2017年陕西省网络空间安全技术大赛·Mobile T2

原创

迷棱 于 2018-03-12 16:23:52 发布 493 收藏
分类专栏： CTF 安卓逆向

本文链接：https://blog.csdn.net/guchenjun789/article/details/79527768

0x00The Marauder's Map

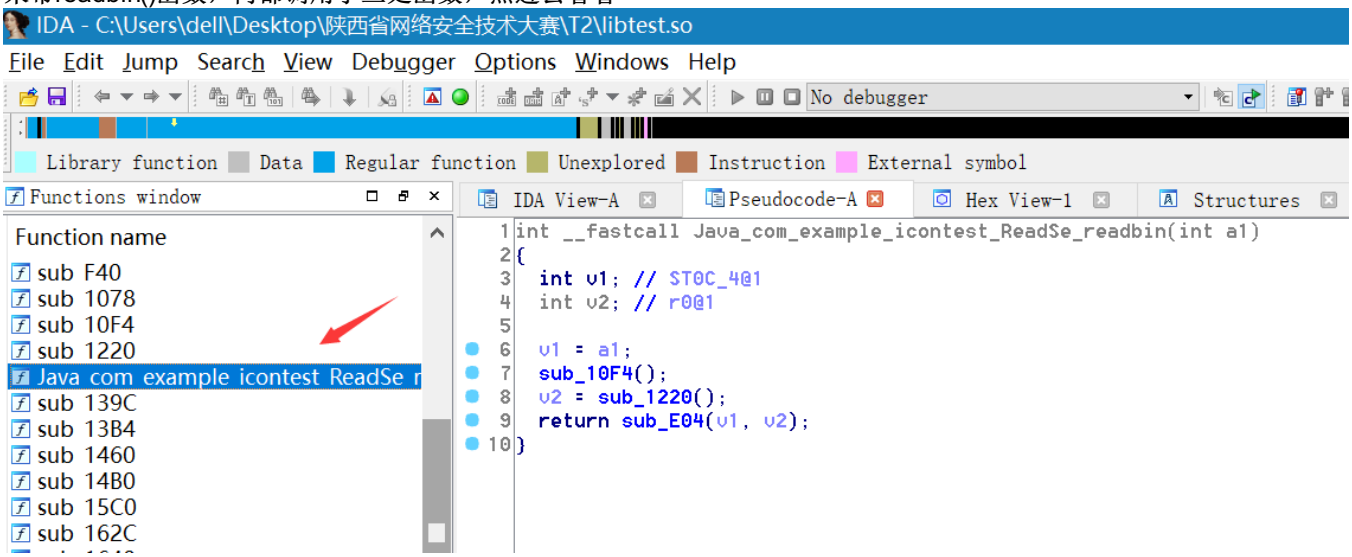
apk链接: https://pan.baidu.com/s/1KxRb7NbR8-z5_rgD4ADl9Q 密码: y7sc

