

So 文件调试记录:

cmd 输入:

1.Adb forward tcp:23900 tcp:23900 (打开端口转发)

```
C:\Users\zhukeyu>adb forward tcp:23900 tcp:23900
```

2.Adb shell (打开安卓的 shell)

2.1Cd /data/local/tmp (放着一些常用的需要的工具)

2.2./as64 -p23900

```
C:\Users\zhukeyu>adb shell
sailfish:/ # cd /data/local/tmp
sailfish:/data/local/tmp # ./as64 -p23900
IDA Android 64-bit remote debug server(ST) v1.22. Hex-Rays (c) 2004-2017
Listening on 0.0.0.0:23900...
=====
[1] Accepting connection from 127.0.0.1...
```

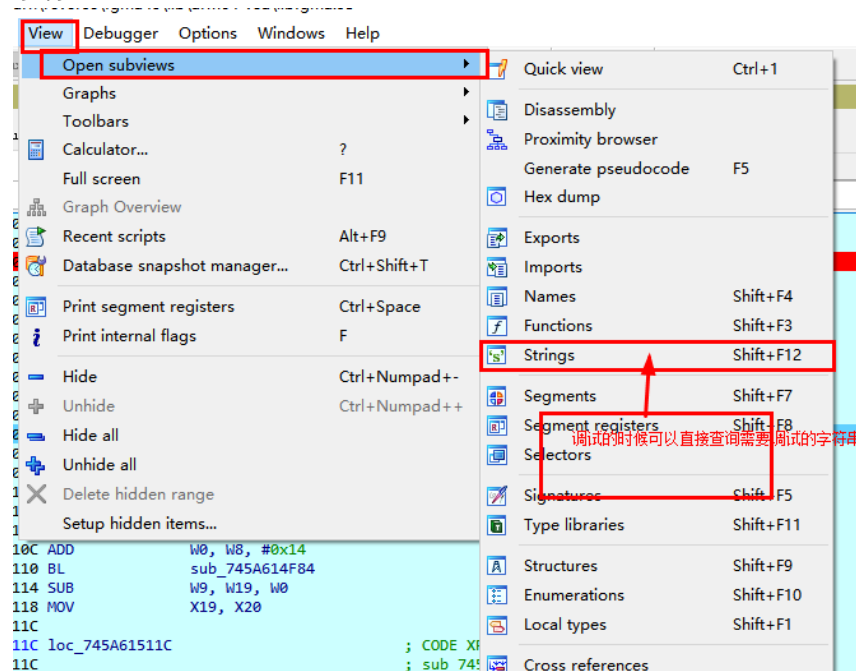
3.Cd /sdcard (存放包的路径)

/data/local/tmp/tcpdump -l wlan0 -s 0 -w fg\_1.pcap(使用 tcpdump 进行抓包)

