Android o-llvm混淆

自我介绍

- 姓名：刘蒙(admeng.liu)
- 部门：大住宿
- 简要介绍：android研发

目录

一.为何用llvm混淆

Android开发中经常需要对敏感信息进行加密，避免不了要将密钥存放在终端设备上，那么如何防止密钥被逆向出来呢？这是一个先有鸡还是先有蛋的悖论。相比较将密钥写在Java层，将其下移到NDK层是个更好的选择，但是ndk就安全？

```c
demo.c
#include <jni.h>
#include <stdio.h>
#define NULL 0
void  printstr(){
    printf("abc");
}
JNIEXPORT jint JNICALL JNI_OnLoad(JavaVM* vm, void* reserved)  {
    JNIEnv *env = NULL;
    if ((*vm)->GetEnv(vm, (void**) &env, JNI_VERSION_1_6) != JNI_OK)
        return JNI_ERR;
    if (env == NULL)
        return JNI_ERR;
    printstr();
    return JNI_VERSION_1_6;
}
```

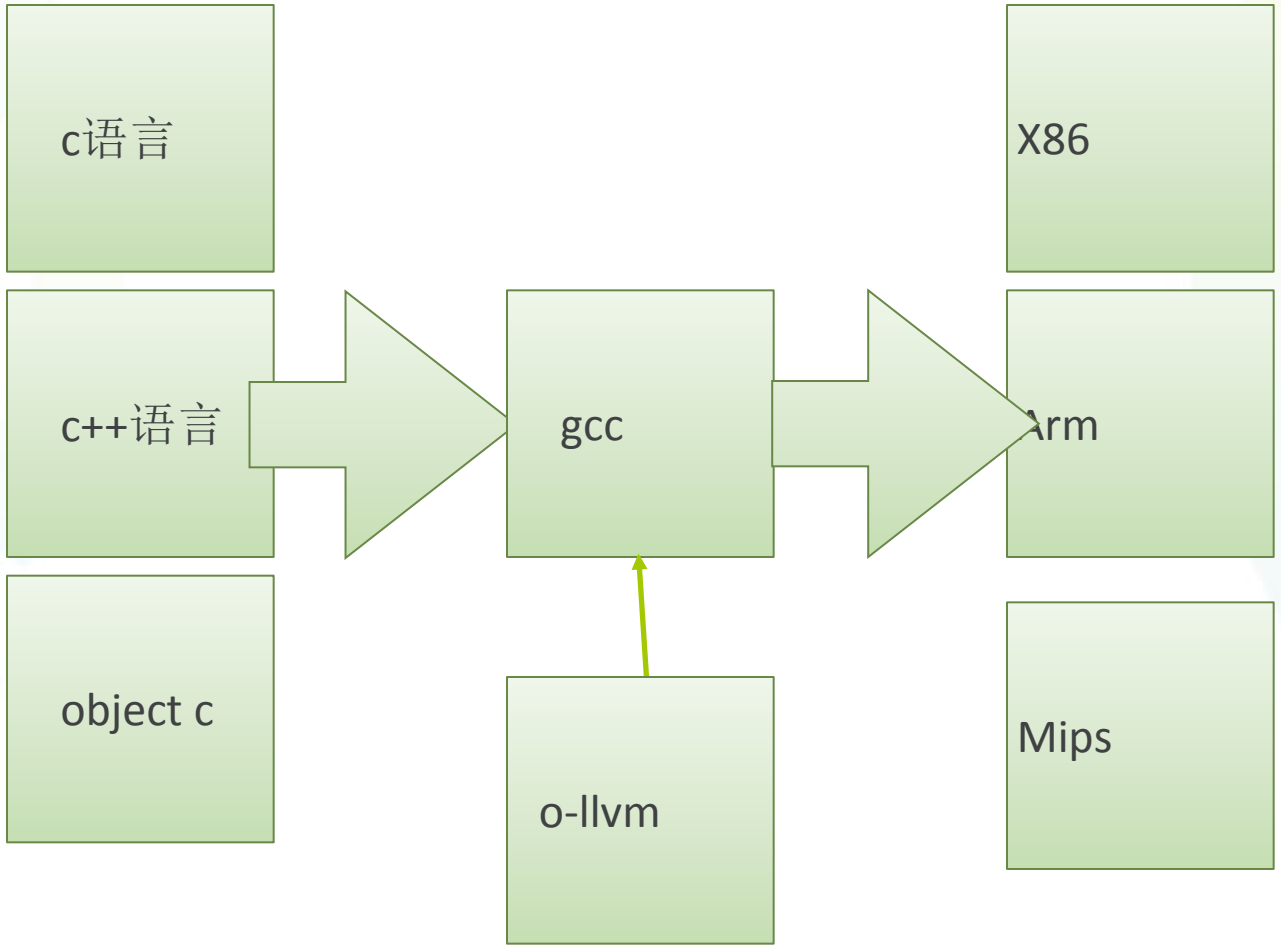
```
58
58 ; .text          ends
58
1C5C ; ===========================================
1C5C
1C5C ; Segment type: Pure data
1C5C                AREA .rodata, DATA, RE
1C5C                ; ORG 0x1C5C
1C5C unk_1C5C       DCB 0x61 ; a
1C5C
1C5D                DCB 0x62 ; b
1C5E                DCB 0x63 ; c
1C5F                DCB    0
1C5F ; .rodata      ends
1C5F
0001C60 ; ===========================================
```

