

# 看雪3万课程笔记-FRIDA高级API实用方法: Frida Hook Native层

原创

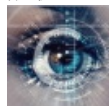
kfyzd2008 于 2022-03-02 14:39:36 发布 186 收藏

分类专栏: [安卓](#) 文章标签: [frida](#) [frida hook](#) [看雪三万](#) [fridahook](#) [安卓逆向](#)

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/kfyzd2008/article/details/123211197>

版权



[安卓专栏收录该内容](#)

19 篇文章 5 订阅

订阅专栏

一、使用工具:

apk: 攻防世界中CTF题, 安装后图表显示黑客精神 xman.apk

工具: IDA, jadx

二、知识点:

- 1、Module.findBaseAddress("libmyjni.so");//获取基地址
- 2、Module.findExportByName("libmyjni.so","n2");//获取函数地址
- 3、Interceptor.attach //根据地址 hook native层函数
- 4、Process.findModuleByName("libart.so");//返回一个模块
- 5、enumerateSymbols //枚举模块符号 (API)

三、分析:

1、首先jadx打开app分析java代码

```
MyApp myApp = (MyApp) getApplication();
int m = MyApp.m;
if (m == 0) {
    str2 = "未注册";
} else if (m == 1) {
    str2 = "已注册";
} else {
    str2 = "已混乱";
}
setTitle("Xman" + str2);
this.btn1 = (Button) findViewById(R.id.button1);
this.btn1.setOnClickListener(new View.OnClickListener() {
    /* class com.gdufs.xman.MainActivity.AnonymousClass1 */

    public void onClick(View v) {
        MyApp myApp = (MyApp) MainActivity.this.getApplication();
        if (MyApp.m == 0) {
            MainActivity.this.doRegister();
            return;
        }
        ((MyApp) MainActivity.this.getApplication()).work();
        Toast.makeText(MainActivity.this.getApplicationContext(), MainActivity.workString, 0).show();
    }
});
```

关键判断点m的值为1则通过为0则调用注册

CSDN @kfyzd2008

```
6 public class MyApp extends Application {
    public static int m = 0;

    public native void initSN();

    public native void saveSN(String str);

    public native void work();

    static {
3      System.loadLibrary("myjni");
    }

6    public void onCreate() {
7      initSN();
8      Log.d("com.gdufs.xman m=", String.valueOf(m));
9      super.onCreate();
    }
}

CSDN @kfyzd2008
```

从上图可以看出关键判断点为MyApp中m的值，从图二可以看出MyApp的方法注册在native层。有initSN、saveSN、work三个函数。

2、解压apk用IDA打开libmyjni.so文件。导出函数中不能直接搜索到上面三个函数，我们直接双击JNI\_ONLOAD，F5查看伪代码

```
1 jint JNI_OnLoad(JavaVM *vm, void *reserved)
2 {
3   if ( !(*vm->GetEnv(vm, (void **)&g_env, 65542) ) GetEnv第二个参数为JNIEnv* 我们手动改一下
4   {
5     j__android_log_print(2, "com.gdufs.xman", "JNI_OnLoad()");
6     native_class = (*(int (__fastcall **)(int, const char *))(*(_DWORD *)g_env + 24))(g_env, "com/gdufs/xman/MyApp");
7     if ( !(*(int (__fastcall **)(int, int, char **, int))(*(_DWORD *)g_env + 860))(g_env, native_class, off_5004, 3) )
8     {
9       j__android_log_print(2, "com.gdufs.xman", "RegisterNatives() --> nativeMethod() ok");
10      return 65542;
11    }
12    j__android_log_print(6, "com.gdufs.xman", "RegisterNatives() --> nativeMethod() failed");
13  }
14  return -1;
15 }
```

```
1 jint JNI_OnLoad(JavaVM *vm, void *reserved)
2 {
3   if ( !(*vm->GetEnv(vm, (void **)&g_env, 65542) )
4   {
5     j__android_log_print(2, "com.gdufs.xman", "JNI_OnLoad()");
6     native_class = (int)(*g_env->FindClass(g_env, "com/gdufs/xman/MyApp"));
7     if ( !(*g_env->RegisterNatives(g_env, (jClass)native_class, (const JNINativeMethod *)off_5004, 3) )
8     {
9       j__android_log_print(2, "com.gdufs.xman", "RegisterNatives() --> nativeMethod() ok");
10      return 65542;
11    }
12    j__android_log_print(6, "com.gdufs.xman", "RegisterNatives() --> nativeMethod() failed");
13  }
14  return -1;
15 }
```

