第二组Writeup:

安装APK，界面显示如下：



点击“点我”按钮，提示“Hello Xman”和“Where to input your flag?”，没有输入框，反编译APK。

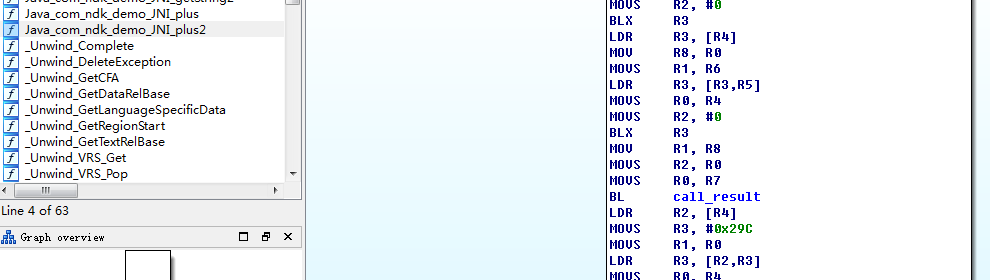


从代码中看到还有一个EditText oneEdit和Button btn2这两个组件，setVisibility为不可见，修改smaili代码二次打包，让输入框和按钮可见。

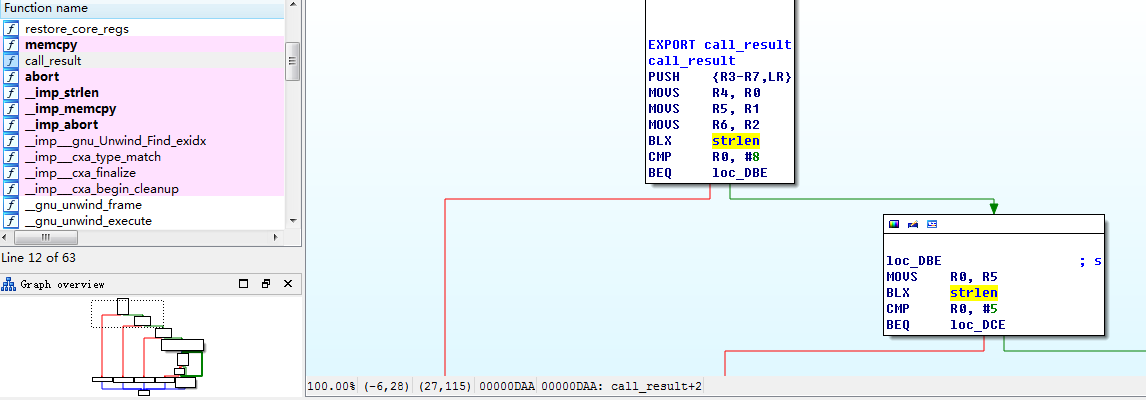
输入框和按钮可见之后，分析发现

String.valueOf(jni.plus2("flag{", oneEdit.getText().toString(), "}")是获取flag的关键地方，plus2是在so里实现的。

ADI分析so：



call\_result获取返回结果，分析call\_result代码：



int \_\_fastcall call\_result(const char \*a1, const char \*a2, const char \*a3)

{

const char \*v3; // r6@1

const char \*v4; // r7@1

const char \*v5; // r8@1

int result; // r0@2

size\_t v7; // r0@9

size\_t v8; // r0@9

size\_t v9; // r0@9

const char \*v10; // r0@9

size\_t v11; // r0@9

signed int v12; // r2@9

size\_t v13; // r12@9

int v14; // r3@9

int v15; // r0@10

int v16; // r1@10

signed int v17; // [sp+0h] [bp-88h]@9

signed int v18; // [sp+4h] [bp-84h]@9

signed int v19; // [sp+8h] [bp-80h]@9

\_\_int16 v20; // [sp+Ch] [bp-7Ch]@9

char v21; // [sp+Eh] [bp-7Ah]@9

int v22; // [sp+64h] [bp-24h]@1

v3 = a1;

v4 = a2;

v5 = a3;

v22 = \_stack\_chk\_guard;

if ( strlen(a1) == 8 )