我们可以看出ndk也没有那么安全，所以我们要针对ndk进行llvm混淆

```c
demo.c
#include <jni.h>
#include <stdio.h>
#define NULL 0
void  printstr(){
    printf("abc");
}
JNIEXPORT jint JNICALL JNI_OnLoad(JavaVM* vm, void* reserved)  {
    JNIEnv *env = NULL;
    if ((*vm)->GetEnv(vm, (void**) &env, JNI_VERSION_1_6) != JNI_OK)
        return JNI_ERR;
    if (env == NULL)
        return JNI_ERR;
    printstr();
    return JNI_VERSION_1_6;
}
```

```
!9DC ; Segment type: Pure data
!9DC                AREA .rodata, DATA
!9DC                ; ORG 0x29DC
!9DC byte_29DC      DCB 0x2E
!9DC
!9DD byte_29DD      DCB 0xC
!9DE byte_29DE      DCB 0x3A
!9DF byte_29DF      DCB 0x31
!9E0 byte_29E0      DCB 0
!9E0 ; .rodata      ends
!9E0
```

llvm解释



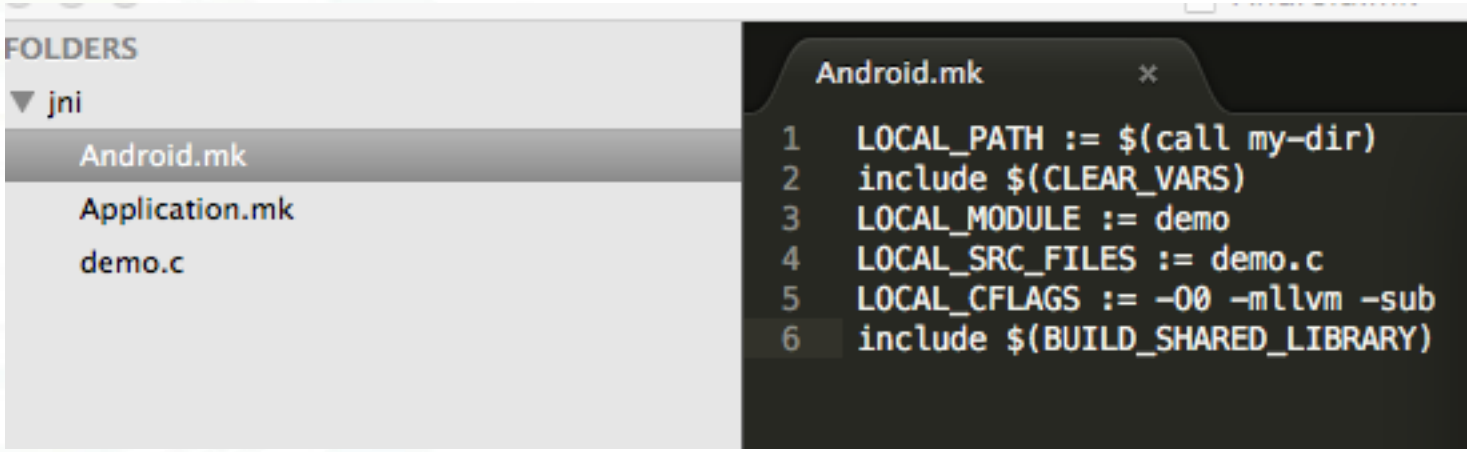
二、llvm使用

1.arm汇编指令

mov	位移指令
sub	减法指令
add	加法指令
and	与指令
or	或指令
xor	异或指令
push和pop	进栈和出栈

2、Instructions Substitution（指令替换）

等价的指令替换：例如a=b+c -> a=b-(-c)



实例说明

```
1  #include <jni.h>
2  #include <stdio.h>
3  #define NULL 0
4  int testsub(int a,int b){
5      a=a+a;
6      b=a-b;
7     return a-(-b);
```

