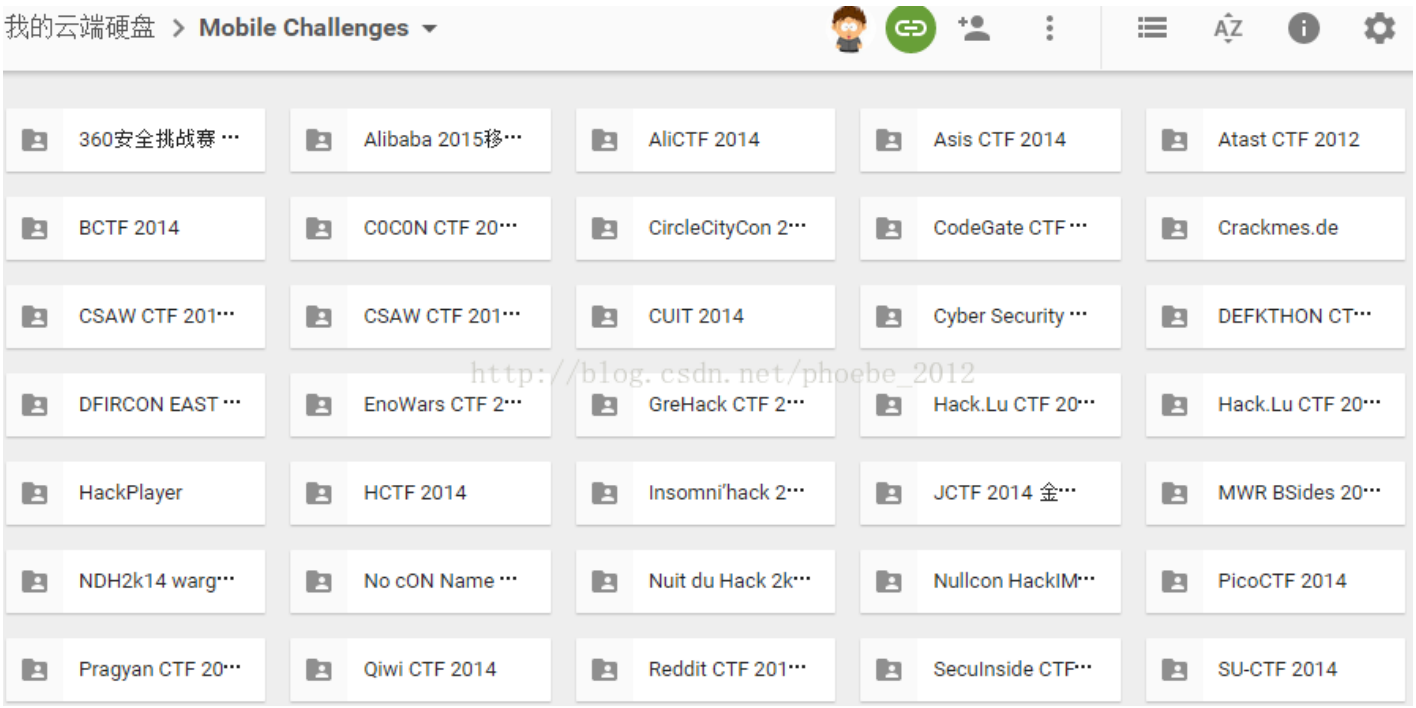
各类移动安全竞赛题/部分writeup收集与整理

原创

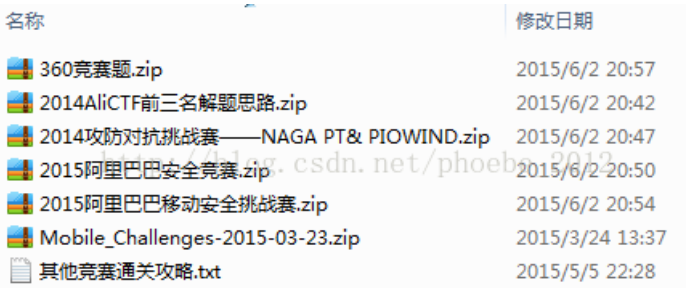
phoebe_2012 于 2015-06-03 20:44:12 发布 1768 收藏 1
分类专栏： android 文章标签： 移动安全 竞赛题 writeup

各类移动安全竞赛题/部分writeup收集与整理

小编偶然发现在google云端网盘上Mobile Challenges有各个网友上传的关于移动安全的题目https://drive.google.com/folderview?id=0B7rtSe_PH_fTWDQ0RC1DeWVoVUE&usp=sharing，有需要的可下载。



