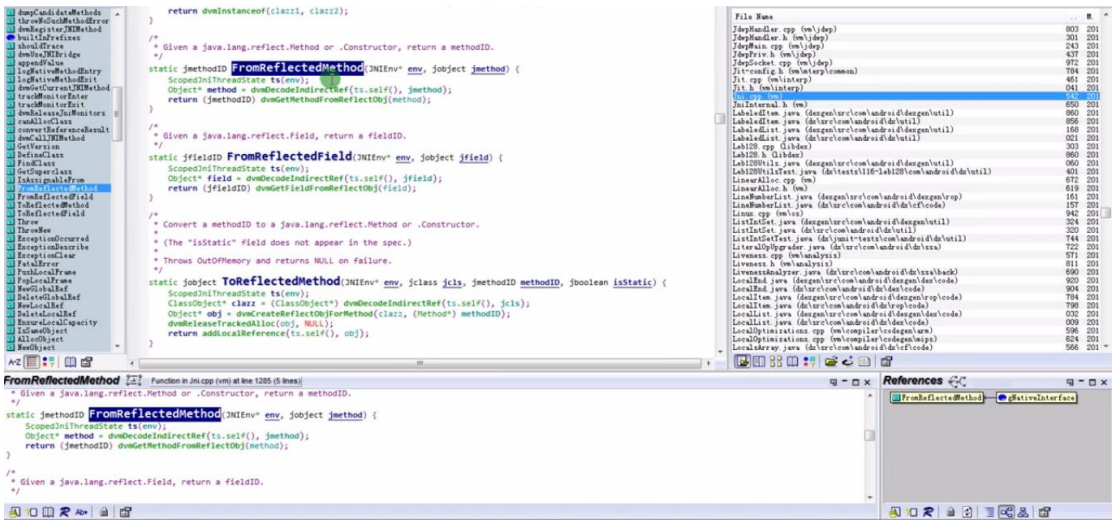
一、方法 1

1. Jni Bridge

Jni 提供了让我们在 C++代码层中直接操作 Dalvik(java)数据的接口。我们可以在 jni 中直接操作相关的数据，修改 Android 中的代码。

Android 源码中的实现：dalvik/vm/jni.cpp

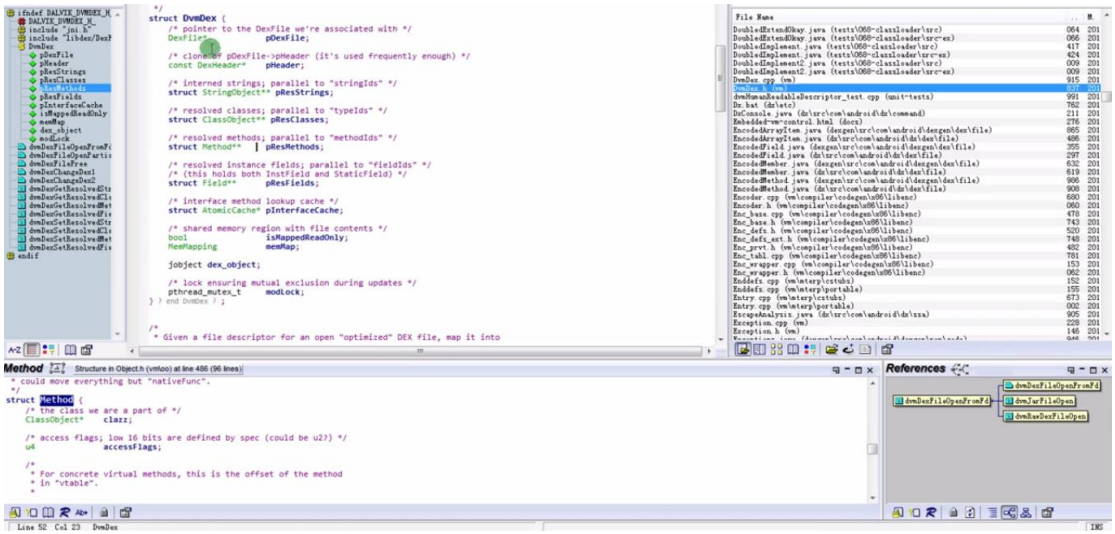


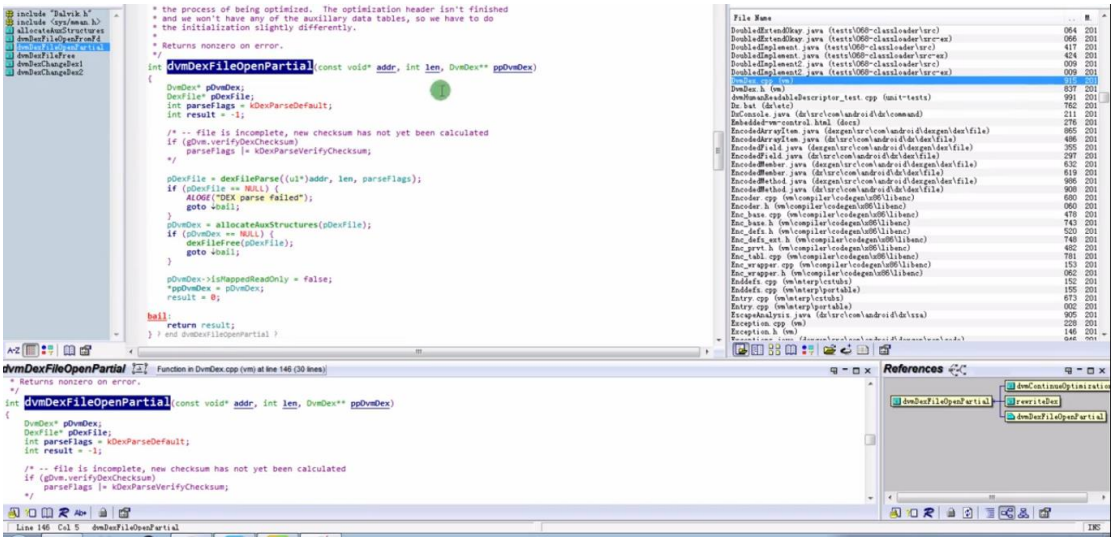
2. Object 结构体

Android 源码位置：dalvik/vm/oo/Object

Android 运行时，解析 Dex 文件，并生成相关的结构体：DvmDex。

该结构体中存储了各种字符串、类、方法等信息。加载的时候调用 dvmDexFileOpenPartial(/dalvik/vm/DvmDex.cpp)对 Dex 文件进行解析，并转化为可执行的结构体。这也是为什么这个函数可以作为脱壳用的函数的原因之一，以前的爱加密可以直接通过钩这个函数进行脱壳。