Adb pull /sdcard/fa\_1.pcap


IDA 双开, 一个为了静态调试, 一个为了动态过程

使用 ida 打开 so 文件, 等待加载完成, 打开 view->open subviews->strings, 可以进行查询字符串。



# 调试步骤

目标：获取 IP 池

 libfgma.so

使用 ida 加载 so 文件,进行 so 文件逆向分析。

Addroid 逆向自由门 app 之静态分析

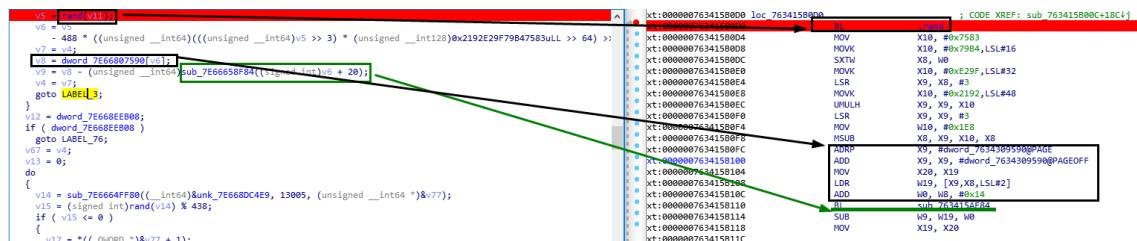
1.先定位到 connect 函数

connect 函数的原型：int connect(int sockfd, const struct sockaddr\* server\_addr, socklen\_t addrlen) 第二个参数是 ip 地址

```
}
LABEL_3:
v93 = v9;
connect(v69, &v92, 16LL);
FD_SET_chk(v69, &v94, 128LL);
++v4;
if ( (signed int)v69 > (signed int)v70 )
    v3 = v69 + 1;
```

2.继续向上查找，第二个参数是从哪里获取的？

从下图可以看出 v9=v8 - sub\_93f84(), v8 是表 dword\_242590 中索引为 v6 的值。



```
v70 = v3;
v10 = socket(2LL, 1LL, 0LL);
*(DWORD *)&v79[4 * v4] = v10;
if ( v10 == -1 )
    return sub_8B094(v66, &byte_29368F);
v69 = v10;
v11 = sub_9AFE4();
v92 = -1157562366;
if ( !(v68 & 1) )
{
    v5 = rand(v11);
    v6 = v5;
    - 488 * ((unsigned __int64)((unsigned __int64)v5 >> 3) * (unsigned __int128)0x2192E29F79B47583uLL >> 64) >> 3);
    v7 = v4;
    v8 = dword_242590[v6];
    v9 = v8 - (unsigned __int64)sub_93f84((signed int)v6 + 20);
    v4 = v7;
    goto LABEL_3;
}
v12 = dword_329B08;
```

3.跳转进函数 sub\_93f84(), 发现返回值其实是另一张表的值,且索引值是上一张表格加 20

```
v8 = dword_7E66807590[v6];
v9 = v8 - (unsigned __int64)sub_7E66658F84((signed int)v6 + 20);
v4 = v7;
goto LABEL_3;
}
```

```
signed __int64 __fastcall sub_7E66658F84(int a1)
{
    signed int v1; // w0
    signed __int64 result; // x0
    int v3; // w19

    v1 = a1 - 10000 * ((1759218605LL * a1 >> 44) + ((unsigned __int64)(1759218605LL * a1) >> 63));
    if ( v1 < 1 || result = dword_7E66807590[v1], !(_DWORD)result )
    {
        result = (unsigned int)v1;
        if ( v1 )
        {
            if ( v1 == 1 )
            {
                result = 1LL;
            }
            else
            {
                v3 = sub_7E66658F84(v1 - 1);
                result = (unsigned int)sub_7E66658F84(v1 - 2) + v3;
            }
        }
        dword_7E66807590[v1] = result;
    }
    return result;
}
```

