

ARM-Tracer & ARM-Analyzer

# Tutorial



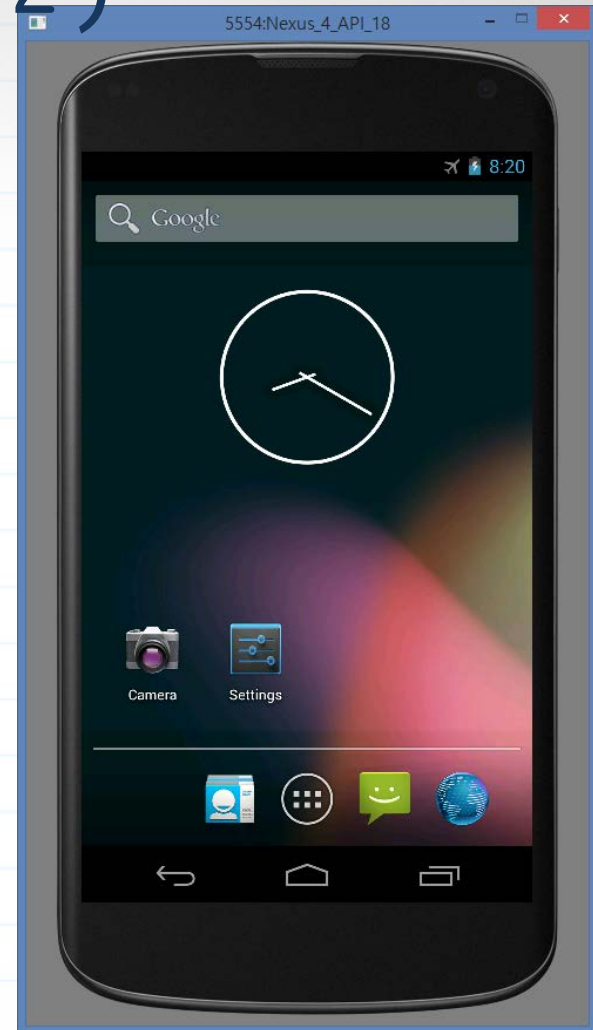
# Contents

1. Test Environment
2. Installing App
3. Checking Crash
4. Tracing Instructions
5. Analyzing Trace Log
6. Verifying

# 1. Test Environment ( 1 / 2 )

## Emulator

- Hardware : Nexus 4 / 4.7 / 768x1280 / xhdpi
- System Image : Jelly Bean / API 18  
armeabi-v7a / Android 4.3
- Use Host GPU ( X )
- Store a snapshot for faster startup ( X )
- RAM : 1536 / VM heap : 64
- Internal Storage : 4 GB / SD card : 1 GB
- Enable keyboard input ( 0 )





# 1. Test Environment ( 2 / 2 )

## Real Device

### 1) Device 1

- Model : LG-E960 ( Nexus4 )
- Android Version : 4.3.1
- Rooting ( 0 )

### 2) Device 2

- Model : SHV-E330S ( Galaxy S4 LTE-A )
- Android Version : 4.4.2
- Rooting ( 0 )