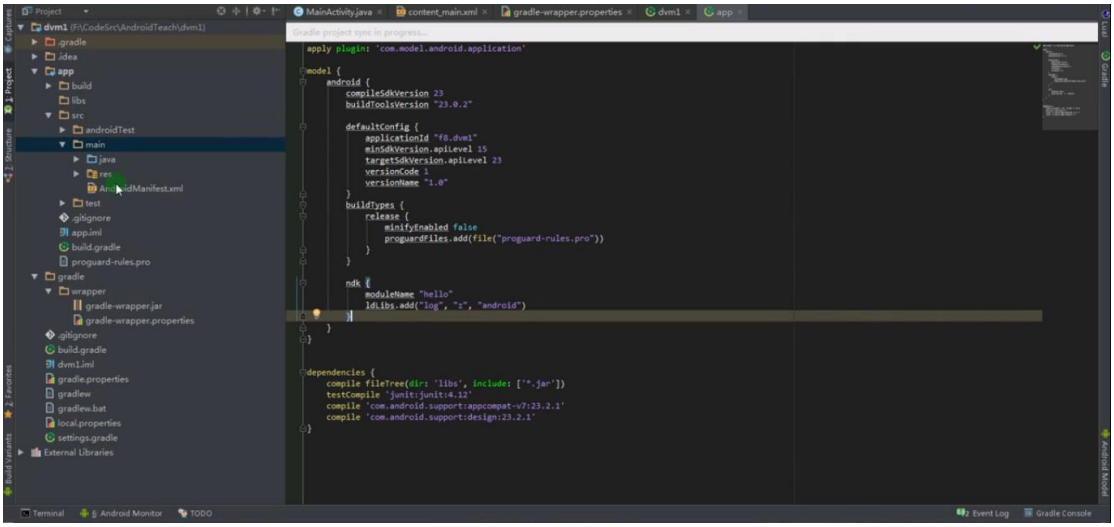




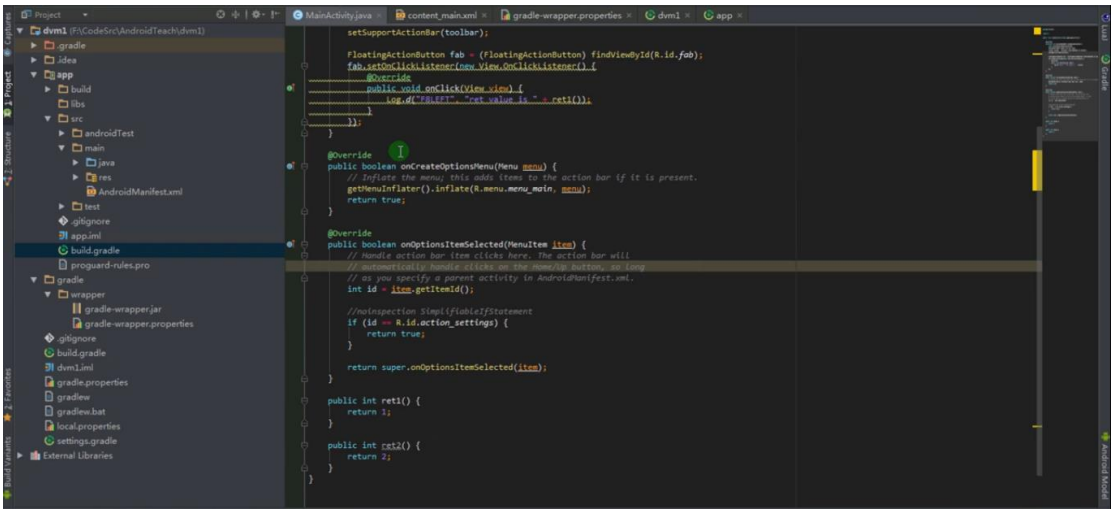
其中，Method 结构体则是根据 DexMethod 生成的执行方法类。Dalvik 执行代码时，都是从 Method 中取出代码来执行的。因此可以直接通过操作 Method 结构体来修改执行的代码。

3. **Android 源码导出**

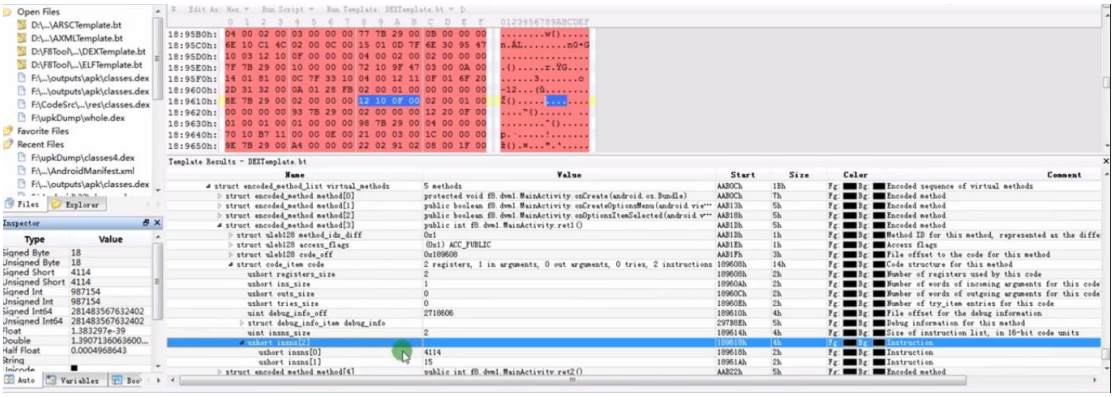
因为安全性，jni 中的结构体是部分导出的，所以需要补全相关的结构体才能操作其数据。

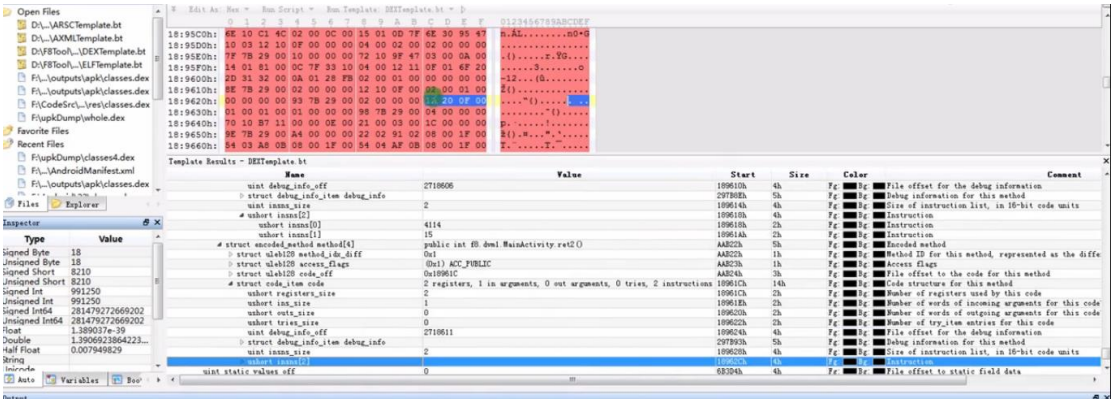


定义了两个返回数值的函数，并且在按钮点击时调用函数 ret1()