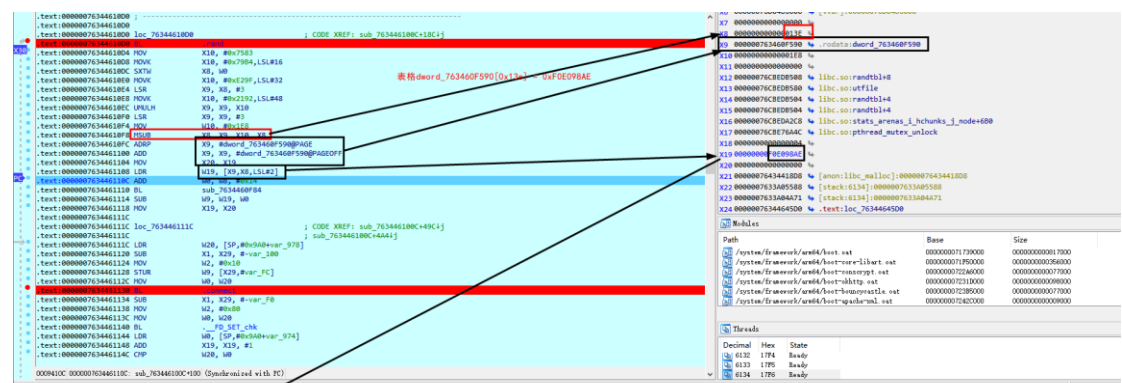
```
xt:000000763415AF84 var_10 = -0x10
xt:000000763415AF84 var_50 = 0
xt:000000763415AF84 ; unwind {
xt:000000763415AF84 STR X21, [X0, #0x10+var_20]!
xt:000000763415AF84 STP X20, X19, [X0, #0x20+var_10]
xt:000000763415AF84 STP X29, X30, [X0, #0x20+var_50]
xt:000000763415AF84 ADD X29, SP, #0x20
xt:000000763415AF84 MOV W0, #0x80AD
xt:000000763415AF84 MOV W0, #0x80AD, LSL#16
xt:000000763415AF84 SHADDL X0, W0, W0, XZR
xt:000000763415AF84 LSR X9, X8, #0x3F
xt:000000763415AF84 AGR X0, W0, #0x2C
xt:000000763415AF84 ADD W0, W0, W0
xt:000000763415AF84 MOV W0, #0x2710
xt:000000763415AF84 MSUB W20, W0, W0, W0
xt:000000763415AF84 ADRP X21, #unk_763436E70@PAGE
xt:000000763415AF84 CMP W20, #1
xt:000000763415AF84 ADD X21, X21, #unk_763436E70@PAGEOFF
xt:000000763415AF84 B.LT loc_763415AFCC
xt:000000763415AF84 LDR W0, [X21, W20, 5XTW#2]
xt:000000763415AF84 CBNZ W0, loc_763415AFCC
xt:000000763415AFCC ; CODE XREF: sub_763415AF84+3Cfj
xt:000000763415AFCC MOV W0, W20
xt:000000763415AFD0 CBZ W20, loc_763415AFF8
```

## Android 逆向自由门 app 之动态调试

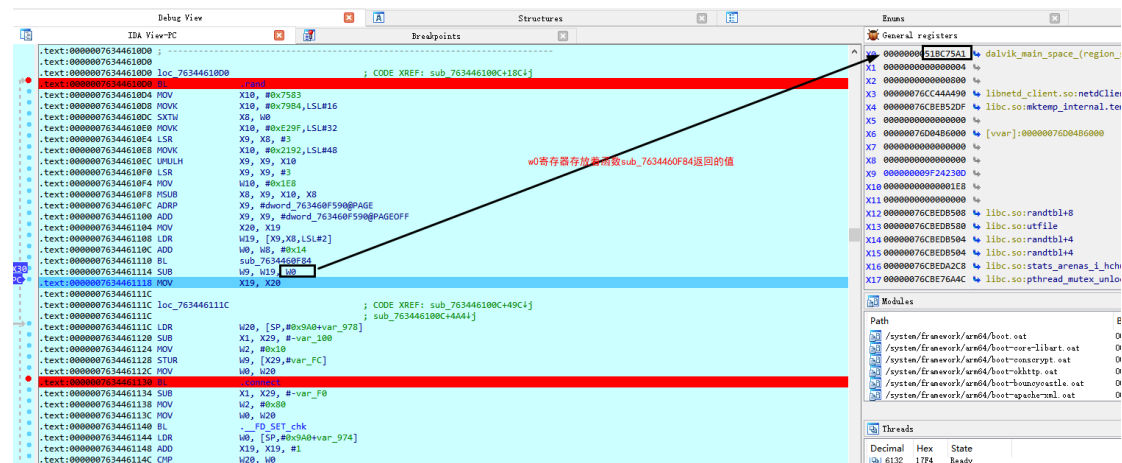
### 1. 设置两个断点

```
.text:00000745A6150D0 loc_745A6150D0 ; CODE XREF: sub_745A61500C+18C4j
.text:00000745A6150D0 BL ;rand
.text:00000745A6150D4 MOV X10, #0x7583
.text:00000745A6150D8 MOVK X10, #0x79B4,LSL#16
.text:00000745A6150DC SXTW X8, W0
.text:00000745A6150E0 MOVK X10, #0xE29F,LSL#32
.text:00000745A6150E4 LSR X9, X8, #3
.text:00000745A6150E8 MOVK X10, #0x2192,LSL#48
.text:00000745A6150EC UMIULH X9, X9, X10
.text:00000745A6150F0 LSR X9, X9, #3
.text:00000745A6150F4 MOV W10, #0x1E8
.text:00000745A6150F8 MSUB X8, X9, X10, X8
.text:00000745A6150FC ADRP X9, #dword_745A7C3590@PAGE
.text:00000745A615100 ADD X9, X9, #dword_745A7C3590@PAGEOFF
.text:00000745A615104 MOV X20, X19
.text:00000745A615108 LDR W19, [X9,X8,LSL#2]
.text:00000745A61510C ADD W0, W8, #0x14
.text:00000745A615110 BL sub_745A614F84
.text:00000745A615114 SUB W9, W19, W0
.text:00000745A615118 MOV X19, X20
.text:00000745A61511C loc_745A61511C ; CODE XREF: sub_745A61500C+49C4j
.text:00000745A61511C ; sub_745A61500C+4A41j
.text:00000745A61511C LDR W20, [SP,#0x9A0+var_978]
.text:00000745A615120 SUB X1, X29, #-var_100
.text:00000745A615124 MOV W2, #0x10
.text:00000745A615128 STUR W9, [X29,#var_FC]
.text:00000745A61512C MOV W0, W20
.text:00000745A615130 BL ;F0_Set_chk
.text:00000745A615134 SUB X1, X29, #-var_F0
.text:00000745A615138 MOV W2, #0x80
```