0x01Java层分析

略，参考:https://www.52pojie.cn/thread-603169-1-1.html

0x02Native层分析

1. **用32位IDA打开test.so文件**
   来带readbin()函数，内部调用了三处函数，点进去看看



2. **来到第一个sub_10F4()**

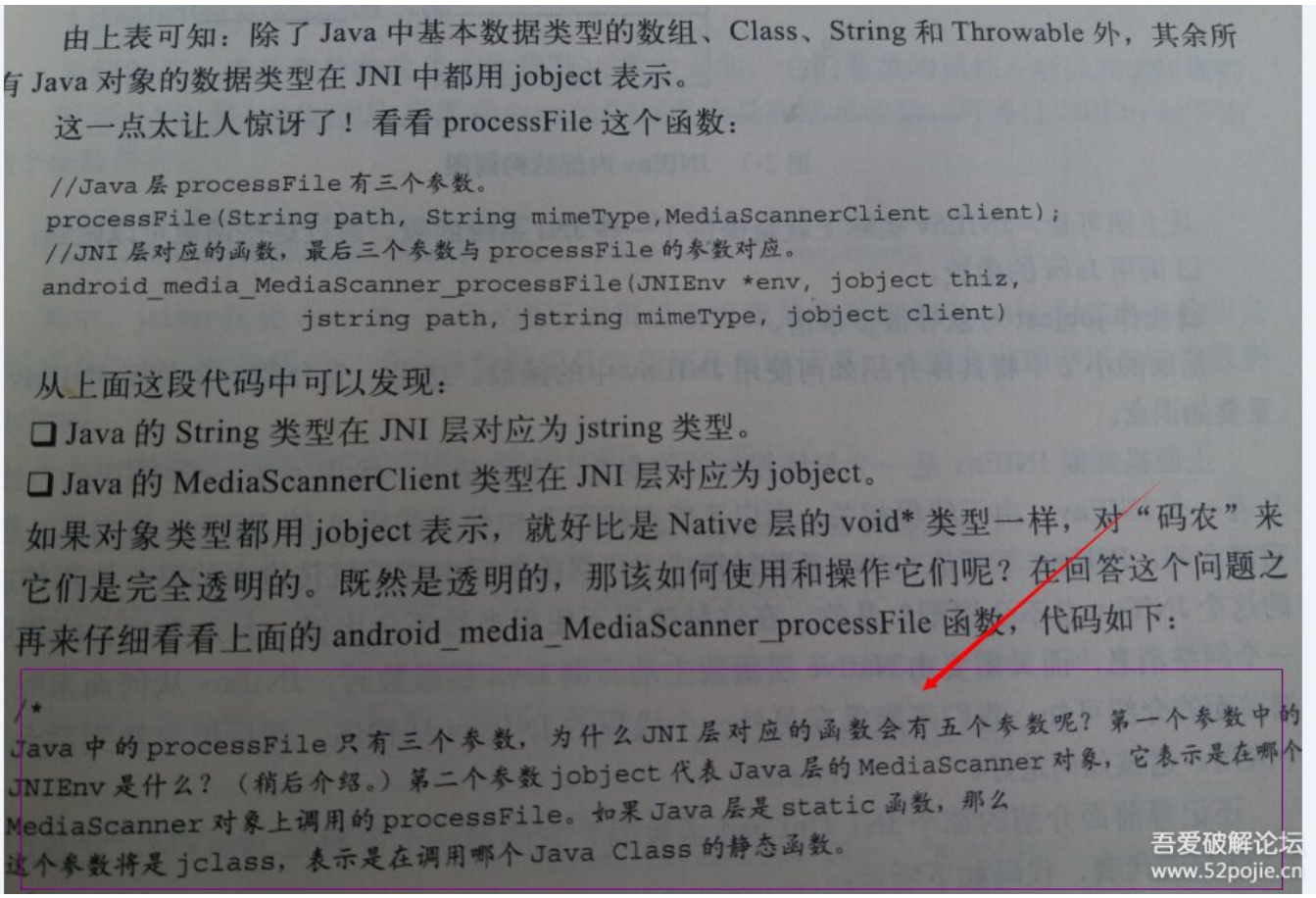
```
1  void *__fastcall sub_10F4(int a1, int a2)
2  {
3    int v2; // ST00_4@1
4    int v3; // ST10_4@1
5    int v4; // ST14_4@1
6    int v5; // ST18_4@1
7    int v7; // [sp+4h] [bp-28h]@1
8    void *dest; // [sp+Ch] [bp-20h]@1
9    int v9; // [sp+1Ch] [bp-10h]@1
10   int n; // [sp+20h] [bp-Ch]@1
11   void *src; // [sp+24h] [bp-8h]@1
12
13   v7 = a1;
14   v2 = a2;
15   dest = 0;
16   v3 = sub_CC4();
17   v4 = sub_E04(v7, "utf-8");
18   v5 = sub_D5C(v7, v3, "getBytes", "(Ljava/lang/String;)[B");
19   v9 = sub_DA8(v7, v2, v5, v4);
20   n = sub_E40(v7, v9);
21   src = (void *)sub_EB8(v7, v9, 0);
22   if ( n > 0 )
23   {
24     dest = malloc(n + 1);
25     memcpy(dest, src, n);
26     *((_BYTE *)dest + n) = 0;
27   }
28   sub_EFC(v7, v9, src, 0);
29   return dest;
30 }
```