还收集了部分竞赛中一些牛人写的解题思路，下载网址如下：http://pan.baidu.com/s/1ntrDgc5 密码：byis



以下是小编参加2015阿里组织的移动安全挑战赛时写的第二题解题思路，那时刚学android逆向，比较菜，各位大神请绕过~

方法一：

运行程序后没有log信息，又看了下源码，它加载了crackme.so动态链接库，调用了其中的函数securityCheck(str)，根据该函数的返回值（0或1）来判断是否输入正确。

首先进行静态分析crackme.so，载入IDA，查看securityCheck函数的源码：

```
signed int __fastcall Java_com_yaotong_crackme_MainActivity_securityCheck(int a1, int a2, int a3)
{
  int v3; // r5@1
  int v4; // r4@1
  int v5; // r0@5
  char *v6; // r2@5
  int v7; // r3@6
  signed int v8; // r1@7

  v3 = a1;
  v4 = a3;
  if ( !byte_6359 )
  {
    sub_2494((int)&unk_6304, 8, (int)&unk_446B, (int)&unk_4468, 2u, 7);
    byte_6359 = 1;
  }
  if ( !unk_635A )
  {
    sub_24F4(&unk_636C, 25, &unk_4530, &unk_4474);
    unk_635A = 1;
  }
  _android_log_print(4, &unk_6304, &unk_636C);
  v5 = (*(int (__fastcall **)(int, int, _DWORD))(*(_DWORD *)v3 + 676))(v3, v4, 0);
  v6 = off_628C;
  while ( 1 )
  {
    v7 = (unsigned __int8)*v6;
    if ( v7 != *(_BYTE *)v5 )
      break;
    ++v6;
    ++v5;
    v8 = 1;
    if ( !v7 )
      return v8;
  }
  return 0;
}
```


得知字符串v6所存的就是密码。查看偏移628c处所存的字符串，是“wojiushidaan”。输入看看，错误。好吧，我也觉得没有那么简单。估计是程序运行后把字符串改了。

```
.data:0000628C off_628C        DCD aWojiushidaan        ; DATA XREF: Java_com_yaotong_crackme_MainActivity
.data:0000628C                                          ; .text:off_1308↑o
.data:0000628C ; .data         ends                     ; "wojiushidaan"
```


刚学了android动态调试（http://www.52pojie.cn/thread-293648-1-1.html），就试试看，在JNI_OnLoad函数上下断点。

(1) 执行android_server：adb shell /data/local/tmp/android_server

(2) 端口转发adb forward tcp:23946 tcp:23946

(3) 调试模式启动程序adb shell am start -D -n com.yaotong.crackme/.MainActivity

(4) IDA附加

(5) 设置调试选项

(6) F9运行

(7) 执行jdb -connectcom.sun.jdi.SocketAttach:hostname=127.0.0.1,port=8700

(8) 静态找到目标函数对应所在模块的偏移地址

JNI_OnLoad函数：0x00001B9C

Java_com_yaotong_crackme_MainActivity_securityCheck：0x000011A8

aWojiushidaan：0x00004450

(9) Ctrl+S找到libcrackme.so基地址0x4924C000，分别与上述三个地址相加得到最终地址。

(10) G跳转至两个函数地址，然后下断点

(11) F9运行

(12) 断下，进行单步调试F8，同时查看0x49250450处的字符串。

```
49250450  77 6F 6A 69 75 73 68 69  64 61 61 6E 00 4C 37 39  wojiushidaan.L79
49250460  00 F4 51 0D 75 AC 53 00  33 66 00 41 3C 5B 0B 5B  ..Q.u.S.3f.A<[.[
49250470  01 53 9F 00 47 56 4D 00  73 21 23 4C 00 42 4D 54  .S..GVM.s!#L.BMT
49250480  00 A9 B5 B8 B7 C1 55 00  6D 37 00 01 49 34 72 03  ......U.m7..I4r.
49250490  4E B8 00 50 53 00 39 58  2C 2B 28 58 3D 21 29 CE  N..PS.9X,+(X=!).
```