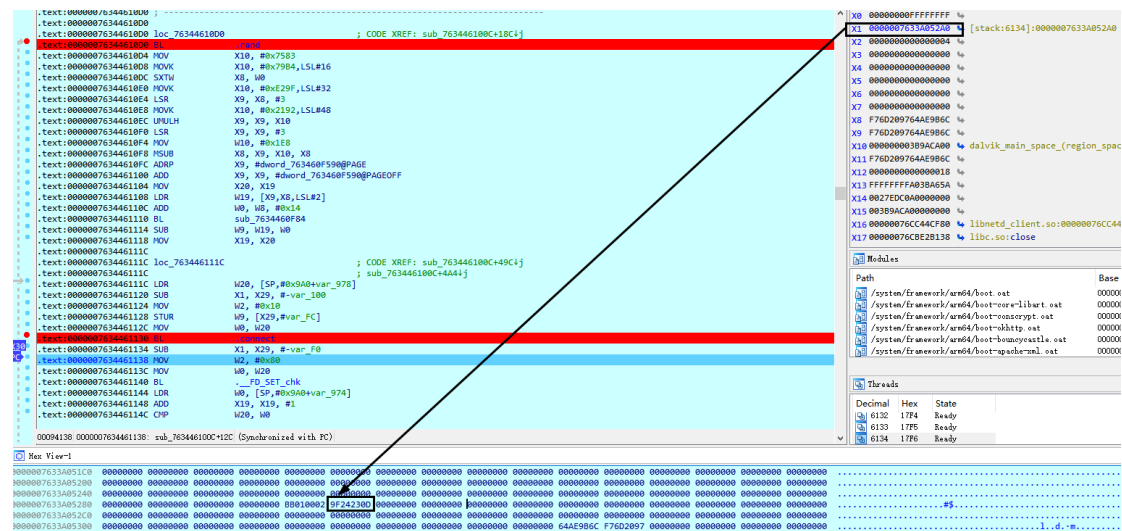
2. 点击运行，成功断在第一个断点处。继续单步执行，可以看到 x9 寄存器中存放的是从表 dword\_742590 获取的值。