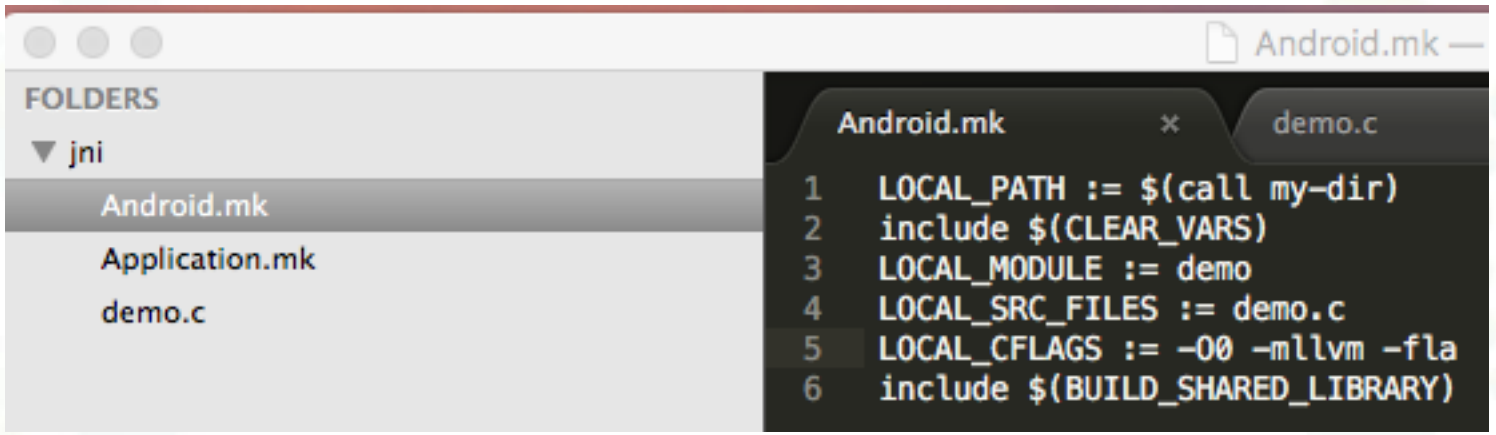
```
jint JNICALL JNI_OnLoad(
v *env = NULL;
*vm)->GetEnv(vm, (void**)
eturn JNI_ERR;
nv == NULL)
eturn JNI_ERR;
ub(2,1);
n JNI_VERSION_1_6;
```

```
                EXPORT testsub
testsub                                 ; CODE XREF
                                        ; DATA XREF

var_18          = -0x18
var_14          = -0x14
var_10          = -0x10

                PUSH    {R4,LR}
                SUB     SP, SP, #0x10
                ADD     R2, SP, #0x18+var_10
                PUSH    {R1}
                POP     {R3}
                PUSH    {R0}
                POP     {R4}
                STR     R0, [R2,#4]
                STR     R1, [R2]
                LDR     R0, [R2,#4]
                ADDS    R0, R0, R0
                STR     R0, [R2,#4]
                LDR     R1, [R2]
                SUBS    R0, R0, R1
                STR     R0, [R2]
                LDR     R1, [R2,#4]
                NEGS    R0, R0
                SUBS    R0, R1, R0
                STR     R4, [SP,#0x18+var_14]
                STR     R3, [SP,#0x18+var_18]
                ADD     SP, SP, #0x10
                POP     {R4,PC}
; End of function testsub
```

```
                EXPORT testsub
                                        ; CODE XREF: j_test
                                        ; DATA XREF: .got:t
testsub

var_18          = -0x18
var_14          = -0x14
var_10          = -0x10

                PUSH    {R4,LR}
                SUB     SP, SP, #0x10
                ADD     R2, SP, #0x18+var_10
                PUSH    {R1}
                POP     {R3}
                PUSH    {R0}
                POP     {R4}
                STR     R0, [R2,#4]
                STR     R1, [R2]
                LDR     R0, [R2,#4]
                ADDS    R0, R0, R0
                STR     R0, [R2,#4]
                LDR     R1, [R2]
                SUBS    R0, R0, R1
                STR     R0, [R2]
                LDR     R1, [R2,#4]
                NEGS    R0, R0
                LDR     R2, =0x2B160BBC
                ADDS    R1, R1, R2
                SUBS    R0, R1, R0
                LDR     R1, =0xD4E9F444
                ADDS    R0, R0, R1
                STR     R4, [SP,#0x18+var_14]
                STR     R3, [SP,#0x18+var_18]
                ADD     SP, SP, #0x10
                POP     {R4,PC}
; End of function testsub
```

3、Control Flow Flattening（平展控制

指令平展化：if这种条件变成switch控制条件



实例说明

```
4                       EXPORT testfla
4 testfla                                       ; CODE XRE
4                                               ; DATA XRE
4
4 var_20                = -0x20
4 var_1C                = -0x1C
4 var_18                = -0x18
4 var_14                = -0x14
4
4                       PUSH    {R4,LR}
6                       SUB     SP, SP, #0x18
8                       ADD     R2, SP, #0x20+var_14
A                       PUSH    {R1}
C                       POP     {R3}
E                       PUSH    {R0}
0                       POP     {R4}
2                       STR     R0, [R2,#4]
4                       STR     R1, [R2]
6                       LDR     R0, [R2,#4]
8                       CMP     R0, #1
A                       STR     R2, [SP,#0x20+var_18]
C                       STR     R3, [SP,#0x20+var_1C]
E                       STR     R4, [SP,#0x20+var_20]
0                       BNE     loc_E3C
2                       B       loc_E34
4 ; ---------------------------------------------------
4
4 loc_E34                                       ; CODE XRE
4                       LDR     R0, [SP,#0x20+var_18]
6                       LDR     R1, [R0,#4]
8                       STR     R1, [R0,#8]
A                       B       loc_E44
C ; ---------------------------------------------------
C
C loc_E3C                                       ; CODE XRE
C                       LDR     R0, [SP,#0x20+var_18]
E                       LDR     R1, [R0]
0                       STR     R1, [R0,#8]
2                       B       loc_E44
4 ; ---------------------------------------------------
4
4 loc_E44                                       ; CODE XRE
4                                               ; testfla+
4                       LDR     R0, [SP,#0x20+var_18]
6                       LDR     R0, [R0,#8]
8                       ADD     SP, SP, #0x18
A                       POP     {R4,PC}
```

