看雪3万课程笔记-Frida 辅助分析ollvm字符串加密（一）

原创

kfyzjd2008 于 2022-03-04 10:48:54 发布 47 收藏
分类专栏： 安卓 文章标签： 安卓逆向 frida fridahook 看雪三万 逆向

本文链接：https://blog.csdn.net/kfyzjd2008/article/details/123271206

一、ollvm的混淆原理：

视频中是这样说的，程序在编译过程中会加密字符串，运行时会解密字符串，所以在运行过程中我们可以获取到它真正的字符串，

二、操作：

IDA打开.so文件，在导出函数中搜索.data会发现三个.datadiv_decode+随机数字段。双击进入

```
byte_37031 ^= 0x57u;
byte_37032 ^= 0x57u;
byte_37033 ^= 0x57u;
byte_37034 ^= 0x57u;
byte_37035 ^= 0x57u;
byte_37036 ^= 0x57u;
byte_37037 ^= 0x57u;
byte_37038 ^= 0x57u;
byte_37039 ^= 0x57u;
byte_3703A ^= 0x57u;
byte_3703B ^= 0x57u;
byte_3703C ^= 0x57u;
byte_3703D ^= 0x57u;
byte_3703E ^= 0x57u;
byte_37041 ^= 0x15u;
byte_37042 ^= 0x15u;
byte_37043 ^= 0x15u;
byte_37044 ^= 0x15u;
byte_37040 ^= 0x15u;
byte_37060 ^= 0x99u;
byte_37061 ^= 0x99u;
byte_37062 ^= 0x99u;
byte_37063 ^= 0x99u;
byte_37064 ^= 0x99u;
byte_37065 ^= 0x99u;
byte_37066 ^= 0x99u;
byte_37067 ^= 0x99u;
byte_37068 ^= 0x99u;
v1.n128_u64[0] = 0x9999999999999999LL;
v1.n128_u64[1] = 0x9999999999999999LL;
xmmword_37050 = (__int128)veorq_s8((int8x16_t)xmmword_370
```

双击一个进入

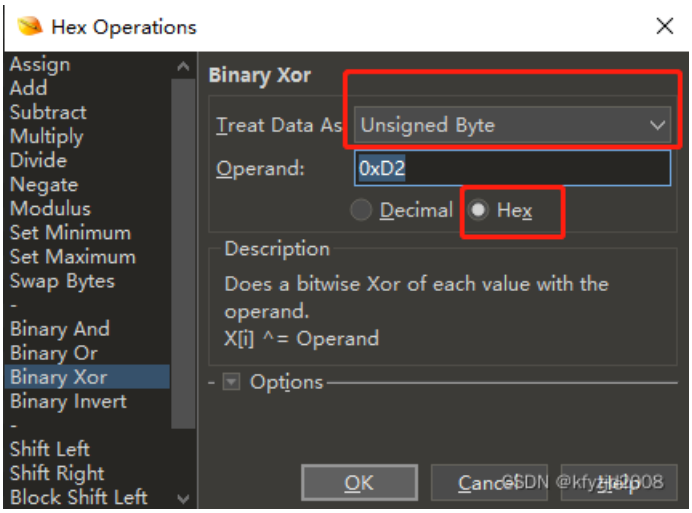
```
data:0000000000037010                   DCQ 0x3B3E273A38147777,
data:0000000000037030 byte_37030        DCB 0x16
data:0000000000037030
data:0000000000037031 byte_37031        DCB 0x15
data:0000000000037031
data:0000000000037032 byte_37032        DCB 0x1E
data:0000000000037032
data:0000000000037033 byte_37033        DCB 0x77
data:0000000000037033
data:0000000000037034 byte_37034        DCB 0x36
data:0000000000037034
data:0000000000037035 byte_37035        DCB 0x25
data:0000000000037035
data:0000000000037036 byte_37036        DCB 0x3A
data:0000000000037036
data:0000000000037037 byte_37037        DCB 0x61
data:0000000000037037
data:0000000000037038 byte_37038        DCB 0x63
data:0000000000037038
data:0000000000037039 byte_37039        DCB 0x7A
data:0000000000037039
data:000000000003703A byte_3703A        DCB 0x21
data:000000000003703A
data:000000000003703B byte_3703B        DCB 0x6F
data:000000000003703B
data:000000000003703C byte_3703C        DCB 0x36
data:000000000003703C
data:000000000003703D byte_3703D        DCB 0x6D
data:000000000003703D
data:000000000003703E byte_3703E        DCB 0x57
data:000000000003703E
data:000000000003703F                   ALIGN 0x20
data:0000000000037040 byte_37040        DCB 0x30
data:0000000000037040
data:0000000000037041 byte_37041        DCB 0x25
data:0000000000037041
data:0000000000037042 byte_37042        DCB 0x27
data:0000000000037042
data:0000000000037043 byte_37043        DCB 0x6D
```


如图，比如我们想看看byte_37031，byte_37031是什么字符。打开010 Editor

file -> new -> new hex file

打开一个新的十六进制文本，输入 15 1E

tool -> hex Operations ->



Operand:填写第一张图中的57，注意：那里是什么就写什么，不能乱写。点OK

此时就是它的真实字符。

用fridahook代码如下：

```
function print_string(addr){
    var base_hello_jni = Module.findBaseAddress("libhello-jni.so");
    if(base_hello_jni){
        var addr_str = base_hello_jni.add(addr);
        console.log("addr:",addr,ptr(addr_str).readCString());

    }
}
```

```
[Mi9 Pro 5G::com.example.hellojni_sign2]-> print_string(0x37040)
addr: 225344 %02x
[Mi9 Pro 5G::com.example.hellojni_sign2]-> print_string(0x37070)
addr: 225392 sign1
[Mi9 Pro 5G::com.example.hellojni_sign2]->
```