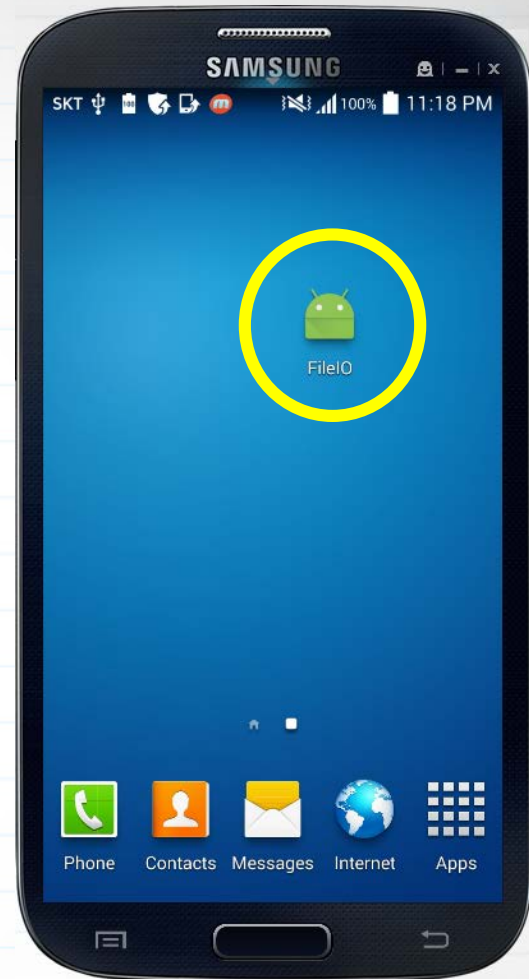
## 2. Installing App

```
C:\Windows\system32\cmd.exe

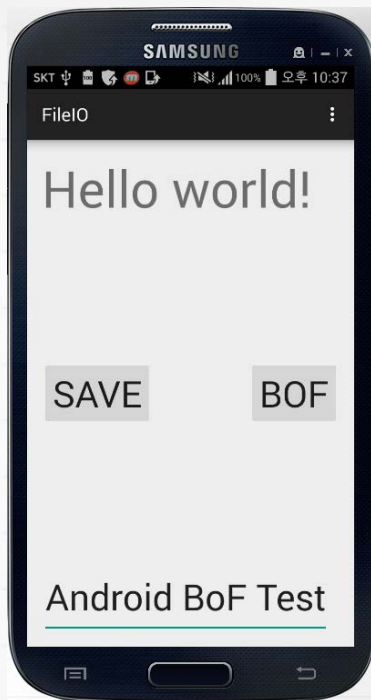
D:\TEMP>adb install SampleApp.apk
8077 KB/s <565629 bytes in 0.068s>
  pkg: /data/local/tmp/SampleApp.apk
Success

D:\TEMP>
```

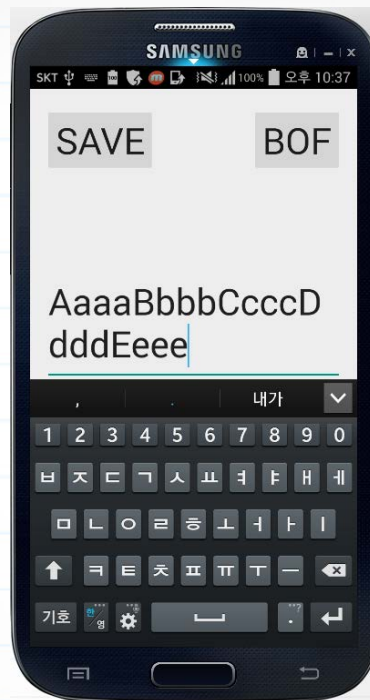
```
cmd> adb install SampleApp.apk
```



### 3. Checking Crash



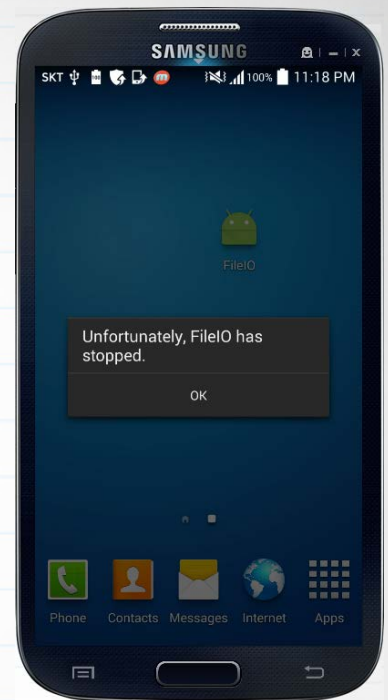
1) Launch App



2) Type  
"AaaaBbbbCccc  
DdddEeee"