3. 继续单步执行，然后单步步入函数 sub\_93f84()中，通过寄存器 x0 带回返回值。



4.  $0xf0e098ae - 0x51bc75a1 = 0x9f24230d$ , 执行到下一个断点, 查看第二个参数的值为  $0x9f24230d$ , 符合预期值。



5. 跳转到查看表格 `dword_242590` 和表格 `dword_31fe70` 查找到对应位置存放着该值, 然后计算 ip 池。  
可以看到表格为一个固定的含有 488 个元素的表格。并且由此可以推断 v6 的取值范围为 0-487



```

.bss:0000007DB92E3E70 ; unsigned int dword_7DB92E3E70[10000]
.bss:0000007DB92E3E70 dword_7DB92E3E70 DCD 0, 1, 1, 2, 3, 5, 8, 0xD, 0x15, 0x22, 0x37, 0x59, 0x90, 0xE9, 0x179
.bss:0000007DB92E3E70 ; DATA XREF: sub_7DB9057F84+3810
.bss:0000007DB92E3E70 ; sub_7DB9057F84+3810
.bss:0000007DB92E3E70 DCD 0x262, 0x3DB, 0x63D, 0xA18, 0x1055, 0x1A6D, 0x2AC2, 0x452F, 0x6FF1
.bss:0000007DB92E3E70 DCD 0xB520, 0x12511, 0x1DA31, 0x2FF42, 0x4D973, 0x7D8B5, 0xCB228, 0x148ADD
.bss:0000007DB92E3E70 DCD 0x213D05, 0x35C7E2, 0x5704E7, 0x8CCCC9, 0xE3D1B0, 0x1709E79, 0x2547029
.bss:0000007DB92E3E70 DCD 0x3C50EA2, 0x6197ECB, 0x9DE8D0D, 0xFF80C38, 0x19D699A5, 0x29CEA5DD
.bss:0000007DB92E3E70 DCD 0x43A53F82, 0x6D73E55F, 0xB11924E1, 0x1E8D0A40, 0xCFA62F21, 0xEE333961
.bss:0000007DB92E3E70 DCD 0xBDD96882, 0xAC0CA1E3, 0x69E60A65, 0x15F2AC48, 0x7FD886AD, 0x95C862F5
.bss:0000007DB92E3E70 DCD 0x15A419A2, 0xAB6F7C97, 0xC1139639, 0x6C8312D0, 0x2D96A909, 0x9A198BD9
.bss:0000007DB92E3E70 DCD 0xC7B064E2, 0x61CA208B, 0x297A859D, 0x8844A658, 0xB4BF2BF5, 0x4003D24D
.bss:0000007DB92E3E70 DCD 0xF4C2FE42, 0x34C6D08F, 0x2989CED1, 0x5E509F60, 0x87DA6E31, 0xE62B0D91
.bss:0000007DB92E3E70 DCD 0x6E0578C2, 0x54308953, 0xC2360515, 0x16668E68, 0xD89C937D, 0xEF0321E5
.bss:0000007DB92E3E70 DCD 0xC79FB562, 0xB6A2D747, 0x7E428CA9, 0x34E563F0, 0xB327F099, 0xE80D5489
.bss:0000007DB92E3E70 DCD 0x98354522, 0x834299A8, 0x1E77DEC0, 0xA1BA7878, 0xC0325745, 0x61ECCFB0
.bss:0000007DB92E3E70 DCD 0x221F2702, 0x840BF6BF, 0xA62B1DC1, 0x2A371480, 0xD0623241, 0xFA9946C1
.bss:0000007DB92E3E70 DCD 0xCABF7902, 0xC5948FC3, 0x909038C5, 0x5624F888, 0xE6B5314D, 0x3CDA29D5
.bss:0000007DB92E3E70 DCD 0x238F5822, 0x606984F7, 0x83F8E019, 0xE4626510, 0x685B4529, 0x4CBDA39
.bss:0000007DB92E3E70 DCD 0xB518EF62, 0x1D6999B, 0xB6EF88FD, 0xB8C62298, 0x6F85A895, 0x287BCE2D
.bss:0000007DB92E3E70 DCD 0x983179C2, 0xC0AD47EF, 0x58DEC1B1, 0x198C09A0, 0x726ACB51, 0x8BF6D4F1
.bss:0000007DB92E3E70 DCD 0xFE61A042, 0x8A587533, 0x88BA1575, 0x13128AA8, 0x9BCA01D, 0xAEDF2AC5
.bss:0000007DB92E3E70 DCD 0x4AABCAE2, 0xF98AF5A7, 0x4436C089, 0x3DC1B630, 0x81F876B9, 0xBFBA2CE9
.bss:0000007DB92E3E70 DCD 0x41B2A3A2, 0x16CD08B, 0x431F742D, 0x448C4488, 0x87ABB8E5, 0xCC37FD9D
.bss:0000007DB92E3E70 DCD 0x53E3B682, 0x2018B41F, 0x73FF6AA1, 0x941B1EC0, 0x81A8961, 0x9C35A821
.bss:0000007DB92E3E70 DCD 0xA4503182, 0x4085D9A3, 0xE4D60B25, 0x2558E4C8, 0xA31EFED, 0x2F8DD485
.bss:0000007DB92E3E70 DCD 0x39BFC4A2, 0x694D9957, 0xA30D5DF9, 0xC5AF750, 0xAF685549, 0x8BC34C99
.bss:0000007DB92E3E70 DCD 0x6B2BA1E2, 0x26EE7E7B, 0x921A905D, 0xB9097ED8, 0x4B240F35, 0x42D8E0D
.bss:0000007DB92E3E70 DCD 0x4F519D42, 0x537F2B4F, 0xA2D0C891, 0xF64FF3E0, 0x99208C71, 0x8F708051
.bss:0000007DB92E3E70 DCD 0x28916CC2, 0xB8021D13, 0xE09389D5, 0x9895A6E8, 0x792930BD, 0x11BED7A5
.bss:0000007DB92E3E70 DCD 0x8AE80862, 0x9CA6E007, 0x278EE869, 0xC435C870, 0xEBC480D9, 0xAFFA7949

