R2, R2,	#1							
	.text:000012BC	ADD	R0, R0,	#1							

0x04

so文件修改完保存好后,将它与原来的反编译文件一起重新打包签名,生成一个新apk。然后将apk重新安装到虚拟机上。运行 后观察输出日志。

com.android.phone	dalvikvm	GC_CONCURRENT freed 387K, 5% free 11369K/11847K, paused 105ms+22ms, total 342 \square
		ms
com.yaotong.crackme	yaotong	aiyou, bucuoo
system_process	dalvikvm	Jit: resizing JitTable from 4096 to 8192

可以发现,日志输出了一个字符串 aiyou, bucuoo, 经验证, 该字符串为正确的密码。





Congratulations!!!You Win!!

 \leftarrow