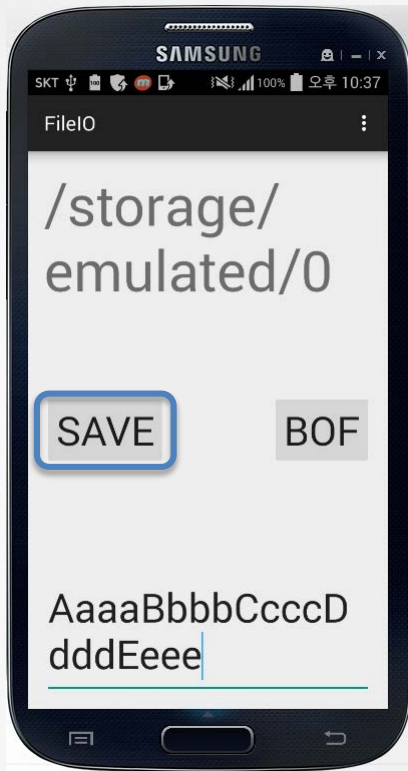


3) Click [SAVE]  
Button



4) Click [BOF]  
Button

## 4. Tracing Instructions ( 1 / 2 )



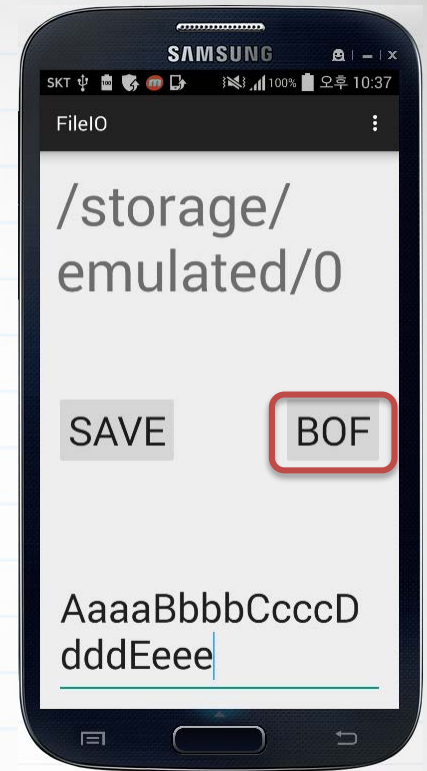
```
C:\Windows\system32\cmd.exe - adb shell
root@ks01lteskt:/data/local/tmp #
root@ks01lteskt:/data/local/tmp # ll
-rwxrwxrwx root root 3354452 2015-11-02 22:42 ARM-Tracer
root@ks01lteskt:/data/local/tmp #
root@ks01lteskt:/data/local/tmp # ps | grep file
root 236 2 0 0 c0566c6c 00000000 $ file-storage
u0_a319 23367 349 964668 31792 ffffffff 400d78f8 $ xyz.whoya.fileio
root@ks01lteskt:/data/local/tmp #
root@ks01lteskt:/data/local/tmp # ./ARM-Tracer -p 23367 -i .txt
target : .txt
attach program pid : 23367
this program pid : 24078

>>> Skip Dump Line : 0
# Find it! __Open Func : 400D6508
# Find it! __pthread_clone Func : 400D7A04

@ [__open] ins : e1a0c007 : e1a0c007
@ [__open + 4] ins : e3a07005 : e3a07005

[libc.sol] Memory : 400B6000 - 400FE000

[*] Attaching Thread List - /proc/23367/task
[-] 23367 - Name: yz.whoya.fileio
State: t (tracing stop)
```



1) Click [SAVE]

Button



2) Run ARM-Tracer

# ./ARM-Tracer -i [SampleApp Pid] -i .txt

3) Click [BOF]