```

6.由此就可以计算 ip 池

dword\_242590[idx] - dword\_31FE70[idx+20]就可以获取到 ip 池。

7.继续进行分析，可以看到此处代码分为了两个分支

```

do
{
    v70 = v3;
    v10 = socket(2LL, 1LL, 0LL);
    *(_DWORD *)&v79[4 * v4] = v10;
    if ( v10 == -1 )
        return sub_7E6650094(v66, &byte_7E668586BF);
    v69 = v10;
    v11 = sub_7E665FFE4();
    v92 = -1157562366;
    if ( !(v68 & 1) )
    {
        v5 = rand(v11);
        v6 = v5
            - 488 * ((unsigned __int64)((unsigned __int64)v5 >> 3) * (unsigned __int128)0x2192E29F79847583ULL >> 64) >> 3);
        v7 = v4;
        v8 = dword_7E66807590[v6];
        v9 = v8 - (unsigned __int64)sub_7E6658F84((signed int)v6 + 20);
        v4 = v7;
        goto LABEL_3; 1: 已经获取到了ip直接跳转进行连接
    }
    v12 = dword_7E668EEB08;
    if ( dword_7E668EEB08 )
        goto LABEL_76; 2: 没有获取到ip跳转到域名处
    v67 = v4;
    v13 = 0;
}

```

8.第一个分支为上面分析的获取 ip 然后建立连接，第二个分支为获取域名

```

LABEL_3:
    v93 = v9;
    connect(v69, &v92, 16LL);
    __FD_SET_chk(v69, &v94, 128LL); 知道ip的地址，connect进行连接，第二个参数为ip地址
    ++v4;
    if ( (signed int)v69 > (signed int)v70 )
        v3 = v69 + 1;
    else
        v3 = v70;
}

LABEL_76:
    v9 = dword_7E668EEB0C[dword_7E668EEB5C];
    dword_7E668EEB5C = dword_7E668EEB5C + 1 - (dword_7E668EEB5C + 1) / v12 * v12;
}
else
{
    v9 = 0;
}
}