当我们手动更改GetEnv中的第二个参数之后，IDA自动识别出了另外两个API，FindClass、RegisterNatives 图中RegisterNatives的第三个参数是注册的方法数组，我们双击进入。

```

.data:00005000 ; sub_11BC+4fo ...
.data:00005001 DCB 0
.data:00005002 DCB 0
.data:00005003 DCB 0
.data:00005004 off_5004 DCD aInitsn ; DATA XREF: JNI_OnLoad+46fo
.data:00005004 ; JNI_OnLoad+46fo ...
.data:00005004 ; "initSN"
.data:00005008 DCD aV ; "()"V
.data:0000500C DCD n1+1
.data:00005010 DCD aSavesn ; "saveSN"
.data:00005014 DCD aLjavaLangStrin ; "(Ljava/lang/String;)V"
.data:00005018 DCD n2+1
.data:0000501C DCD aWork ; "work"
.data:00005020 DCD aV ; "()"V
.data:00005024 DCD n3+1
.data:00005024 ; .data ends

```

CSDN @kfyzjd2008

此处注册了我们上面说的三个参数。双击进入saveSN我们查看此处代码。

```

1 int __fastcall n2(int a1, int a2, int a3)
2 {
3     int v5; // r7
4     const char *v7; // r6
5     const char *v8; // r5
6     int v9; // r4
7     int v10; // r0
8     char v11; // r2
9     signed int v12; // [sp+8h] [bp-38h]
10    char v13[20]; // [sp+10h] [bp-30h] BYREF
11
12    v5 = j_fopen("/sdcard/reg.dat", "w"); ← 打开文件
13    if ( !v5 )
14        return j__android_log_print(3, "com.gdufs.xman", byte_2E5A);
15    strcpy(v13, "W3_arE_whO_we_ARE");
16    v7 = (const char *)((*int (__fastcall **)(int, int, _DWORD))(*(_DWORD *)a1 + 676))(a1, a3, 0);
17    v8 = v7;
18    v12 = j_strlen(v7);
19    v9 = 2016;
20    while ( 1 )
21    {
22        v10 = v8 - v7;
23        if ( v8 - v7 >= v12 )
24            break;
25        if ( v10 % 3 == 1 )
26        {
27            v9 = (v9 + 5) % 16;
28            v11 = v13[v9 + 1];
29        }
30        else if ( v10 % 3 == 2 )
31        {
32            v9 = (v9 + 7) % 15;
33            v11 = v13[v9 + 2];
34        }
35        else
36        {
37            v9 = (v9 + 3) % 13;
38            v11 = v13[v9 + 3];
39        }
40        *v8++ ^= v11; ← 写入文件、关闭文件
41    }
42    j_fputs(v7, v5); ← 写入文件、关闭文件
43    return j_fclose(v5);
44 }

```

CSDN @kfyzjd2008

我们将函数的第一个参数a1按Y修改为JNIEnv\* a1 (native层的函数第一个参数都是JNIEnv\*)

我们将函

```
LDA View 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
int __fastcall n2(JNIEnv *a1, int a2, int a3)
{
    int v5; // r7
    const char *v7; // r6
    const char *v8; // r5
    int v9; // r4
    int v10; // r0
    char v11; // r2
    signed int v12; // [sp+8h] [bp-38h]
    char v13[20]; // [sp+10h] [bp-30h] BYREF
1
2    v5 = j_fopen("/sdcard/reg.dat", "w+");
3    if ( !v5 )
4        return j__android_log_print(3, "com.gdufs.xman", byte_2E5A);
5    strcpy(v13, "W3 are who are");
6    v7 = (*a1)->GetStringUTFChars(1, a3, 0);
7    v8 = v7;
8    v12 = j_strlen(v7);
9    v9 = 2016;
10   while ( 1 )
11
12
13
14
15
16
17
18
19
20
```