Button





## 4. Tracing Instructions ( 2 / 2 )

```

e r0<790acde8> : /storage/emulated/0/test.txt
- r1<00020000> : 131072
- * - * - * - * - LOGGING START - * - * - * - * -
< 2>-[SWI] sys_open : /storage/emulated/0/test.txt
< 269>-[SWI] sys_fstat64, sys_oabi_fstat64
< 554>-[SWI] sys_read
< 648>-[SWI] sys_read
[#: READ_ADDR : BEFBB0E0 - BEFBB0F3 / READ_SIZE : 20<20>
[-] READ_DATA : <00000000> 61616141 62626242 63636343 64646444 65656545

< 795>-[SWI] sys_close

<9> --- [23367 / b7f<2943>] IS THIS CRASH ?
<0> : "SIGSEGV /* Segmentation violation (ANSI). */" ---

r00=0xBEFBB0B4 r01=0xBEFBB1BC r02=0x00000000 r03=0x68686848
r04=0x72764FC0 r05=0x41F32E48 r06=0x00000000 r07=0x41B33DA4
r08=0xBEFBB220 r09=0x41B33D9C r10=0x41F32E58 r11=0x65656545
r12=0x72764FD4 sp=0xBEFBB0C0 lr=0x72761D74 pc=0x66666644
cpsr=0x400D0010

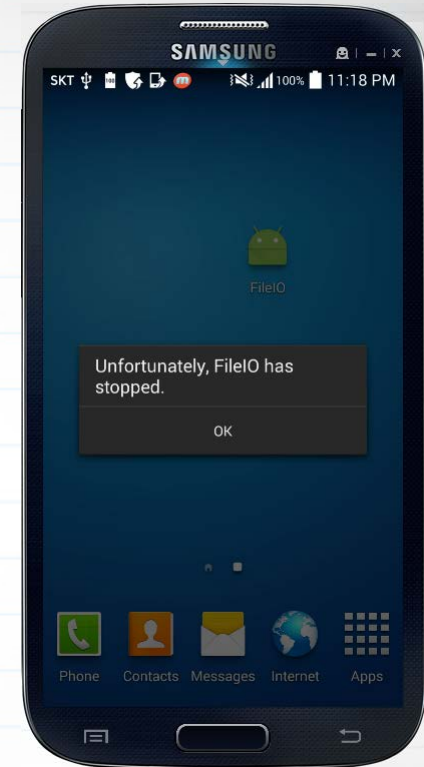
[*] CRASH PC : 0x66666644 00 00 F0 01
- * - * - * - * - LOGGING FINISH - * - * - * - * -

FINAL status : b7f

Total Time : 0.000000
Total Instructions : 2088
Total Pass Instruction : 72 = 48<ARM> + 24<Thumb>
# of Instruction executed : Inf
# of Lock Handler called : 8

Input Data File : /storage/emulated/0/test.txt
Input Data Size : 20
Input Data Memory : BEFBB0E0 - BEFBB0F3
Process PID, Target TID : 23367, 23367
Killed
137!root@ks0ilteskt:/data/local/tmp # ll
-rwxrwxrwx root root 3354452 2015-11-02 22:42 ARM-Tracer
-rw----- root root 334080 2015-11-02 22:54 dump.log
root@ks0ilteskt:/data/local/tmp #

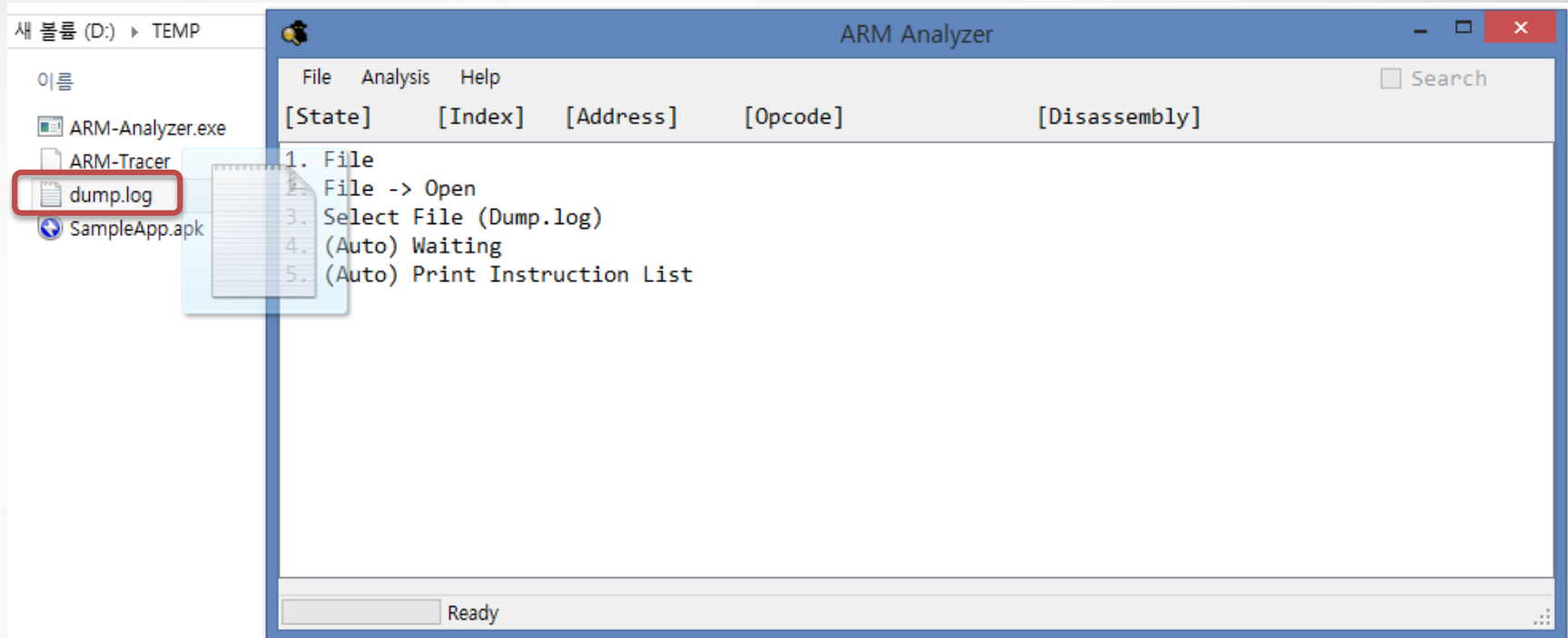
```