发现还是很多函数的调用，点进sub_CC4()进去看看

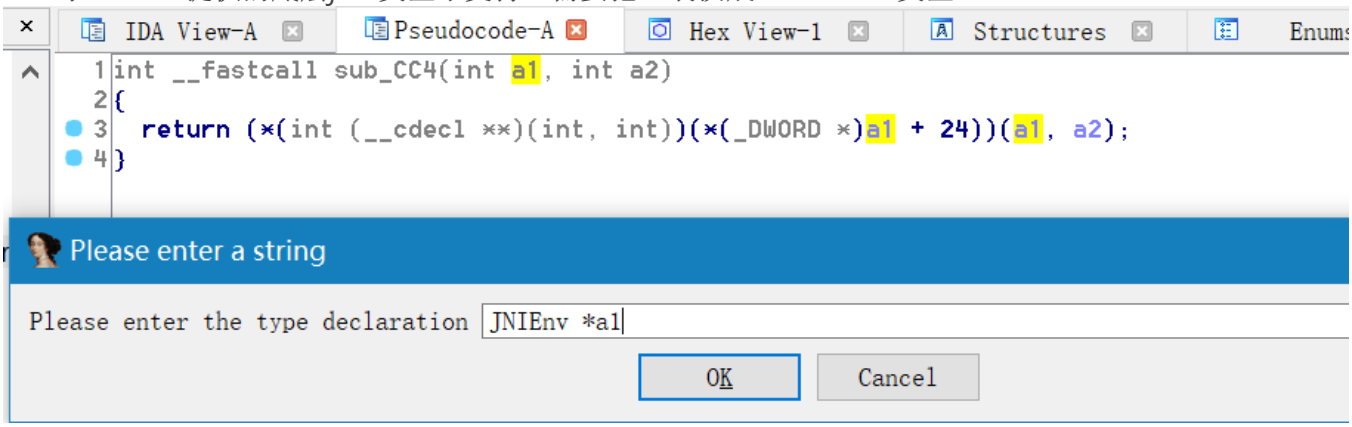
IDA View-A | Pseudocode-A | Hex View-1 | Structures

```
1  int __fastcall sub_CC4(int a1, int a2)
2  {
3    return (*(int (__cdecl **)(int, int))(*(_DWORD *)a1 + 24))(a1, a2);
4  }
```

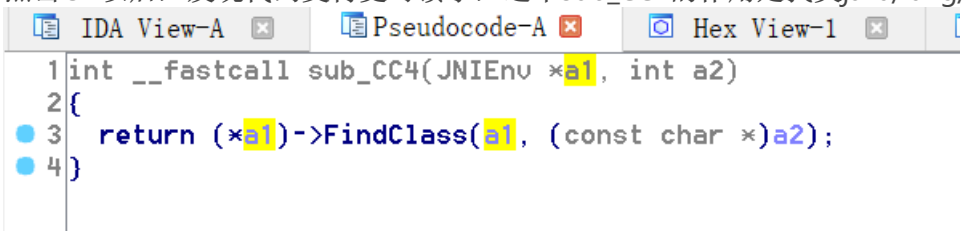
参考资料:https://www.52pojie.cn/thread-603169-1-1.html