demo.c — jni

id.mk × demo.c ×

```
include <jni.h>
include <stdio.h>
define NULL 0
t testfla(int a,int b){
    if(a==1){
        return a;
    }else{
        return b;
    }

IEXPORT jint JNICALL JNI_OnLoad(JavaVM* v
    JNIEnv *env = NULL;
    if ((*vm)->GetEnv(vm, (void**) &env, JN
        return JNI_ERR;
    if (env == NULL)
        return JNI_ERR;
    testfla(2,1);
    return JNI_VERSION_1_6;
```

```
4                       EXPORT testfla
4 testfla                                       ; CODE XREF:
4                                               ; DATA XREF:
4
4 var_34                = -0x34
4 var_30                = -0x30
4 var_2C                = -0x2C
4 var_28                = -0x28
4 var_24                = -0x24
4 var_20                = -0x20
4 var_1C                = -0x1C
4
4                       PUSH    {R4,LR}
6                       SUB     SP, SP, #0x2C
8                       ADD     R2, SP, #0x34+var_1C
A                       PUSH    {R1}
C                       POP     {R3}
E                       PUSH    {R0}
0                       POP     {R4}
2                       STR     R0, [R2,#8]
4                       STR     R1, [R2,#4]
6                       LDR     R0, [R2,#8]
8                       STR     R0, [R2,#0x10]
A                       LDR     R0, =0xFE346BD7
C                       STR     R0, [R2]
E                       STR     R2, [SP,#0x34+var_20]
0                       STR     R3, [SP,#0x34+var_24]
2                       STR     R4, [SP,#0x34+var_28]
4                       B       loc_E36
6 ; ---------------------------------------------------
6
6 loc_E36                                       ; CODE XREF:
6                                               ; testfla:lo
6                       LDR     R0, [SP,#0x34+var_20]
8                       LDR     R1, [R0]
A                       LDR     R2, =0x4EDB9CF6
C                       CMP     R1, R2
E                       STR     R1, [SP,#0x34+var_2C]
0                       BGT     loc_E62
2                       B       loc_E44
4 ; ---------------------------------------------------
4
4 loc_E44                                       ; CODE XREF:
4                       LDR     R0, =0xFE346BD7
6                       LDR     R1, [SP,#0x34+var_2C]
8                       CMP     R1, R0
```

4、Bogus Control Flow(虚假的控制流)

- **指令虚假化：增加虚假的代码模块**
- **1.参数-perBCF=20其应用于所有函数，其概率为20%。默认值:100**
- **2.参数-boguscf-loop=3在一个函数上应用它3次。默认值:1**
- **3.参数-boguscf-prob=40一个基本的组将会被混淆，概率为40%。默认值:30**

例子

Android.mk | demo.c

```
1  #include <jni.h>
2  #include <stdio.h>
3  #define NULL 0
4  int testbcf(int a,int b){
5      return a+10;
6  }
7  JNIEXPORT jint JNICALL JNI_OnLoad(JavaVM*
8      JNIEnv *env = NULL;
9      if ((*vm)->GetEnv(vm, (void**) &env, J
10         return JNI_ERR;
11     if (env == NULL)
12         return JNI_ERR;
```

```
f(2,1);
 JNI_VERSION_1_6;
```

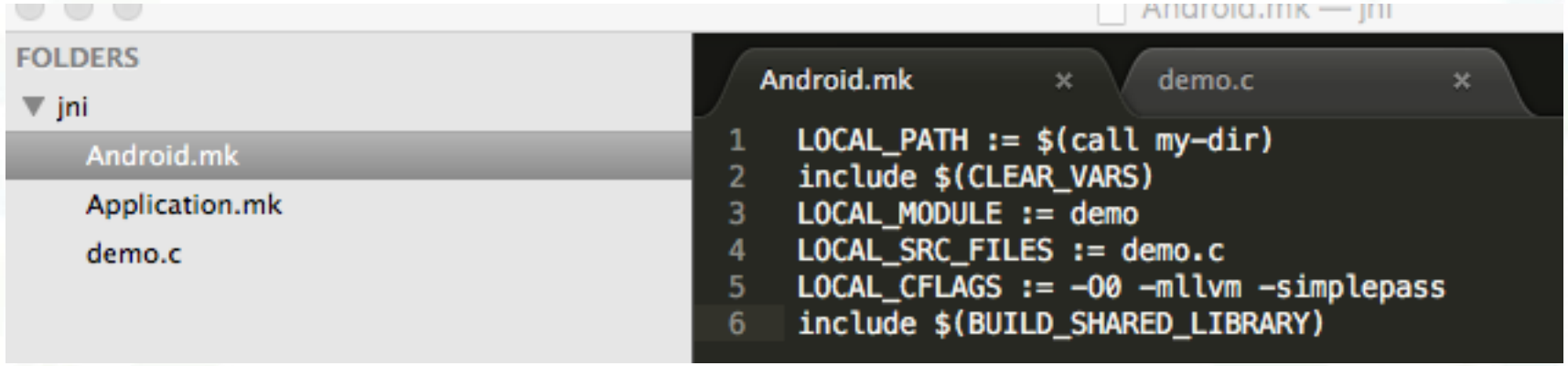
```
                EXPORT testbcf
testbcf                                 ; CODE XREF
                                        ; DATA XREF

var_18          = -0x18
var_14          = -0x14
var_10          = -0x10

                PUSH    {R4,LR}
                SUB     SP, SP, #0x10
                ADD     R2, SP, #0x18+var_10
                PUSH    {R1}
                POP     {R3}
                PUSH    {R0}
                POP     {R4}
                STR     R0, [R2,#4]
                STR     R1, [R2]
                LDR     R0, [R2,#4]
                ADDS    R0, #0xA
                STR     R4, [SP,#0x18+var_14]
                STR     R3, [SP,#0x18+var_18]
                ADD     SP, SP, #0x10
                POP     {R4,PC}
; End of function testbcf
```

```
                EXPORT testbcf
8C testbcf                              ; CODE XREF:
8C                                      ; DATA XREF:
8C
8C var_18       = -0x18
8C var_14       = -0x14
8C var_10       = -0x10
8C var_7        = -7
8C
8C              PUSH    {R4-R7,LR}
8E              ADD     R7, SP, #0xC
90              SUB     SP, SP, #0x14
92              MOV     R6, SP
94              LDR     R2, =(x_ptr - 0x3F80)
96              LDR     R3, =(_GLOBAL_OFFSET_TABLE_
98              ADD     R3, PC ; _GLOBAL_OFFSET_TABL
9A              ADDS    R2, R2, R3 ; x_ptr
9C              LDR     R2, [R2] ; x
9E              LDR     R2, [R2]
A0              LDR     R4, =(y_ptr - 0x3F80)
A2              ADDS    R3, R4, R3 ; y_ptr
A4              LDR     R3, [R3] ; y
A6              LDR     R3, [R3]
A8              SUBS    R4, R2, #1
AA              MULS    R4, R2
AC              MOVS    R2, #1
AE              TST     R4, R2
B0              STR     R1, [R6,#0x20+var_10]
B2              STR     R0, [R6,#0x20+var_14]
B4              STR     R3, [R6,#0x20+var_18]
B6              BEQ     loc_EC2
B8              B       loc_EBA
BA ; ---------------------------------------------
BA
BA loc_EBA                              ; CODE XREF:
BA              LDR     R0, [R6,#8]
BC              CMP     R0, #9
BE              BGT     loc_F1C
C0              B       loc_EC2
C2 ; ---------------------------------------------
C2
C2 loc_EC2                              ; CODE XREF:
C2                                      ; testbcf+34
C2              MOV     R0, SP
```