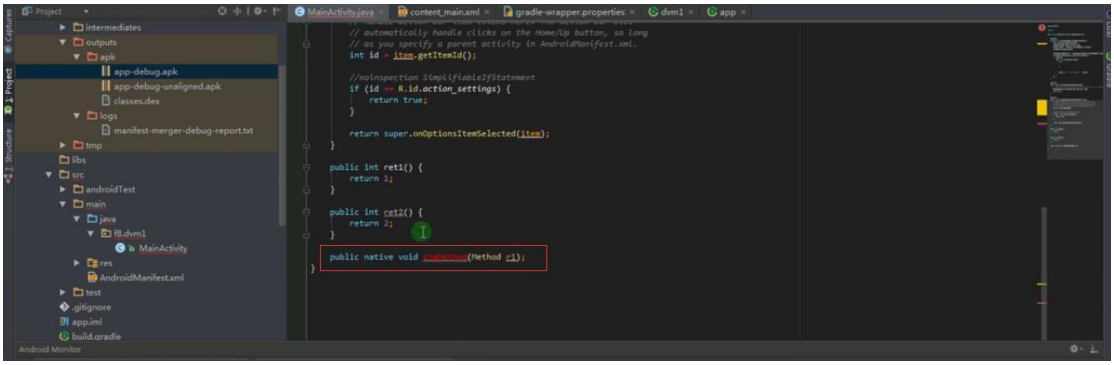


编译运行后取出 classes.dex 文件进行查看，得到两个函数的 isns 结构

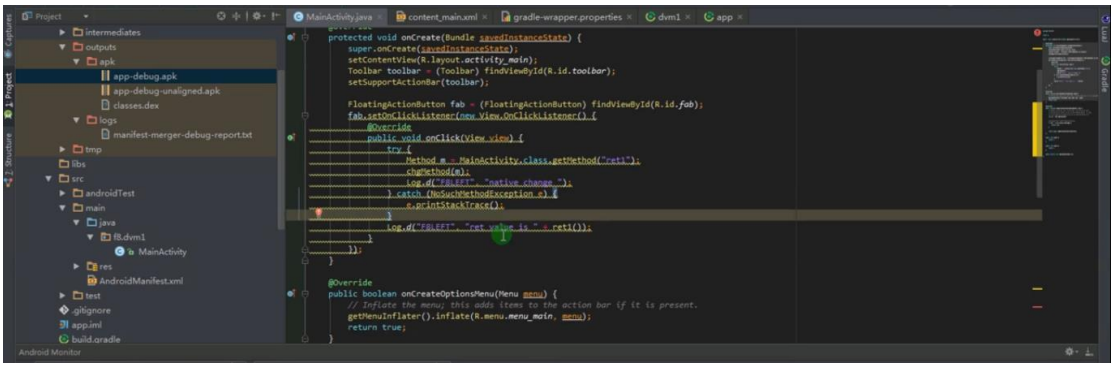




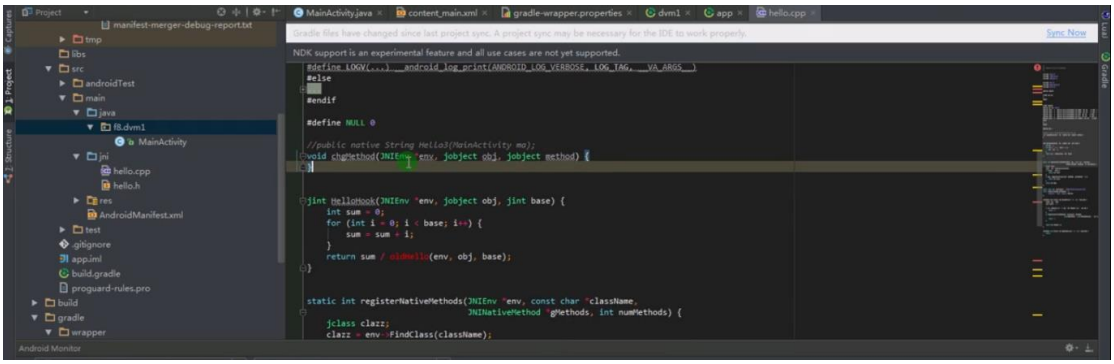
通过反射来获取函数调用



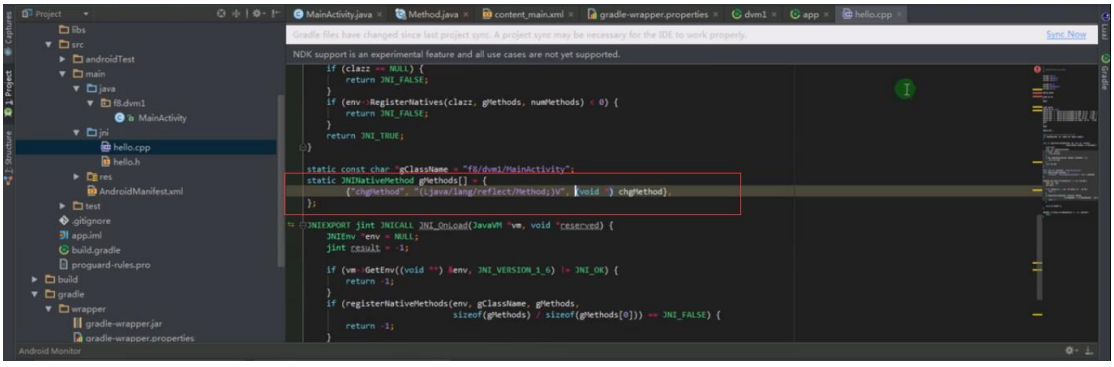
```
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public int ret1() {
        return 1;
    }

    public int ret2() {
        return 2;
    }

    public native void chgHook(Method r1);
}
```