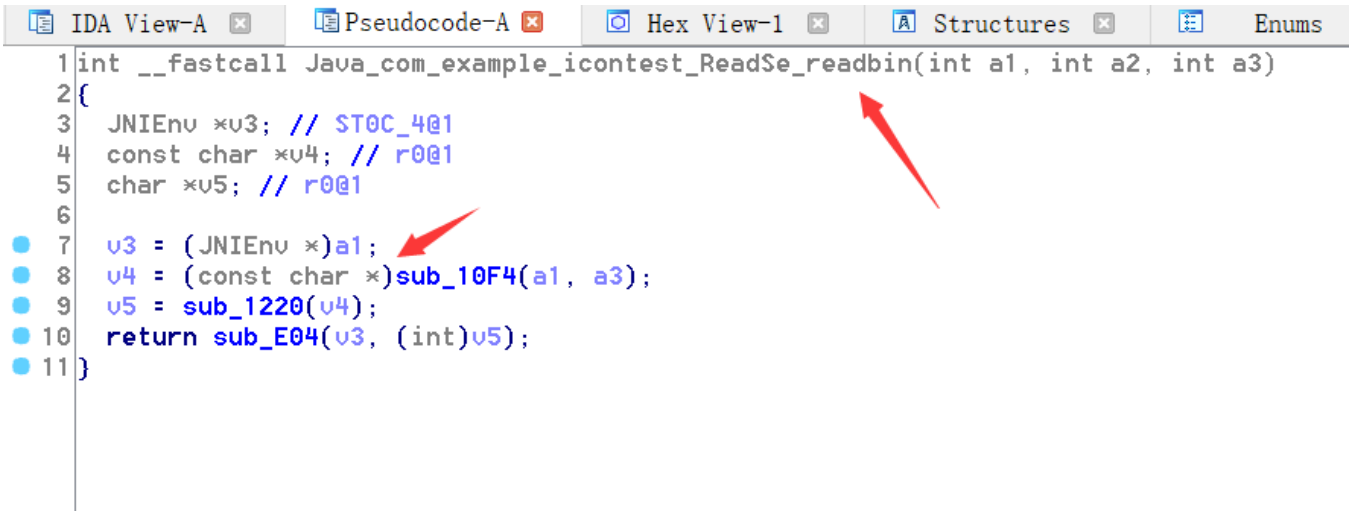
IDA对Android提供的底层java类型不支持，需要把a1装换成JNIEnv *a1 类型



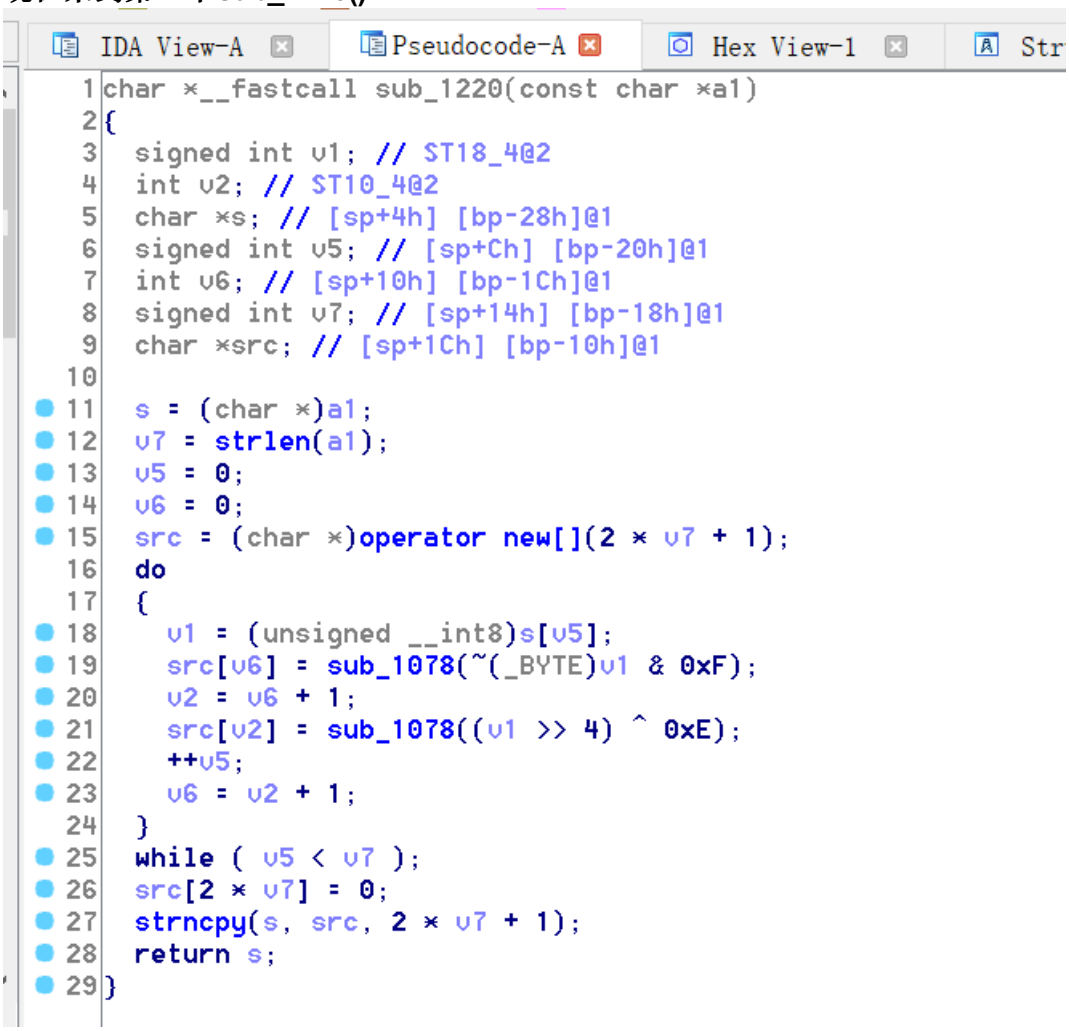
点击OK以后，发现代码变得更可读了，这个sub_CC4的作用是找类java/lang/String