CSDN @kfyjd2008

此时识别出来了一个API GetStringUTFChars

此处我们hook这个函数和这个API分别看看他们传入的参数

```
function hook_native(){
    var base_myjni = Module.findBaseAddress("libmyjni.so");//获取基地址

    if (base_myjni){
        var n2 = Module.findExportByName("libmyjni.so","n2");//获取函数地址 函数地址 - 基地址 = 静态偏移地址(ida
        console.log("base_myjni:",base_myjni,"n2:",n2);//thumb模式函数,0x000011F8,实际地址: 0xcc2691f9 ,thumb
        Interceptor.attach(n2,{//根据地址 hook native层函数
            onEnter: function(args){
                console.log("n2 onEnter:",args[0],args[1],args[2]);
            },onLeave: function(retval){

            }
        });
    }
}
```

```

function hook_libart(){
  //获取模块地址
  var module_libart = Process.findModuleByName("libart.so");//android常用API都在此文件中
  var symbols = module_libart.enumerateSymbols();//枚举模块中的所有API，返回一个对象数组

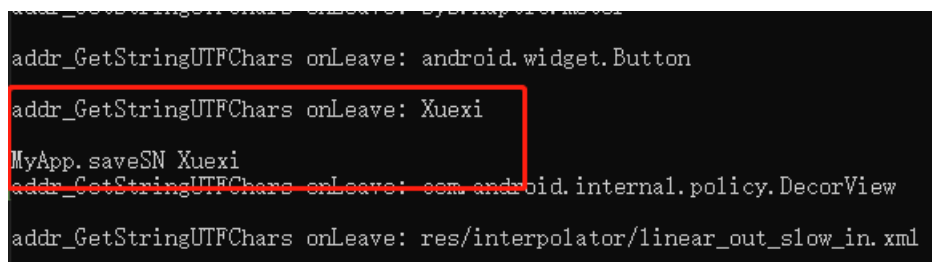
  var addr_GetStringUTFChars = null;
  for (var i = 0; i < symbols.length; i++){
    // console.log(symbols[i].name);
    var name = symbols[i].name;
    if (name.indexOf("art") >= 0){
      if((name.indexOf("CheckJNI") == -1) && (name.indexOf("JNI") >= 0)){
        if(name.indexOf("GetStringUTFChars") >= 0){
          console.log(name);
          addr_GetStringUTFChars = symbols[i].address;//获取API地址
        }
      }
    }
  }

  if(addr_GetStringUTFChars){
    //hook函数地址
    Interceptor.attach(addr_GetStringUTFChars,{
      onEnter:function(args){

      },onLeave:function(retval){
        console.log("addr_GetStringUTFChars onLeave:",ptr(retval).readCString() + "\r\n");//将地址用
      }
    })
  }
}
}

```

重新输入内容查看控制台输出：



```

addr_GetStringUTFChars onLeave: android.widget.Button
addr_GetStringUTFChars onLeave: Xuexi
MyApp.saveSN Xuexi
addr_GetStringUTFChars onLeave: com.android.internal.policy.DecorView
addr_GetStringUTFChars onLeave: res/interpolator/linear_out_slow_in.xml

```

此处值与我们输入的值相同，但是并没有什么卵用，后面还有一点查看堆栈的代码：

```

if(addr_GetStringUTFChars){
    //hook函数地址
    Interceptor.attach(addr_GetStringUTFChars,{
        onEnter:function(args){
            //查看堆栈调用
            console.log("addr_GetStringUTFChars onEnter called from:\n" +
                Thread.backtrace(this.context, Backtracer.FUZZY)
                    .map(DebugSymbol.fromAddress).join('\n') + '\n');

        },onLeave:function(retval){
            console.log("addr_GetStringUTFChars onLeave:",ptr(retval).readCString() + "\r\n");//将地址用
        }
    })
}
}
}
}

```

上面的分析基本没啥信息，我们再次来到IDA中的三个函数，这次我们进入initSN中查看代码：

```

1 int __fastcall n1(int a1)
2 {
3     int v2; // r0
4     int v3; // r4
5     int v4; // r0
6     int v5; // r7
7     const char *v6; // r5
8     int v8; // r0
9     int v9; // r1
10
11     v2 = j_fopen("/sdcard/reg.dat", "r+");
12     v3 = v2;
13     if ( !v2 )
14     {
15         v4 = a1;
16         return setValue(v4, 0);
17     }
18     j_fseek(v2, 0, 2);
19     v5 = j_ftell(v3);
20     v6 = (const char *)j_malloc(v5 + 1);
21     if ( !v6 )
22     {
23         j_fclose(v3);
24         v4 = a1;
25         return setValue(v4, 0);
26     }
27     j_fseek(v3, 0, 0);
28     j_fread(v6, v5, 1, v3);
29     v6[v5] = 0;
30     if ( !j_strcmp(v6, "EoPAoY62@EIRD") )
31     {
32         v8 = a1;
33         v9 = 1;
34     }
35     else
36     {
37         v8 = a1;
38         v9 = 0;
39     }
40     setValue(v8, v9);
41     return j_fclose(v3);
42 }