```

9.第二个分支继续向上查找，可以看到域名也是从一个表格中获取的。

```

LABEL_58:
v35[v30] = 0;
v73 = v76;
v72 = *(_OWORD *)&v74;
if ( v77 & 1 )
    operator delete(v78);
if ( v72 & 1 )
    v38 = (char *)v73;
else
    v38 = (char *)v73 + 1;
v39 = gethostbyname(v38); 先获取域名
if ( v39 )
{
    v40 = *(_DWORD **)(v39 + 24);
    v41 = *v40;
    if ( *v40 )
    {
        v42 = 0LL;
        v43 = dword_7E668EEB08;
        v44 = &dword_7E668EEB08[dword_7E668EEB08];
        do
        {
            v44[v42++] = *v41;
            v41 = v40[(unsigned int)v42];
        }
        while ( v41 );
        dword_7E668EEB08 = v43 + v42;
    }
}
if ( v72 & 1 )
    operator delete(v73);
++v13;
}
while ( v13 != 2 );
v12 = dword_7E668EEB08;
v4 = v67;
if ( dword_7E668EEB08 )
{

```

## 10.修改指令进行动态调试

### 将 TBZ 指令修改为 TBNZ 指令

.text:0000007634464190	STUR	W8, [X29, #var_100]
.text:0000007634464194	LDR	W8, [SP, #0x9A0+var_97C]
.text:0000007634464198	TBZ	W8, #0, loc_76344640D0
.text:000000763446419C	ADRP	X8, #dword_76346F9B08@PAGE
.text:00000076344641A0	LDR	W8, [X8, #dword_76346F9B08@PAGEOFF]
.text:00000076344641A4	CBNZ	W8, loc_7634464480
.text:00000076344641A8	STR	X19, [SP, #0x9A0+var_988]
.text:00000076344641AC	MOV	W19, WZR
.text:00000076344641B0	B	loc_76344641C0
.text:00000076344641B4	;	
.text:00000076344641B4	loc_76344641B4	
.text:00000076344641B4	; CODE XREF: sub_7634	
.text:00000076344641B4	; sub_763446400C+424	
.text:00000076344641B4	ADD	W19, W19, #1
00094198 0000007634464198: sub_763446400C+18 (Synchronized with PC)		
α View-1		
07634464150	00 D4 94 1A 7F 16 00 F1 E0 1A 00 54 E0 2F 00 09	.J.....T...
07634464160	40 00 80 52 21 00 80 52 E2 03 1F 2A B1 D1 FF 97	@.R!..R...*....
07634464170	1F 04 00 31 E8 83 02 91 00 75 33 B8 00 2B 00 54	...1.....y3...+T
07634464180	E0 2B 00 B9 98 1B 00 94 48 00 80 52 2B 60 B7 72	.....H..R('..r
07634464190	A8 03 10 B8 E8 27 40 B9 83 F9 07 58 A8 14 00 00	.....6.....
076344641A0	08 09 48 B9 E8 16 00 35 F3 0F 00 F9 F3 03 1F 2A	..K...S.....
076344641B0	04 00 00 14 73 06 00 11 7F 0A 00 71 A0 15 00 54	...S.....q...T
076344641C0	00 14 00 F0 E8 03 02 91 A1 59 86 52 00 A4 13 91	.....Y..R....
076344641D0	6C D8 FF 97 47 CD FF 97 28 AD 84 52 08 B4 B2 72	1...G...(.R...r
076344641E0	08 7C 28 9B 08 FD 60 D3 08 01 00 08 09 7D 08 13	.)((...`.....).
076344641F0	29 7D 48 0B E8 03 42 39 CA 36 80 52 35 81 0A 1B	))H....9...RS...
07634464200	BF 02 00 71 4D 05 00 54 F8 E7 48 A9 16 7D 41 D3	...qM..T.....)A.
07634464210	1F 01 00 72 F8 03 1F 2A F4 02 99 9A D7 02 9B 9A	...r...*.....



```

.text:000007634464194 LDR     W8, [SP,#0x9A0+var_97C]
.text:000007634464198 TBNZ     W8, #0, loc_76344640D0
.text:00000763446419C ADRP     X8, #dword_76346F9B08@PAGE
.text:0000076344641A0 LDR      W8, [X8,#dword_76346F9B08@PAGEOFF]
.text:0000076344641A4 CBNZ     W8, loc_7634464480
.text:0000076344641A8 STR      X19, [SP,#0x9A0+var_988]
.text:0000076344641AC MOV      W19, WZR
.text:0000076344641B0 B        loc_76344641C0
.text:0000076344641B4 ;
.text:0000076344641B4
.text:0000076344641B4 loc_76344641B4 ; CODE XREF: sub_76344
.text:0000076344641B4 ; sub_763446400C+4241j
.text:0000076344641B4 ADD      W19, W19, #1
00094198 0000007634464198: sub_763446400C+18 (Synchronized with PC)