)0000E8C: testbcf (Synchronized with Hex View-1)

5、自定义pass

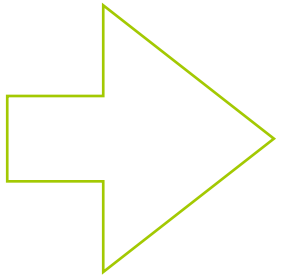
(1) simplepass



例子

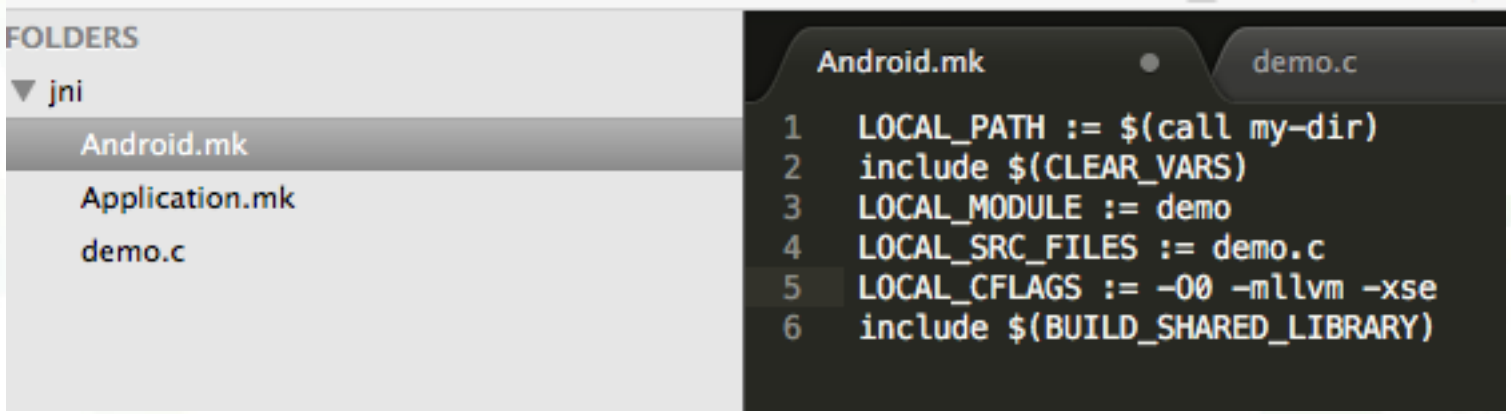

```
1   #include <jni.h>
2   #include <stdio.h>
3   #define NULL 0
4   int testsimplepass(int a,int b){
5       return a+b;
6   }
7   JNIEXPORT jint JNICALL JNI_OnLoad(JavaVM* vm, void* reserved)  {
8       JNIEnv *env = NULL;
9       if ((*vm)->GetEnv(vm, (void**) &env, JNI_VERSION_1_6) != JNI_
10          return JNI_ERR;
            L)
            _ERR;
            (2,1);
            SION_1_6;
```

```
18 ; =============== S U B R O U T I N E ===============
18
18
18                  EXPORT testsimplepass
18 testsimplepass                       ; CODE XREF
18                                      ; DATA XREF
18
18 var_18           = -0x18
18 var_14           = -0x14
18 var_10           = -0x10
18
18                  PUSH    {R4,LR}
1A                  SUB     SP, SP, #0x10
1C                  ADD     R2, SP, #0x18+var_10
1E                  PUSH    {R1}
20                  POP     {R3}
22                  PUSH    {R0}
24                  POP     {R4}
26                  STR     R0, [R2,#4]
28                  STR     R1, [R2]
2A                  LDR     R0, [R2,#4]
2C                  ADDS    R0, R0, R1
2E                  STR     R3, [SP,#0x18+var_14]
30                  STR     R4, [SP,#0x18+var_18]
32                  ADD     SP, SP, #0x10
34                  POP     {R4,PC}
34 ; End of function testsimplepass
34
```



```
18
18
18                  EXPORT testsimplepass
18 testsimplepass                       ; CODE XREF:
18                                      ; DATA XREF:
18
18 var_18           = -0x18
18 var_14           = -0x14
18 var_10           = -0x10
18
18                  PUSH    {R4,LR}
1A                  SUB     SP, SP, #0x10
1C                  ADD     R2, SP, #0x18+var_10
1E                  PUSH    {R1}
20                  POP     {R3}
22                  PUSH    {R0}
24                  POP     {R4}
26                  STR     R0, [R2,#4]
28                  STR     R1, [R2]
2A                  LDR     R0, [R2,#4]
2C                  NEGS    R1, R1
2E                  SUBS    R0, R0, R1
30                  STR     R3, [SP,#0x18+var_14]
32                  STR     R4, [SP,#0x18+var_18]
34                  ADD     SP, SP, #0x10
36                  POP     {R4,PC}
```