{

if ( strlen(v4) == 5 )

{

if ( strlen(v5) == 1 )

{

v17 = 1684234849;

v18 = 1751606885;

v19 = 1818978921;

v20 = \*(\_WORD \*)"mn";

v21 = aAbcdefghijklmn[14];

v7 = strlen((const char \*)&v17);

memcpy((char \*)&v17 + v7, "opqrstuvwxye1", 0xEu);

v8 = strlen((const char \*)&v17);

memcpy((char \*)&v17 + v8, "\_3a5X7890ABCDE", 0xFu);

v9 = strlen((const char \*)&v17);

memcpy((char \*)&v17 + v9, "FGHIJKLMNOPQRST", 0x10u);

v10 = (const char \*)decode("iNETPPMM");

strcat((char \*)&v17, v10);

v11 = strlen(v3);

v12 = 31;

v13 = v11;

v14 = 0;

do

{

if ( v14 == v13 )

{

result = (int)"true";

goto LABEL\_3;

}

v15 = v3[v14];

v16 = \*((\_BYTE \*)&v17 + (v12 & 0x3F));

++v14;

v12 += 31;

}

while ( v15 == v16 );

result = (int)"wrong";

}

else

{

result = (int)"wrong";

}

}

else

{

result = (int)"wrong";

}

}

else

{

result = (int)"wrong";

}

LABEL\_3:

if ( v22 != \_stack\_chk\_guard )

\_stack\_chk\_fail(result);

return result;

}

F5直接得到以上C代码，判断java代码中jni.plus2("flag{", oneEdit.getText().toString(), "}")的第一个参数长度是否为1，第二个参数长度是否为8，第三个参数是否为5，而java调用的代码是写死的第一个参数是“flag{”，第三个参数是“}”，长度刚好是5,1，顺序相反，这个地方要仔细分析才能发现，或者动态调试的时候发现老是在if ( strlen(v4) == 5 )判断的地方返回“wrong”。这里需要再次进行二次打包，修改第一个、第三个参数内容，或者调换第一个第三个参数的顺序，满足第一个参数长度为1，第三个参数长度为5即可。

第二个参数是输入框输入的内容，从反编译得到的c代码中看出if ( v3[v8] != aAbcdefghijklmn[v7 & 0x3F] )进行判断，实质是输入字符依次同字符串“abcdefghijklmnopqrstuvwxye1\_3a5X7890ABCDEFGHIJKLMNOPQRSTcVSXnZm\_”的第((i+1)\*31)%64位进行比较。

“abcdefghijklmnopqrstuvwxye1\_3a5X7890ABCDEFGHIJKLMNOPQRSTcVSXnZm\_”是由以下代拼接码获取到的；



decode("iNETPPMM")需要进行进一步解密。

加密解密的算法都已给出：

char\* decode(char code[]) {

int i,n,len;

char prim[100];

char temp\_char,trans\_char;

int temp\_num,trans\_num;

len = strlen(code);

for(i = 0;i < len;i++) {

temp\_char = code[i]; //Consider about the corresponding digit of character

if (temp\_char <= 'z' && temp\_char >= 'a')

temp\_num = temp\_char - 'a' + 1;

else if (temp\_char <= 'Z' && temp\_char >= 'A')

temp\_num = temp\_char - 'A' + 27;

n = temp\_num % 3; //Mode 3 and compute primitive number

switch(n) {

case 0:

trans\_num = temp\_num/3;

break;

case 1:

trans\_num = 35 + temp\_num/3;

break;

case 2:

trans\_num = 18 + temp\_num/3;

break;

default:

break;

}

//Transform number to character

if (trans\_num > 26 && trans\_num <= 52)

trans\_char = 'A' + trans\_num - 27;

else if (trans\_num > 0 && trans\_num <= 26)

trans\_char = 'a' + trans\_num - 1;

prim[i] = trans\_char;

}

prim[len] = '\0';

return prim;