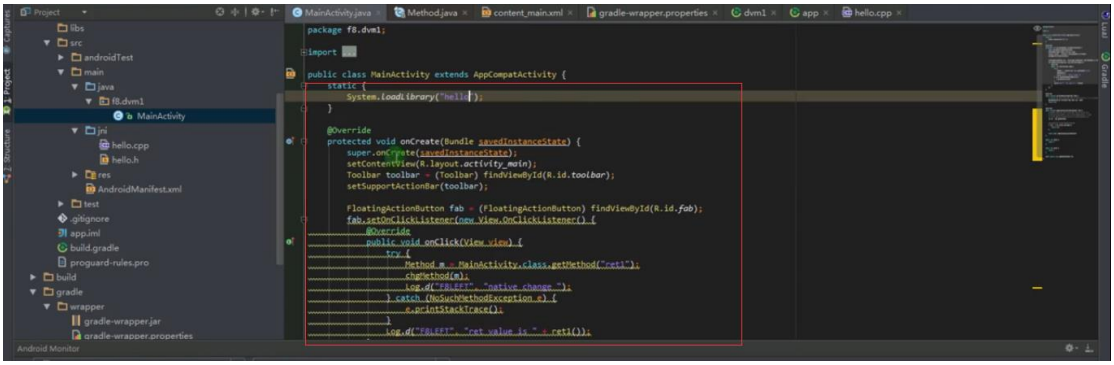
```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                try {
                    Method m = MainActivity.class.getMethod("ret1");
                    chgMethod(m);
                    Log.d("FBLET", "native change ");
                } catch (NoSuchMethodException e) {
                    e.printStackTrace();
                }
                Log.d("FBLET", "ret value is " + ret1());
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }
```



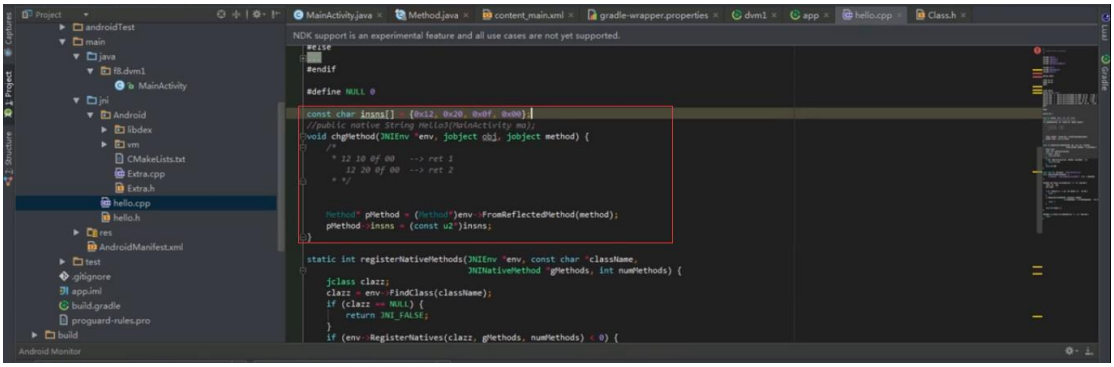
```
#define LOGV(...)  __android_log_print(ANDROID_LOG_VERBOSE, LOG_TAG, __VA_ARGS__)
#else
#endif

#define NULL 0

//public native String Hello3(MainActivity ma);
void chgMethod(JNIEnv *env, jobject obj, jobject method) {
}


jint HelloHook(JNIEnv *env, jobject obj, jint base) {
    int sum = 0;
    for (int i = 0; i < base; i++) {
        sum = sum + i;
    }
    return sum / oldHello(env, obj, base);
}


static int registerNativeMethods(JNIEnv *env, const char *className,
                                 JNINativeMethod *gMethods, int numMethods) {
    jclass clazz;
    clazz = env->FindClass(className);
```

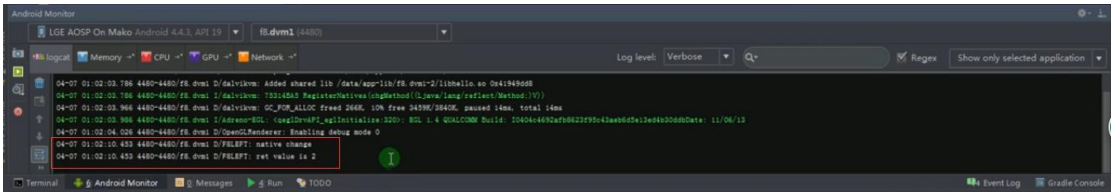


```
    if (clazz == NULL) {
        return JNI_FALSE;
    }
    if (env->RegisterNatives(clazz, gMethods, numMethods) < 0) {
        return JNI_FALSE;
    }
    return JNI_TRUE;
}

static const char *gClassName = "f8/dvm1/MainActivity";
static JNINativeMethod gMethods[] = {
        {"chgMethod", "(Ljava/lang/reflect/Method;)V", (void *) chgMethod},
};

JNIEXPORT jint JNICALL JNI_OnLoad(JavaVM *vm, void *reserved) {
    JNIEnv *env = NULL;
    jint result = -1;

    if (vm->GetEnv((void **) &env, JNI_VERSION_1_6) != JNI_OK) {
        return -1;
    }
    if (registerNativeMethods(env, gClassName, gMethods,
                              sizeof(gMethods) / sizeof(gMethods[0])) == JNI_FALSE) {
        return -1;
    }
```