返回上一层，在sub_10F4对其他几个函数第一个参数a1同样修改JNIEnv *a1，看看这些函数在干嘛
改完以后再返回上一层的 readbin()，再点进sub_1220()，然后退出来，发现代码已经变化了
sub_10F4前面多了(const char *)，于是我们猜测sub_10F4()是将jstring转成char*



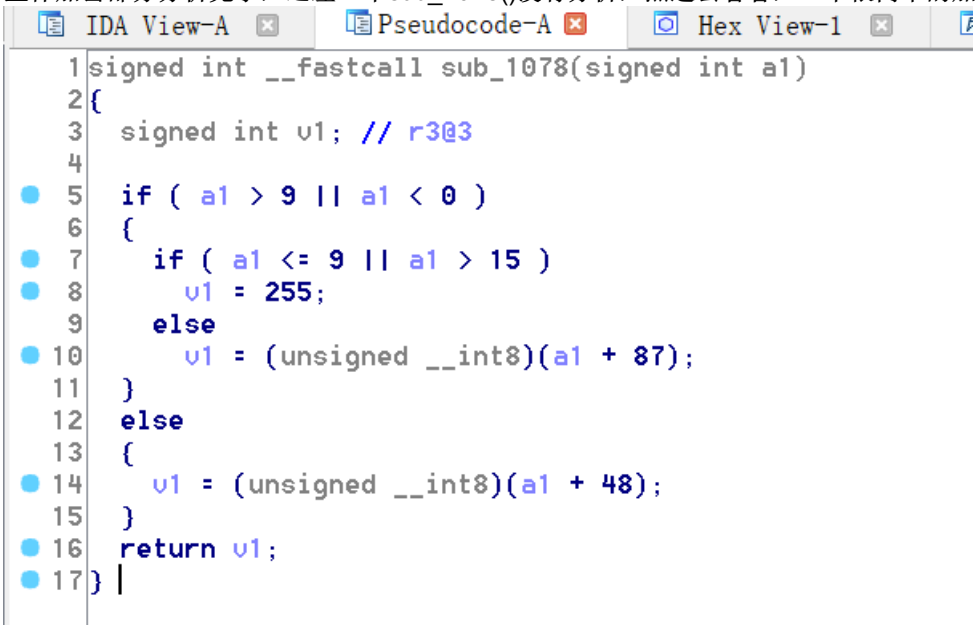
3. **现在来到第二个sub_1220()**



贴上全部代码，方便查看，并做了注释

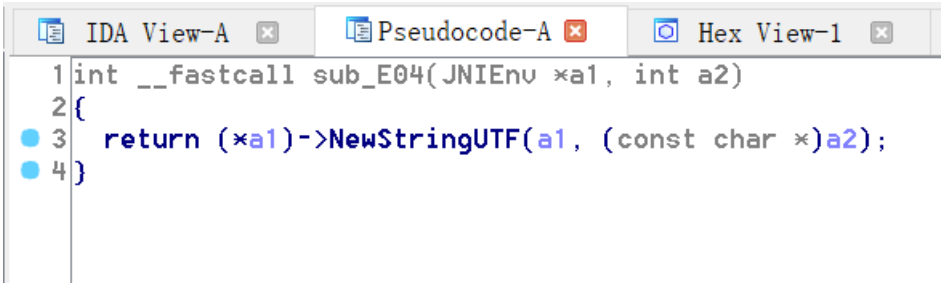
```
char *__fastcall sub_1220(const char *a1)
{
  signed int v1; // ST18_4@2
  int v2; // ST10_4@2
  char *s; // [sp+4h] [bp-28h]@1
  signed int v5; // [sp+Ch] [bp-20h]@1
  int v6; // [sp+10h] [bp-1Ch]@1
  signed int v7; // [sp+14h] [bp-18h]@1
  char *src; // [sp+1Ch] [bp-10h]@1

  s = (char *)a1;                           //a1是我在app中输入的字符数组
  v7 = strlen(a1);
  v5 = 0;
  v6 = 0;
  src = (char *)operator new[](2 * v7 + 1); //分配字符数组长度*2+1长度的内存给src
  do
  {
    v1 = (unsigned __int8)s[v5];            //v1 = s[0]、s[1]、s[2]...
    src[v6] = sub_1078(~(_BYTE)v1 & 0xF);   //src[0] = func(...)、src[2] =
    v2 = v6 + 1;                            //v2 = 1、3
    src[v2] = sub_1078((v1 >> 4) ^ 0xE);    //src[1] = func(...)、src[3] =
    ++v5;                                   //v5 = 1、2
    v6 = v2 + 1;                            //v6 = 2、4
  }
  while ( v5 < v7 );                        //执行字符数组长度次循环0~n-1
  src[2 * v7] = 0;
  strncpy(s, src, 2 * v7 + 1);              //将操作完成的src内容赋值给s,然后返回s
  return s;
}
```

主体加密部分分析完了，还差一个sub_1078()没有分析，点进去看看，一个很简单的加密逻辑，一目了然



4. **来到第三个sub_E04()**