(2).字符串pass


FOLDERS
jni
Android.mk
Application.mk
demo.c
Android.mk
demo.c
1 LOCAL_PATH := $(call my-dir)
2 include $(CLEAR_VARS)
3 LOCAL_MODULE := demo
4 LOCAL_SRC_FILES := demo.c
5 LOCAL_CFLAGS := -O0 -mllvm -xse
6 include $(BUILD_SHARED_LIBRARY)

例子

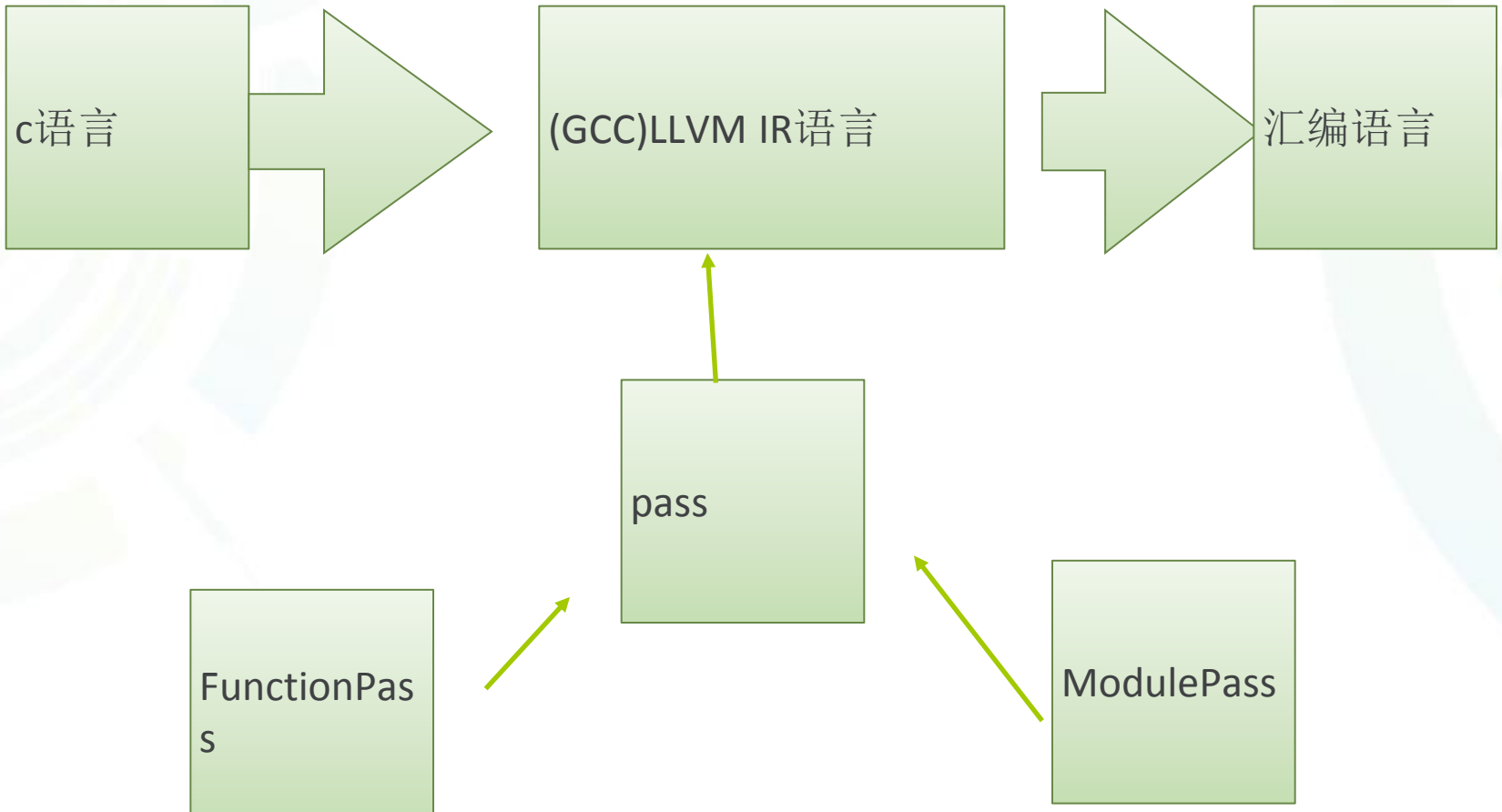

```c
#include <jni.h>
#include <stdio.h>
#define NULL 0
void testxse(){
    printf("123");
}
JNIEXPORT jint JNICALL JNI_OnLoad(JavaV
    JNIEnv *env = NULL;
    if ((*vm)->GetEnv(vm, (void**) &env
        return JNI_ERR;
    if (env == NULL)
        return JNI_ERR;
    testxse();
    return JNI_VERSION_1_6;
}
```

```
; =============== S U B R O U T I N E ===============

                EXPORT testxse
testxse                                 ; CODE XREF: j
                                        ; DATA XREF: .

var_C           = -0xC

; FUNCTION CHUNK AT .text:00002420 SIZE 00000002 BYTES

                PUSH    {R7,LR}
                SUB     SP, SP, #8
                LDR     R0, =(unk_25A4 - 0x3F7C)
                LDR     R1, =(_GLOBAL_OFFSET_TABLE_ -
                ADD     R1, PC ; _GLOBAL_OFFSET_TABLE_
                ADDS    R0, R0, R1 ; unk_25A4
                BL      loc_2420
; ---------------------------------------------------------
                STR     R0, [SP,#0x10+var_C]
                ADD     SP, SP, #8
                POP     {R7,PC}
; End of function testxse
```

```
                EXPORT testxse
testxse                                 ; CODE XREF: j
                                        ; DATA XREF: .

var_28          = -0x28
var_24          = -0x24
var_20          = -0x20
var_1C          = -0x1C

; FUNCTION CHUNK AT .text:00002474 SIZE 00000002 BYTES

                PUSH    {R4,R5,R7,LR}
                SUB     SP, SP, #0x20
                ADD     R0, SP, #0x30+var_20
                LDR     R1, =(__stack_chk_guard_ptr -
                LDR     R2, =(_GLOBAL_OFFSET_TABLE_ -
                ADD     R2, PC ; _GLOBAL_OFFSET_TABLE_
                ADDS    R1, R1, R2 ; __stack_chk_guard
                LDR     R1, [R1] ; __stack_chk_guard
                LDR     R3, [R1]
                STR     R3, [R0]
                LDR     R3, =(byte_25F8 - 0x3F7C)
                ADDS    R2, R3, R2 ; byte_25F8
                LDRB    R3, [R2]
                MOVS    R4, #0x4F
                EORS    R3, R4
                ADD     R4, SP, #0x30+var_1C
                ADDS    R4, #3
                STRB    R3, [R4]
                LDRB    R3, [R2,#(byte_25F9 - 0x25F8)]
                MOVS    R5, #0x6E
                EORS    R3, R5
                STRB    R3, [R4,#1]
                LDRB    R3, [R2,#(byte_25FA - 0x25F8)]
                MOVS    R5, #0x59
                EORS    R3, R5
                STRB    R3, [R4,#2]
                LDRB    R3, [R2,#(byte_25FB - 0x25F8)]
                MOVS    R5, #0x31
                EORS    R3, R5
                STRB    R3, [R4,#3]
                LDRB    R2, [R2,#(byte_25FC - 0x25F8)]
                MOVS    R3, #0x36
                EORS    R2, R3
                STRB    R2, [R4,#4]
```