需要导入部分源码才能使 Method*被识别



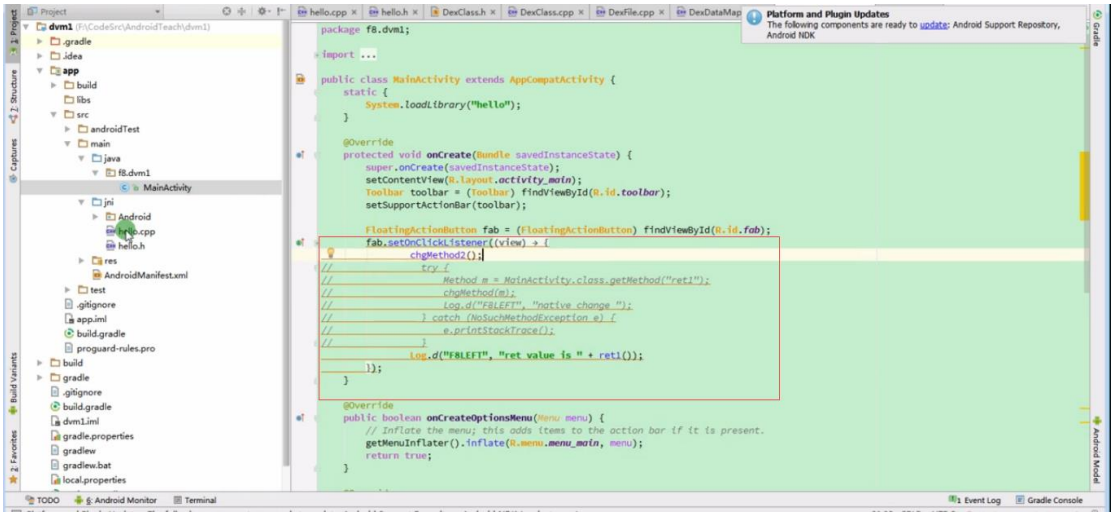


最后调用函数被更改成功

二、方法 2

1. DexFile 修改

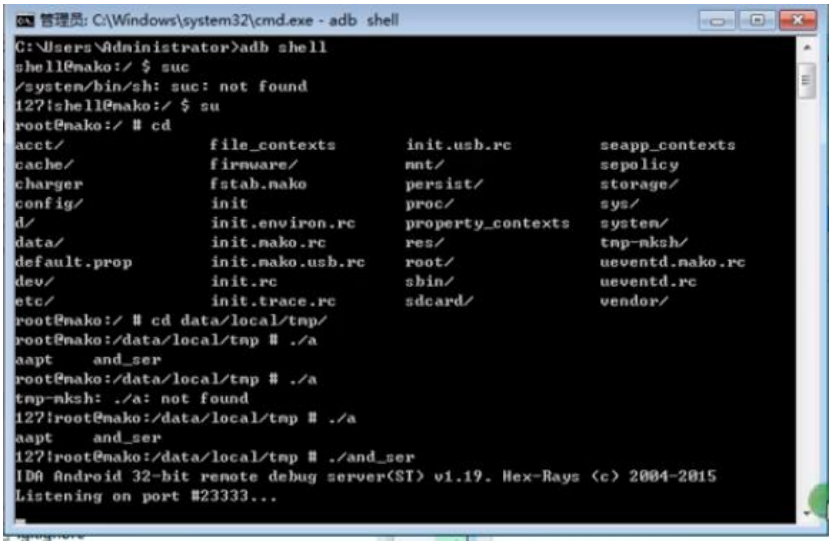
根据示例一，可以看到直接修改内存中的 DexFile 文件数据可以达到修改代码的效果。具体为：DexFile -> DexClassDef -> DexClassData -> DexMethod -> DexCode -> insns

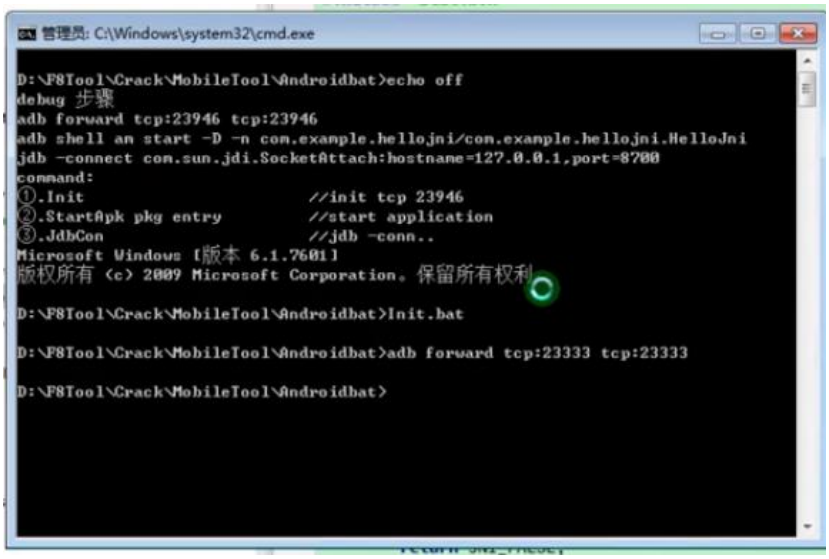


```
package f8.dvm1;

import ...

public class MainActivity extends AppCompatActivity {
    static {
        System.loadLibrary("hello");
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener((view) -> {
                chgMethod2();
//                try {
//                    Method m = MainActivity.class.getMethod("ret1");
//                    chgMethod(m);
//                    Log.d("F8LEFT", "native change ");
//                } catch (NoSuchMethodException e) {
//                    e.printStackTrace();
//                }
                Log.d("F8LEFT", "ret value is " + ret1());
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }
```