   将int a1改成JNIEnv *a1后，可以明白这步是将参数a2重新转为 jstring类型并返回



5. **逻辑整理**
   sub_10F4()将jstring转成char*
   sub_1220()加密算法实现部分
   sub_E04()将char*转成jstring

0x04 C语言代码实现

Part I：逻辑构想

已知加密算法和加密后的密文，求未加密前的明文?

首先看sub_1078()，这个加密function是不允许我们逆向知道return值去找param的，因为它是范围判断，然后有一部分赋值到255（固定值），意味着不可能找到传参了。而上一篇T1却不一样，它分别是0和1、2和3...这样的交换位置，固逆向通过结果返回值找传参是可行的。

由于这里都是ascii操作的，而ascii范围又在0~255之间，

在do while循环内每次都给src连续两位赋值，即src[0] = ..、src[1] = ..然后是src[2] = ..、src[3] = ..，

并且src赋值给s后就是跟java层的paramString传参通过equal比较的。

因此，通过ascii去遍历，经过func，如果两个str[i]和str[i+1]和加密后的密文都能匹配上了，说明这个ascii能经过计算拼凑出正确的密文，然后进入下一次循环判断后面两位。

Part II：代码实现

```
#include <stdio.h>
#include <stdlib.h>
int func(int a1){
 if(a1>9 || a1<0){
  if(a1<=9 || a1>15){
   return 255;
  }else{
   return a1 + 87;
  }
 }else{
  return a1 + 48;
 }
}
int charToInt(char c){
 int n = c;
 return n;
}
char intToChar(int n){
 char tmp = n;
 return tmp;
}
int main(){
 char str[] = "9838e888496bfda98afdbb98a9b9a9d9cdfa29";  //length = 38;
 char s[] = "";
 int len = 0;
 for(int i=0; i<38; i=i+2){      //每次0~255遍历ASCII去验证匹配str[i]和str[i+1]两个字符是否一致，完成str字符串的全
  for(int j=0; j<255; j++){
   if((func(~j & 0xf) == charToInt(str[i]))&&(func((j>>4)^0xe) == charToInt(str[i+1]))){    //暴力匹配
    s[len++] = intToChar(j);
    break;
   }
  }
 }
 printf("%s",s);
 system("pause");
 return 0;
}
```

跑出真码flag{Y0uG0Tfutur3@}fdbb98a9b9a9d9cdfa29
就是大括号里的内容了。