}

char\* encode(char prim[]) {

int i,n,len;

char code[100];

char temp\_char,trans\_char;

int temp\_num,trans\_num;

len = strlen(prim);

for(i = 0;i < len;i++) {

temp\_char = prim[i]; //Consider about the corresponding number of character

if (temp\_char <= 'z' && temp\_char >= 'a')

temp\_num = temp\_char - 'a' + 1;

else if (temp\_char <= 'Z' && temp\_char >= 'A')

temp\_num = temp\_char - 'A' + 27;

trans\_num = temp\_num\*3 % 52;

if (trans\_num > 26 && trans\_num <= 52) //Transform number to character

trans\_char = 'A' + trans\_num - 27;

else if (trans\_num > 0 && trans\_num <= 26)

trans\_char = 'a' + trans\_num - 1;

code[i] = trans\_char;

}

code[len] = '\0';

return code;

}

编写以下poc获取flag：

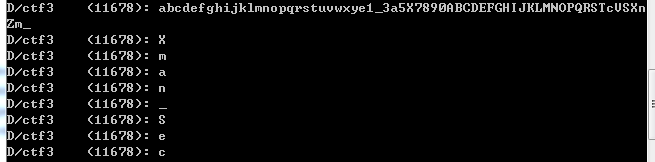
String dics="abcdefghijklmnopqrstuvwxye1\_3a5X7890ABCDEFGHIJKLMNOPQRSTcVSXnnmm";

**for**(**int** i=0;i<8;i++)

{

Log.*d*("ctf3",""+dic[((i+1)\*31)%64]);

}



Flag为”Xman\_Sec”

附件#so的c源码：

#include "com\_ndk\_demo\_AndroidNDKDemoActivity.h"

char\* call\_result (char\* a, char\* b,char\* c)

{

if(strlen(a)!=8)

{

return "wrong";

}

if(strlen(b)!=5)

{

return "wrong";

}

if(strlen(c)!=1)

{

return "wrong";

}

char \*dic="abcdefghijklmnopqrstuvwxye1\_3a5X7890ABCDEFGHIJKLMNOPQRSTcVSXnZm\_";

//char \*result = malloc(strlen(a)+strlen(b)+1);

//strcpy(result, "flag{");

//strcat(result, a);

//strcat(result, b);

int i=0;

for(i=0;i<strlen(a);i++)

{

if(a[i]!=dic[((i+1)\*31)%64])

{

return "wrong";

}

}

return "true";

}

JNIEXPORT jstring JNICALL Java\_com\_ndk\_demo\_JNI\_getString(JNIEnv\* env, jobject obj)

{

//·µ»ØÒ»¸ö×Ö·û´®

return (\*env)->NewStringUTF(env,"Hello XMAN!");

}

/\*

\* Class: com\_ndk\_demo\_JNI

\* Method: plus

\* Signature: (II)I

\*/

JNIEXPORT jint JNICALL Java\_com\_ndk\_demo\_JNI\_plus(JNIEnv\* env, jobject obj, jint a, jint b)

{

//·µ»Ø¼ÆËã½á¹û

return a+b;

}

JNIEXPORT jstring JNICALL Java\_com\_ndk\_demo\_JNI\_plus2(JNIEnv\* env, jobject obj,jstring c, jstring a, jstring b)

{

char \*str1 = (\*env)->GetStringUTFChars(env, a, 0);

char \*str2 = (\*env)->GetStringUTFChars(env, b, 0);

char \*str3 = (\*env)->GetStringUTFChars(env, c, 0);

char \*re=call\_result(str1,str2,str3);

const char \*cc="hello";

jstring jrstr;

jrstr=(\*env)->NewStringUTF(env, re);

return jrstr;

}

JNIEXPORT jstring JNICALL Java\_com\_ndk\_demo\_JNI\_getString2(JNIEnv\* env, jobject obj)

{

//·µ»ØÒ»¸ö×Ö·û´®

return (\*env)->NewStringUTF(env,"Where to input your flag?");

}