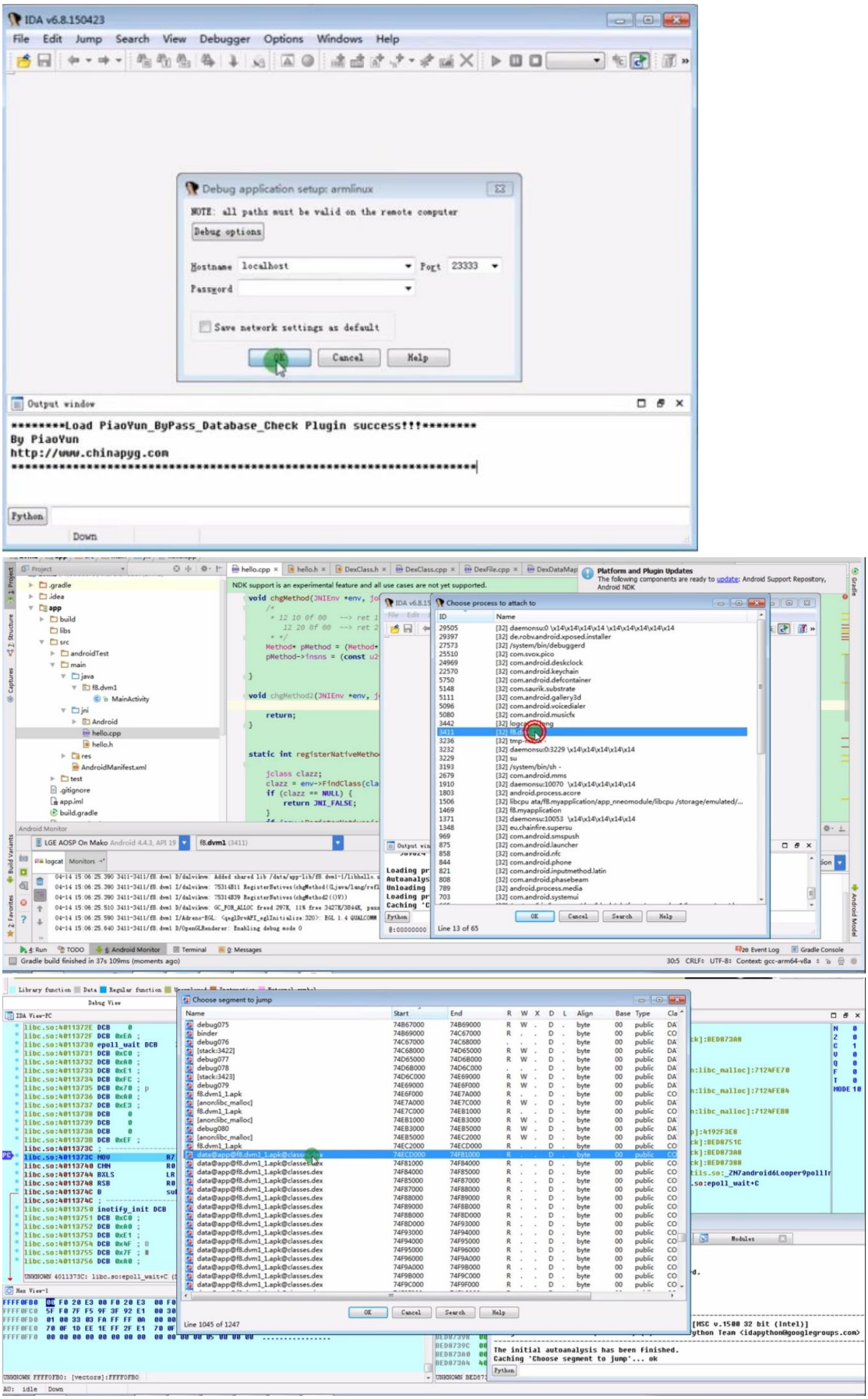
```
    pMethod->insns = (const u2*)insns;

}


static int registerNativeMethods(JNIEnv *env, const char *className,
                                 JNINativeMethod *gMethods, int numMethods) {
    jclass clazz;
    clazz = env->FindClass(className);
    if (clazz == NULL) {
        return JNI_FALSE;
    }
    if (env->RegisterNatives(clazz, gMethods, numMethods) < 0) {
        return JNI_FALSE;
    }
    return JNI_TRUE;
}

static const char *gClassName = "f8/dvm1/MainActivity";
static JNINativeMethod gMethods[] = {
        {"chgMethod", "(Ljava/lang/reflect/Method;)V", (void *) chgMethod},
        {"chgMethod2", "()V", (void *)chgMethod2},
};

JNIEXPORT jint JNICALL JNI_OnLoad(JavaVM *vm, void *reserved) {
    JNIEnv *env = NULL;
    jint result = -1;

    if (vm->GetEnv((void **) &env, JNI_VERSION_1_6) != JNI_OK) {
        return -1;
    }
    if (registerNativeMethods(env, gClassName, gMethods,
                              sizeof(gMethods) / sizeof(gMethods[0])) == JNI_FALSE) {
        return -1;
    }
```

```
#include <string.h>
#include <stdlib.h>
#define NULL 0

const char insns[] = {0x12, 0x20, 0x0f, 0x00};
//public native String Hello3(MainActivity ma);
void chgMethod(JNIEnv *env, jobject obj, jobject method) {
    /*
     * 12 10 0f 00   --> ret 1
       12 20 0f 00  --> ret 2
     * */
    Method* pMethod = (Method*)env->FromReflectedMethod(method);
    pMethod->insns = (const u2*)insns;

}

void chgMethod2(JNIEnv *env, jobject obj) {
    return;
}


static int registerNativeMethods(JNIEnv *env, const char *className,
                                 JNINativeMethod *gMethods, int numMethods) {
    jclass clazz;
    clazz = env->FindClass(className);
    if (clazz == NULL) {
        return JNI_FALSE;
    }
    if (env->RegisterNatives(clazz, gMethods, numMethods) < 0) {
        return JNI_FALSE;
    }
    return JNI_TRUE;
}

static const char *gClassName = "f8/dvm1/MainActivity";
static JNINativeMethod gMethods[] = {
```


管理员: C:\Windows\system32\cmd.exe - adb shell
C:\Users\Administrator>adb shell
shell@mako:/ $ suc
/system/bin/sh: suc: not found
127|shell@mako:/ $ su
root@mako:/ # cd
acct/ file_contexts init.usb.rc seapp_contexts
cache/ firmware/ mnt/ sepolicy
charger fstab.mako persist/ storage/
config/ init proc/ sys/
d/ init.environ.rc property_contexts system/
data/ init.mako.rc res/ tmp-mksh/
default.prop init.mako.usb.rc root/ ueventd.mako.rc
dev/ init.rc sbin/ ueventd.rc
etc/ init.trace.rc sdcard/ vendor/
root@mako:/ # cd data/local/tmp/
root@mako:/data/local/tmp # ./a
aapt and_ser
root@mako:/data/local/tmp # ./a
tmp-mksh: ./a: not found
127|root@mako:/data/local/tmp # ./a
aapt and_ser
127|root@mako:/data/local/tmp # ./and_ser
IDA Android 32-bit remote debug server(ST) v1.19. Hex-Rays (c) 2004-2015
Listening on port #23333...


管理员: C:\Windows\system32\cmd.exe
D:\F8Tool\Crack\MobileTool\Androidbat>echo off
debug 步骤
adb forward tcp:23946 tcp:23946
adb shell am start -D -n com.example.hellojni/com.example.hellojni.HelloJni
jdb -connect com.sun.jdi.SocketAttach:hostname=127.0.0.1,port=8700
command:
①.Init //init tcp 23946
②.StartApk pkg entry //start application
③.JdbCon //jdb -conn..
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。
D:\F8Tool\Crack\MobileTool\Androidbat>Init.bat
D:\F8Tool\Crack\MobileTool\Androidbat>adb forward tcp:23333 tcp:23333
D:\F8Tool\Crack\MobileTool\Androidbat>


Project
hello.cpp
hello.h
DexClass.h
DexClass.cpp
DexFile.cpp
DexDataMap
Platform and Plugin Updates
The following components are ready to update: Android Support Repository, Android NDK
NDK support is an experimental feature and all use cases are not yet supported.
.gradle
.idea
app
build
libs
src
androidTest
main
java
f8.dvm1
MainActivity
jni
Android
hello.cpp
hello.h
res
AndroidManifest.xml
test
.gitignore
app.iml
build.gradle
proguard-rules.pro
* Allocate a single chunk for the DexDataMap per se as well as the
* two arrays.
size_t size = 0;
DexDataMap* map = NULL;
* Avoiding pulling in safe_iop for safe_iopf.
// if (!safe_mul(&size, maxCount, sizeof(u4) + sizeof(u2)) ||
// !safe_add(&size, size, sizeof(DexDataMap))) {
return NULL;
// }
map = (DexDataMap*) malloc(size);
if (map == NULL) {
return NULL;
}
map->count = 0;
map->max = maxCount;
Messages Gradle Build