三.混淆原理

- 在编译期间通过pass获取LLVM IR语言指令集进行逻辑操作



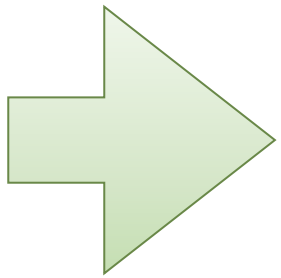
1.o-llvm系统编译环境搭建

- (1)centos 6.x
- (2) python2.7
- (3)gcc4.8.2
- (4)glibc2.14
- (5)下载git源码 https://github.com/obfuscator-llvm/obfuscator.git

2.llvm IR语法

@	全局标识符
%	局部标识符
i32	32位的整型
i32*	指向32位整数的指针，也就是4个字节
alloca	在栈上分配内存
align 4	按4字节对齐
load和store	装载和写入内存
add	加法指令
sub	减法指令
br	跳转指令
switch	条件

```
int main() {
    int a=6,b=9;
    return a+b;
}
```



```
define i32 @main() #0 {
entry:
  %retval = alloca i32, align 4
  %a = alloca i32, align 4
  %b = alloca i32, align 4
  store i32 0, i32* %retval
  store i32 6, i32* %a, align 4
  store i32 9, i32* %b, align 4
  %0 = load i32, i32* %a, align 4
  %1 = load i32, i32* %b, align 4
  %add = add nsw i32 %0, %1
  ret i32 %add
}
```

3.functionpass

- (1) 覆写构造函数

```
FunctionPass (char &pid)
```

- (2)覆写runOnFunction函数

```
virtual bool   runOnFunction (Function &F)=0
```

4.modulePass

- (1) 覆写构造函数

```
ModulePass (char &pid)
```

- (2)覆写runOnModule函数

```
virtual bool   runOnModule (Module &M)=0
```

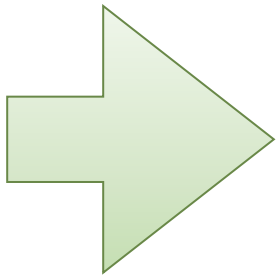
5.简单pass编写

```
#include "llvm/Transforms/Obfuscation/SimplePass.h"

using namespace llvm;

namespace {
   struct SimplePass : public FunctionPass {
      static char ID; // Pass identification, replacement for typeid

      SimplePass() : FunctionPass(ID) {}

      bool runOnFunction(Function &F) override {
         return false;
      }
char SimplePass::ID = 0;
static RegisterPass<SimplePass> X("test", "simple pass");
Pass *llvm::createSimplePass() { return new SimplePass(); }
```