```

Hex View-1

```

307634464150 00 04 94 1A 7F 16 00 F1 E0 1A 00 54 E0 2F 00 89 .R.....T...
307634464160 40 00 80 52 21 00 80 52 E2 03 1F 2A B1 D1 FF 97 @.R!...R...*
307634464170 1F 04 00 31 E8 83 02 91 00 79 33 88 80 2B 00 54 ...1....y3...+T
307634464180 E0 2B 00 89 98 18 00 94 48 0A 80 52 28 60 87 72 .....H...R("r
307634464190 A8 03 10 88 E8 27 40 89 C8 F9 07 37 A8 14 00 80 .....7.....
3076344641A0 08 09 4B 89 E8 16 00 35 F3 0F 00 F9 F3 03 1F 2A ..K....5.....
3076344641B0 04 00 00 14 73 06 00 11 7F 0A 00 71 A0 15 00 54 ...S.....q...T
3076344641C0 00 14 00 F0 E8 03 02 91 A1 59 86 52 00 A4 13 91 .....Y.R....
3076344641D0 6C DB FF 97 47 CD FF 97 28 AD 84 52 08 B4 B2 72 1...G...(.R...r
3076344641E0 08 7C 28 98 08 FD 0D D3 08 01 00 0B 09 7D 08 13 .|(...'.....}.
3076344641F0 29 7D 48 08 E8 03 42 39 CA 36 80 52 35 81 0A 1B )H....9...R5...
307634464200 BF 02 00 71 4D 05 00 54 FB E7 48 A9 16 7D 41 D3 ...qM..T.....}A
307634464210 1F 01 00 72 F8 03 1F 2A F4 02 99 9A 07 02 9B 9A ...F...*.....
307634464220 5F 03 1F 2A 00 0A 00 13 0A 03 1F 2A 00 0A 00 13 ...*.....

```

将断点下在判断的地方，运行可以看到不会再进入 loc\_763415A0D0,然后继续运行，可以获取到寄存器 x1 中存放的域名。

The screenshot displays the Immunity Debugger interface with several panels:

- Assembly View:** Shows assembly code for the function `sub_763415A00C+18`. A red arrow points to the instruction `TBNZ W8, #0, loc_763415A0D0`, which is highlighted in red. Below it, the instruction `ADRP X8, #dword_76343EFB08@PAGE` is highlighted in blue.
- Hex View:** Shows the corresponding hex dump of the assembly code.
- Registers:** The `X1` register is highlighted, showing the value `0000007643545000`, which is a pointer to a memory location.
- Modules:** A list of loaded modules is shown, including `system/framework/arm64/boot.oat`, `system/framework/arm64/boot-core-libart.oat`, and `system/framework/arm64/boot-conscrypt.oat`.
- Threads:** A list of threads is shown, including `5302`, `5301`, `5300`, `5299`, and `5298`.