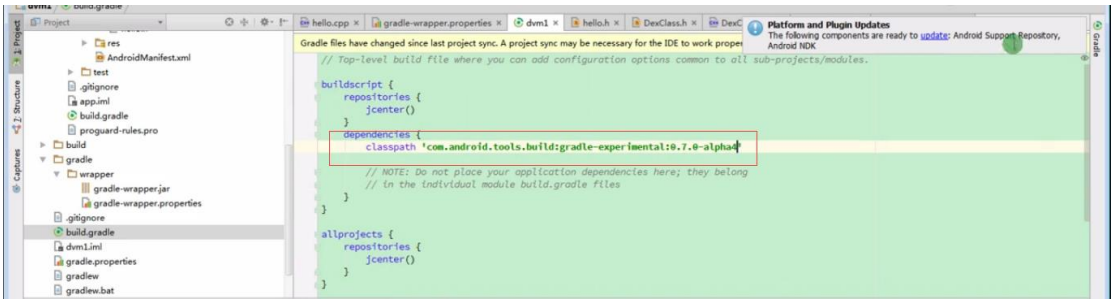
2. 内存中 **DexFile** 定位

遍历 Map 的函数

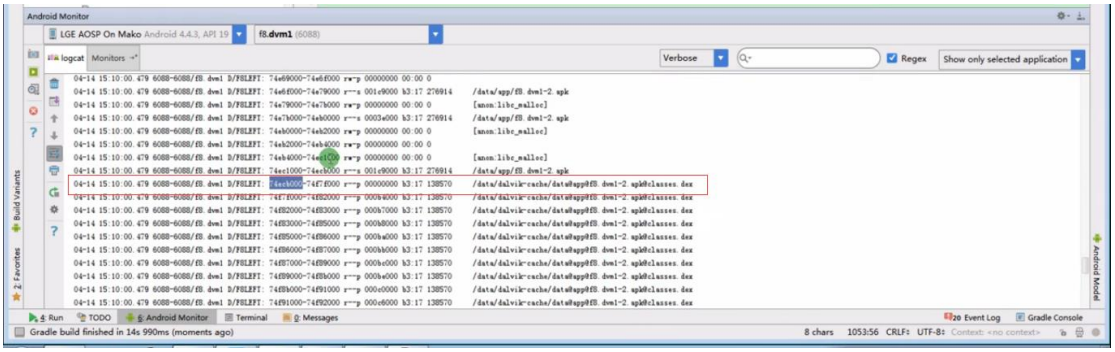
```
void* get_module_base(pid_t pid, const char* module_name)
{
    FILE *fp;
    long addr = 0;
    char *pch;
    char filename[32];
    char line[1024];
    if (pid < 0) {
        /* self process */
        snprintf(filename, sizeof(filename), "/proc/self/maps", pid);
    } else {

    } else {
        snprintf(filename, sizeof(filename), "/proc/%d/maps", pid);
    }
    fp = fopen(filename, "r");
    if (fp != NULL) {
        while (fgets(line, sizeof(line), fp)) {
            LOGD(line);
            if (strstr(line, module_name)) {
                pch = strtok( line, "-" );
                addr = strtoul( pch, NULL, 16 );
                break;

    }
    fp = fopen(filename, "r");
    if (fp != NULL) {
        while (fgets(line, sizeof(line), fp)) {
            LOGD(line);
            if (strstr(line, module_name)) {
                pch = strtok( line, "-" );
                addr = strtoul( pch, NULL, 16 );
                break;
            }
        }
        fclose(fp) ;
    }
    return (void *)addr;
}
 vBase = (u1*)get_module_base(-1, "/data/dalvik-cache/data@app@f8.dvmmodify");
```