```

CSDN @kfyjzd2008

此处有多个setValue函数，我们双击进入查看：

```
1 int __fastcall setValue(int a1, int a2)
2 {
3     int v4; // r5
4     int v5; // r0
5
6     v4 = (*(int (__fastcall **)(int, const char *)))(_DWORD *)a1 + 24)(a1, "com/gdufs/xman/MyApp");
7     v5 = (*(int (__fastcall **)(int, int, const char *, const char *)))(_DWORD *)a1 + 576)(a1, v4, "m", "I");
8     return (*(int (__fastcall **)(int, int, int, int)))(_DWORD *)a1 + 636)(a1, v4, v5, a2);
9 }
```

此处将第一个参数改为JNIEnv\*

```
1 int __fastcall setValue(JNIEnv *a1, int a2)
2 {
3     jclass v4; // r5
4     jfieldID v5; // r0
5
6     v4 = (*a1)->FindClass(a1, "com/gdufs/xman/MyApp");
7     v5 = (*a1)->GetStaticFieldID(a1, v4, "m", "I");
8     return ((int (__fastcall *)(JNIEnv *, jclass, jfieldID, int)))(*a1)->SetStaticIntField)(a1, v4, v5, a2);
9 }
```

将第一个参数更改为JNIEnv\*

此时自动识别出函数FindClass, GetStaticFieldID, SetStaticIntField。看API名称大致意思为获取一个类, 获取静态值, 赋值静态值。

现在我们对这三个函数进行hook看看他们的参数。在 hook\_libart 函数中加入:

```
if(name.indexOf("FindClass") >= 0){
    console.log(name);
    addr_FindClass = symbols[i].address;
}

if(name.indexOf("GetStaticFieldID") >= 0){
    console.log(name);
    addr_GetStaticFieldID = symbols[i].address;
}

if(name.indexOf("SetStaticIntField") >= 0){
    console.log(name);
    addr_SetStaticIntField = symbols[i].address;
}
```

循环后加入

```

if(addr_GetStringUTFChars){
    //hook函数地址
    Interceptor.attach(addr_GetStringUTFChars,{
        onEnter:function(args){
            //查看堆栈调用
            // console.log("addr_GetStringUTFChars onEnter called from:\n" +
            //     Thread.backtrace(this.context, Backtracer.FUZZY)
            //     .map(DebugSymbol.fromAddress).join('\n') + '\n');

            },onLeave:function(retval){
                // console.log("addr_GetStringUTFChars onLeave:",ptr(retval).readCString() + "\r\n");
            }
        })
    }

if(addr_FindClass){
    //hook函数地址
    Interceptor.attach(addr_FindClass,{
        onEnter:function(args){
            console.log("addr_FindClass: ",ptr(args[1]).readCString());
        },onLeave:function(retval){

        }
    })
}

if(addr_GetStaticFieldID){
    //hook函数地址
    Interceptor.attach(addr_GetStaticFieldID,{
        onEnter:function(args){
            console.log("addr_GetStaticFieldID: ",ptr(args[2]).readCString(),ptr(args[3]).readCString()
            ),onLeave:function(retval){

            }
        })
    }

if(addr_SetStaticIntField){
    //hook函数地址
    Interceptor.attach(addr_SetStaticIntField,{
        onEnter:function(args){
            console.log("addr_SetStaticIntField: ");
            console.log("addr_SetStaticIntField: ",args[3]);
        },onLeave:function(retval){

        }
    })
}
}
}

```

再次执行:



```
addr_FindClass: java/lang/reflect/ParameterizedType
addr_FindClass: com/gdufs/xman/MyApp
function hook_java() { [ecmascript code] }
addr_FindClass: java/lang/String
addr_FindClass: java/lang/String
addr_FindClass: com/gdufs/xman/MyApp
addr_GetStaticFieldID: m I
addr_SetStaticIntField: 0x0
addr_FindClass: java/io/FileNotFoundException
addr_FindClass: [B
addr_FindClass: [B
addr_FindClass: [B
```

从图中我们可以看到m被赋值了0

此时视频给出了两个解决方案：

方案1：直接更改/sdcard/reg.dat文件内容为："EoPAoY62@EIRD"

方案2：使用frida代码写入/sdcard/reg.dat文件内容为"EoPAoY62@EIRD"

```
function write_reg_dat(){
    var file = new File("/sdcard/reg.dat","w");
    file.write("EoPAoY62@EIRD");
    file.flush();
    file.close();
}
```

但是这两种方法我测试好像不行。不知道为什么。

正确的输入值应该是：201608Am!2333