### 3) Check Log File – dump.log

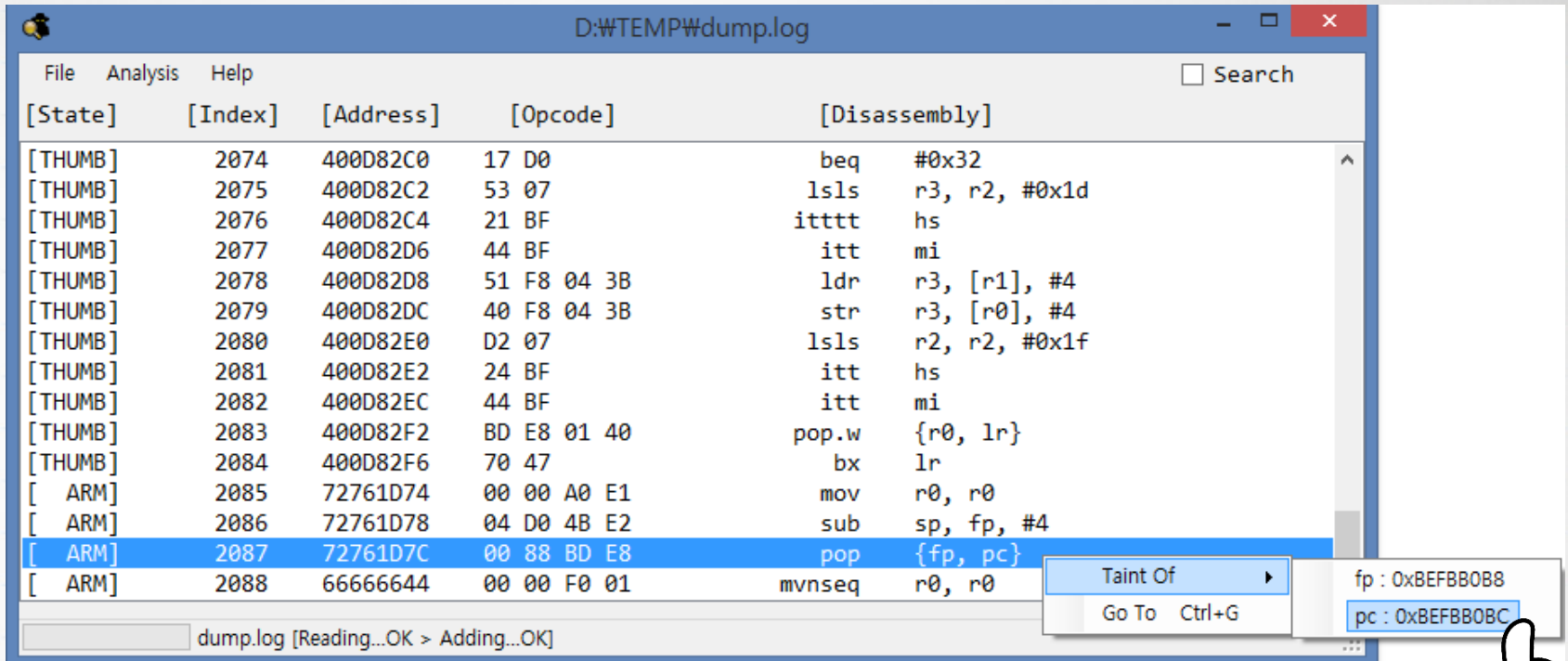


## 5. Analyzing Trace Log ( 1 / 3 )



- 1) Pull out trace log file  
`cmd> adb pull /data/local/tmp/dump.log`
- 2) Open the trace log file with ARM-Analyzer  
You can just Drag & Drop “dump.log” 😊

## 5. Analyzing Trace Log ( 2 / 3 )



D:\TEMP\dump.log

[State]	[Index]	[Address]	[Opcode]	[Disassembly]
[THUMB]	2074	400D82C0	17 D0	beq #0x32
[THUMB]	2075	400D82C2	53 07	lsls r3, r2, #0x1d
[THUMB]	2076	400D82C4	21 BF	itttt hs
[THUMB]	2077	400D82D6	44 BF	itt mi
[THUMB]	2078	400D82D8	51 F8 04 3B	ldr r3, [r1], #4
[THUMB]	2079	400D82DC	40 F8 04 3B	str r3, [r0], #4
[THUMB]	2080	400D82E0	D2 07	lsls r2, r2, #0x1f
[THUMB]	2081	400D82E2	24 BF	itt hs
[THUMB]	2082	400D82EC	44 BF	itt mi
[THUMB]	2083	400D82F2	BD E8 01 40	pop.w {r0, lr}
[THUMB]	2084	400D82F6	70 47	bx lr
[ARM]	2085	72761D74	00 00 A0 E1	mov r0, r0
[ARM]	2086	72761D78	04 D0 4B E2	sub sp, fp, #4
[ARM]	2087	72761D7C	00 88 BD E8	pop {fp, pc}
[ARM]	2088	66666644	00 00 F0 01	mvnseq r0, r0

dump.log [Reading...OK > Adding...OK]