当单步调试JNI_OnLoad函数到某一步时，发现字符串变为“aiyou,bucuoo”。

```
libcrackme.so:4924DC54 SUB             R0, R11, #-var_20
libcrackme.so:4924DC58 BLX             R7
libcrackme.so:4924DC5C BL              unk_4924D7F4
PC libcrackme.so:4924DC60 LDR          R0, [R4]
libcrackme.so:4924DC64 MOV             R6, #4
libcrackme.so:4924DC68 MOV             R1, R5
libcrackme.so:4924DC6C ORR             R6, R6, #0x10000
libcrackme.so:4924DC70 MOV             R2, R6
libcrackme.so:4924DC74 LDR             R3, [R0,#0x18]
libcrackme.so:4924DC78 MOV             R0, R4
libcrackme.so:4924DC7C BLX             R3
libcrackme.so:4924DC80 CMP             R0, #0
```

```
49250410  48 00 90 E5 08 80 BD E8  08 40 2D E9 09 FF FF EB  H....■...@-.....
49250420  4C 30 90 E5 07 20 D3 E5  02 01 83 E0 08 00 80 E2  L0... ·........■.
49250430  08 80 BD E8 08 40 2D E9  3D F3 FF EB 08 40 2D E9  .■...@-.=....@-.
49250440  3B F3 FF EB 00 00 00 00  00 00 00 00 00 00 00 00  ;...............
49250450  61 69 79 6F 75 2C 62 75  63 75 6F 6F 00 4C 37 39  aiyou,bucuoo.L79
49250460  00 F4 51 0D 75 AC 53 00  33 66 00 41 3C 5B 0B 5B  ..Q.u.S.3f.A<[.[
49250470  01 53 9F 00 47 56 4D 00  73 21 23 4C 00 42 4D 54  .S..GVM.s!#L.BMT
49250480  00 A9 B5 B8 B7 C1 55 00  6D 37 00 01 49 34 72 03  ......U.m7..I4r.
49250490  4E B8 00 50 53 00 39 58  2C 2B 28 58 3D 21 29 CE  N..PS.9X,+(X=!).
```

结果：aiyou,bucuoo

方法二：

dump内存。

程序运行在手机上（经测试，模拟器上貌似不行）。

通过dd命令，将内存里的数据拷贝出来。

dd if=/proc/6821/mem of=/sdcard/mem bs=1skip= 1074085888 count=20480

dd if=/proc/6821/mem of=/sdcard/mem bs=1skip=1336799232 count=28672

```
root@android:/ # cat /proc/6821/maps | grep libcrackme.so
cat /proc/6821/maps | grep libcrackme.so
4fadf000-4fae0000 r-xp 00000000 b3:13 2361       /data/app-lib/com.yaotong.crack
me-1/libcrackme.so
4fae0000-4fae1000 rwxp 00001000 b3:13 2361       /data/app-lib/com.yaotong.crack
me-1/libcrackme.so
4fae1000-4fae3000 r-xp 00002000 b3:13 2361       /data/app-lib/com.yaotong.crack
me-1/libcrackme.so
4fae3000-4fae4000 rwxp 00004000 b3:13 2361       /data/app-lib/com.yaotong.crack
me-1/libcrackme.so
4fae4000-4fae5000 r--p 00004000 b3:13 2361       /data/app-lib/com.yaotong.crack
me-1/libcrackme.so
4fae5000-4fae6000 rw-p 00005000 b3:13 2361       /data/app-lib/com.yaotong.crack
me-1/libcrackme.so
root@android:/ # dd if=/proc/6821/mem of=/sdcard/mem bs=1 skip=1336799232 count=
28672
/mem bs=1 skip=1336799232 count=28672                                          <
28672+0 records in
28672+0 records out
28672 bytes transferred in 1.021 secs (28082 bytes/sec)
root@android:/ #
```


mem文件用IDA打开，查看0x00004450处的值。

```
00004450  61 69 79 6F 75 2C 62 75  63 75 6F 6F 00 4C 37 39  aiyou,bucuoo.L79
```

方法三：

执行android_server：adb shell /data/local/tmp/android_server

端口转发adb forward tcp:23946 tcp:23946

在模拟器上运行程序

ps找到程序pid

命令：kill -19 <pid> 可以让进程暂停

IDA附加

libcrackme.so基址为0x4914C000

aWojiushidaan：0x00004450

查看0x49150450处的内存

```
49150450  61 69 79 6F 75 2C 62 75  63 75 6F 6F 00 4C 37 39  aiyou,bucuoo.L79
```

第三题的解决方案：

(1) lndroid(LoCCS实验室开发的工具)貌似要在google手机编译后才能运行成功；

(2) Zjdroid修复一下再内存dump

(3) 过反调试 j_j_ptrace、dump内存、修复dex、修复bakesmali、修复dextojar 或 JEB查看，最后应用源代码分析

最后给大家推荐一些优秀站点：

[1] http://bbs.pediy.com/

[2] http://www.52pojie.cn/

[3] http://blog.dutsec.cn/

[4] http://l-team.org/

[5] http://le4f.net/