四.大客户端混淆效果

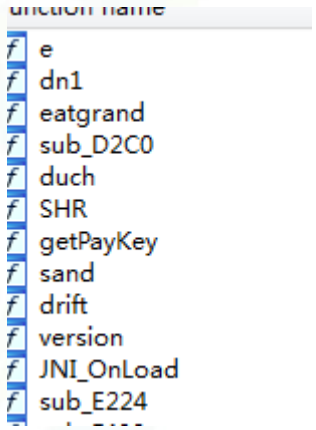
(1)字符串混淆部分

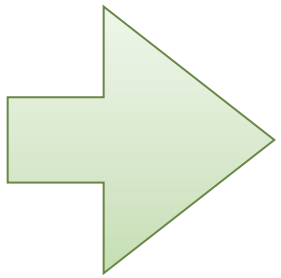
odata:0010C...	00000039	C	(Ljava/lang/Object;Ljava/lang/Object;)Lj
odata:0010CE...	00000005	C	size
odata:0010CE...	00000006	C	UTF-8
odata:0010CE...	00000023	C	{%s,\"deviceInfo\":{%s,%s,%s,%s,%s}}
odata:0010CE...	00000005	C	info
odata:0010CE...	0000000D	C	base64 error
odata:0010CE...	00000013	C	/system/build.prop
odata:0010CE...	0000000E	C	/default.prop
odata:0010CE...	00000015	C	/system/default.prop
odata:0010CE...	00000011	C	/data/local.prop
odata:0010CE...	0000000E	C	/proc/cpuinfo
odata:0010CE...	0000000E	C	/proc/meminfo
odata:0010CE...	0000000E	C	/proc/version
odata:0010CE...	0000001D	C	/sys/class/net/wlan0/address
odata:0010CE...	0000001C	C	/sys/class/net/eth0/address
odata:0010CF...	00000036	C	/sys/devices/system/cpu/cpu0/cpufreq/
odata:0010CF3F	00000014	C	pm list packages -3

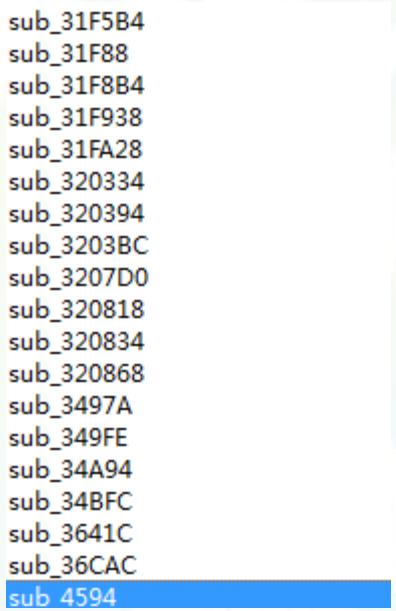


ddress	Length	Type	String
.text:0003749B	00000005	C	\"P@Q@
.text:000374AB	00000005	C	CGH.I
.text:0003756F	00000009	C	(vCA'i2?\x1B
.text:000375BC	0000000A	C	j2?\x1BwCA'!F
.text:000375EF	00000005	C	`hF\b8
.text:00037601	00000005	C	FjF\b:
.text:00037607	00000007	C	F\najF\b:
.text:0003760F	00000007	C	FJajF\b:
.text:00037619	00000005	C	ajF\b:
.text:00037621	00000005	C	ajF\b:
.text:00037627	00000007	C	F\nbjF\b:
.text:0003762F	00000007	C	FJbjF\b:
.text:00037639	00000005	C	bjF\b:
.text:00037641	00000005	C	bjF\b:
.text:00037647	00000007	C	F\ncjF\b:
.text:0003769D	00000007	C	Y(C!%h@
.text:000376A9	00000005	C	TXx7%
.text:000376DF	00000007	C	Y\bCV!H@
.text:0003773F	00000007	C	Y\bC8!H@
.text:0003775D	00000005	C	y.!dF

(2)函数混淆部分

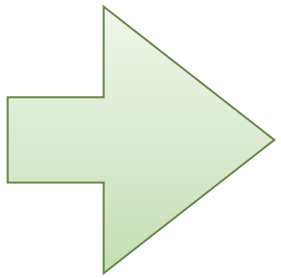

e
dn1
eatgrand
sub_D2C0
duch
SHR
getPayKey
sand
drift
version
JNI_OnLoad
sub_E224




sub_31F5B4
sub_31F88
sub_31F8B4
sub_31F938
sub_31FA28
sub_320334
sub_320394
sub_3203BC
sub_3207D0
sub_320818
sub_320834
sub_320868
sub_3497A
sub_349FE
sub_34A94
sub_34BFC
sub_3641C
sub_36CAC
sub_4594

(3)ida的F5功能失效

```
int __fastcall JNI_OnLoad(_DWORD *a1)
{
  _DWORD *v1; // r4@1
  signed int v2; // r1@1
  signed int v3; // r0@4
  signed int v4; // r0@7
  int v5; // r0@24
  char v6; // r1@24
  int v7; // r0@33
  _DWORD *v9; // [sp+0h] [bp-18h]@2
  _DWORD *v10; // [sp+4h] [bp-14h]@24
  char v11; // [sp+8h] [bp-10h]@7

  v1 = a1;
  v2 = 397169556;
  do
  {
    while ( 1 )
    {
      while ( 1 )
      {
        while ( 1 )
```



Unexplored Instructi

IDA View-A Pseu

```
void sub_4594()
{
  JUMPOUT(0);
}
```

去哪儿? Qunar.Com 聪明你的旅行 | IT大咖说 知识分享平台

Q&A

去哪儿? Qunar.Com 聪明你的旅行