Taint Of  
Go To Ctrl+G

fp : 0xBEFBB0B8  
pc : 0xBEFBB0BC

3) You can start taint analysis at “pop” instruction

Mouse Right Button → Choose “pc” operand

## 5. Analyzing Trace Log ( 3 / 3 )

TaintResult

OK SAVE

pc : 0xBEFBB0BC  
[ ARM ] 2087 72761D7C 00 88 BD E8 pop {fp, pc}

[Dst] : [Src]	[State]	[Index]	[Address]	[Opcode]	[Disassembly]
r3 : 0xBEFBB0E8	Found! Probably Exploitable? //			0x00000008 = 8	
r3 : 0xBEFBB0E8	[ ARM ]	1222	72761E84	00 30 D3 E5	ldrb r3, [r3]
r3 : r3	[ ARM ]	1223	72761E88	05 30 83 E2	add r3, r3, #5
r2 : r3	[ ARM ]	1224	72761E8C	73 20 EF E6	uxtb r2, r3
0xBEFBB14C : r2	[ ARM ]	1230	72761EA4	00 20 C3 E5	strb r2, [r3]
r2 : 0xBEFBB14C	[ THUMB ]	1518	400D9340	51 F8 04 2B	ldr r2, [r1], #4
0xBEFBB1B0 : r2	[ THUMB ]	1529	400D936C	E0 E8 02 23	strd r2, r3, [r0], #8
r3 : 0xBEFBB1B0	[ ARM ]	1727	72761EF8	00 30 D3 E5	ldrb r3, [r3]
r3 : r3	[ ARM ]	1728	72761EFC	02 30 43 E2	sub r3, r3, #2
r2 : r3	[ ARM ]	1729	72761F00	73 20 EF E6	uxtb r2, r3
0xBEFBB14C : r2	[ ARM ]	1735	72761F18	00 20 C3 E5	strb r2, [r3]
r2 : 0xBEFBB14C	[ THUMB ]	2012	400D9340	51 F8 04 2B	ldr r2, [r1], #4
0xBEFBB1B0 : r2	[ THUMB ]	2023	400D936C	E0 E8 02 23	strd r2, r3, [r0], #8
d17H : 0xBEFBB1B0	[ THUMB ]	2071	400D82B4	61 F9 8D 0A	vld1.32 {d16, d17}, [r1]!
0xBEFBB0BC : d17H	[ THUMB ]	2072	400D82B8	40 F9 8D 0A	vst1.32 {d16, d17}, [r0]!
pc : 0xBEFBB0BC	[ ARM ]	2087	72761D7C	00 88 BD E8	pop {fp, pc}

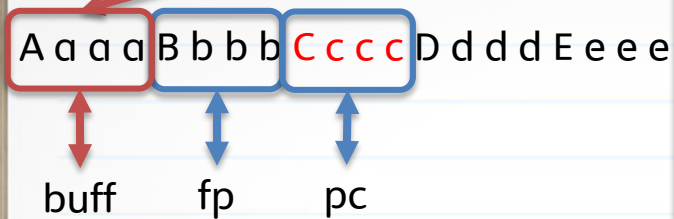
4) Taint Result

5) Get File Offset – 8

A a a a B b b b C c c c D d d d E e e e

## 6. Verifying

Input Data :



Taint Result Chain

ldrb	r3, [r3]
add	r3, r3, #5
uxtb	r2, r3
strb	r2, [r3]
ldr	r2, [r1], #4
strd	r2, r3, [r0], #8
ldrb	r3, [r3]
sub	r3, r3, #2
uxtb	r2, r3
strb	r2, [r3]
ldr	r2, [r1], #4
strd	r2, r3, [r0], #8
vld1.32	{d16, d17}, [r1]!
vst1.32	{d16, d17}, [r0]!
pop	{fp, pc}

```
void bof(char * big, int num) {  
    char buff[4] = {0, };  
    memcpy(buff, big, num);  
    return;  
}
```

```
JNIEXPORT void JNICALL Java_xyz_whoya_fileio_MainActivity_BoF  
(JNIEnv * env, jobject thisObj, jstring inJNISTR)  
{
```

```
    const char *inCStr = (*env)->GetStringUTFChars(env, inJNISTR, NULL);
```

```
    FILE *fd = fopen( inCStr, "r" );
```

```
    char a[100] = {0,};
```

```
    char b[100] = {0,};
```

```
    char c[100] = {0,};
```

```
    int i, size = 0;
```

```
    size = fread( a, 1, 100, fd );
```

```
    fclose( fd );
```

```
    for ( i = 0 ; i < size ; i++ ) {
```

```
        b[i] = a[i] + 5;
```

```
    }
```

```
    strcpy(c, b);
```

```
    for ( i = 0 ; i < size ; i++ ) {
```

```
        b[i] = c[i] - 2;
```

```
    }
```

```
    strcpy(c, b);
```

```
    bof(c, size);
```

```
}
```



# Thank you!

- Tutorial : Tutorial.pdf
- Sample App : SampleApp.apk
- Sample App Project : SampleApp\_AndroidStudio1.4\_Project.zip
- ARM-Tracer : ARM-Tracer
- ARM-Analyzer : ARM-Analyzer.exe ( Word Size Tracking )
- Trace Log : dump.log