定位到 insns，修改

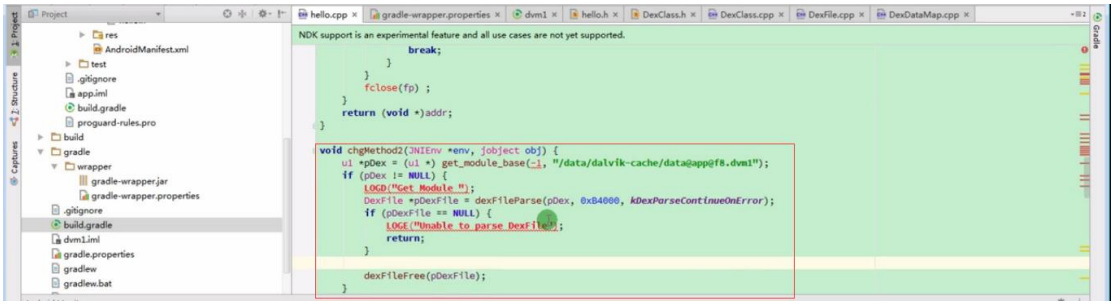
重置 Map 属性

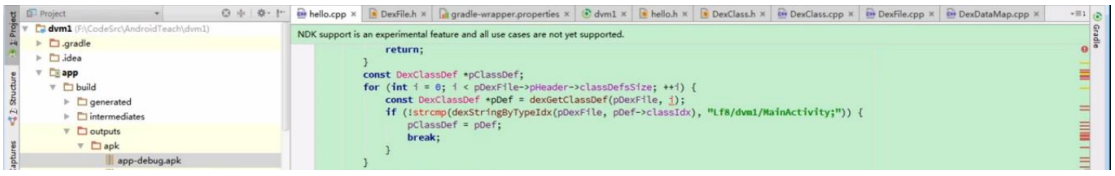


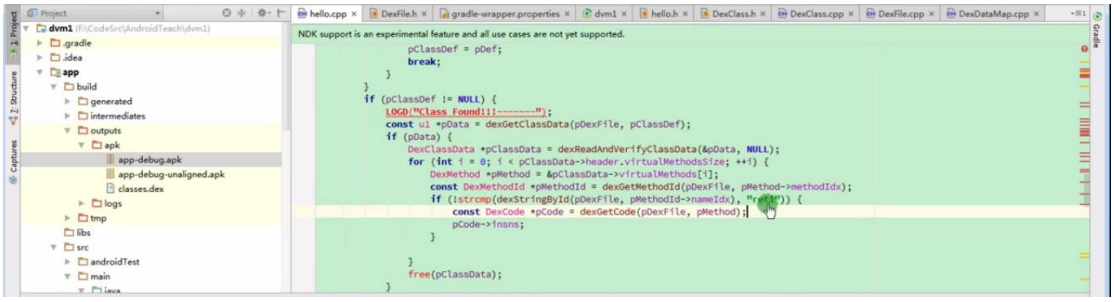
内存 Map 已经被打印出，找到 dex 的位置



解析出 Dex 文件





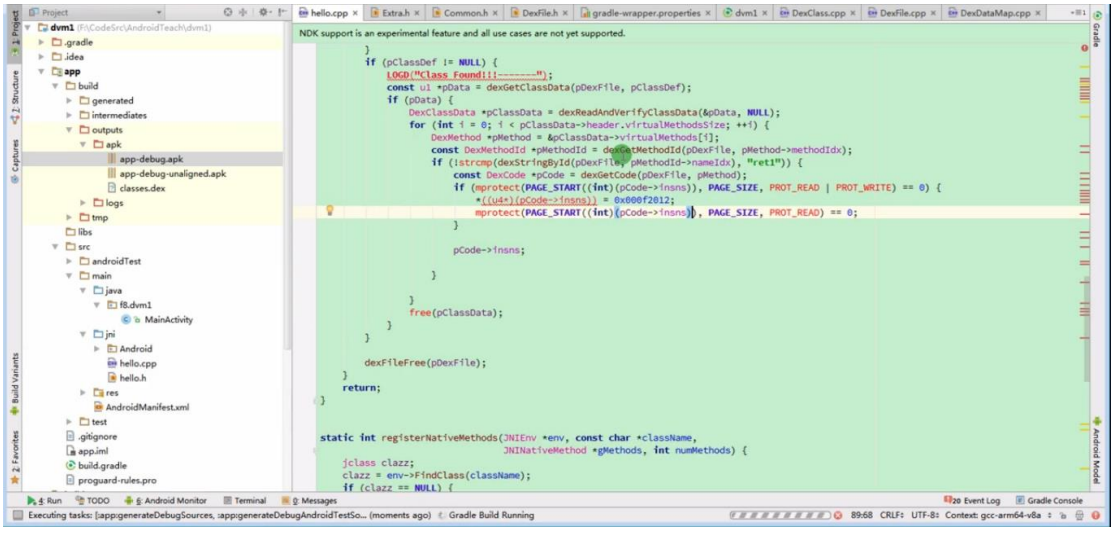


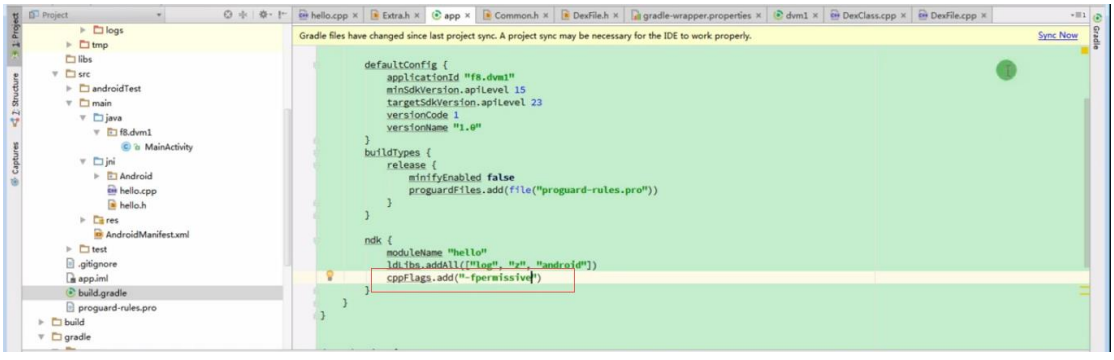
将 dex 的内存访问属性从只读修改为可写

重置 Map 属性

```
#include <asm-generic/mman-common.h>
#include <sys/mman.h>
#include <limits.h>
```

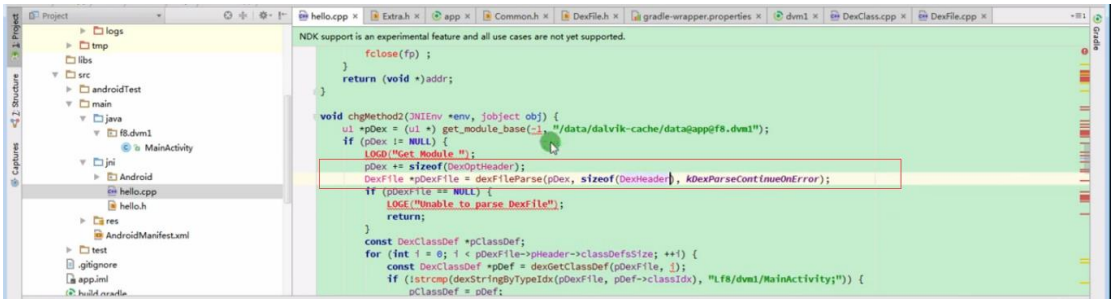
```
if (mprotect(PAGE_START((int) (pCode->insns)), PAGE_SIZE,PROT_READ |
PROT_WRITE | PROT_EXEC) == 0) {
    *(u4 *) (pCode->insns) = 0x000f2012;
    mprotect(PAGE_START((int) (pCode->insns)), PAGE_SIZE, PROT_READ |
PROT_EXEC);
}
```

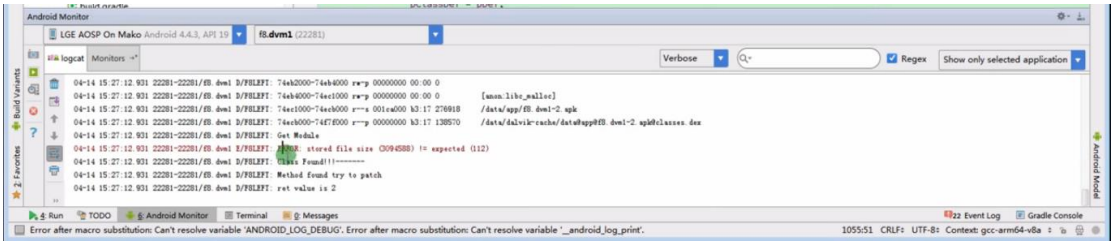




cppFlag 添加编译选项

编译报错，数据太大，需要进行修改





最后修改成功

3. **修改方法定位**

dexClassDef 遍历，获取 MethodId，对比 MethodName 与 proto，获取目标 Method，然后对相应的 DexCode 进行修改。

由于 Dex 加载到内存中是只有只读权限的，所以需要修改内存页的权限才能正常地修改 DexCode 数据。

具体解析 Dex 结构的方法可以参考：https://github.com/F8LEFT/FDA