看雪3万课程笔记-Frida 辅助分析ollvm字符串加密（二）

原创

kfyzjd2008 已于 2022-03-10 14:36:21 修改 97 收藏
分类专栏： 安卓 文章标签： 安卓逆向 frida 逆向 frida hook 看雪三万
于 2022-03-10 13:59:17 首次发布
版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
本文链接：https://blog.csdn.net/kfyzjd2008/article/details/123398205
版权


安卓 专栏收录该内容
19 篇文章 5 订阅
订阅专栏

接上一篇文章。

一、使用到工具

hellojni_2.0.1.apk

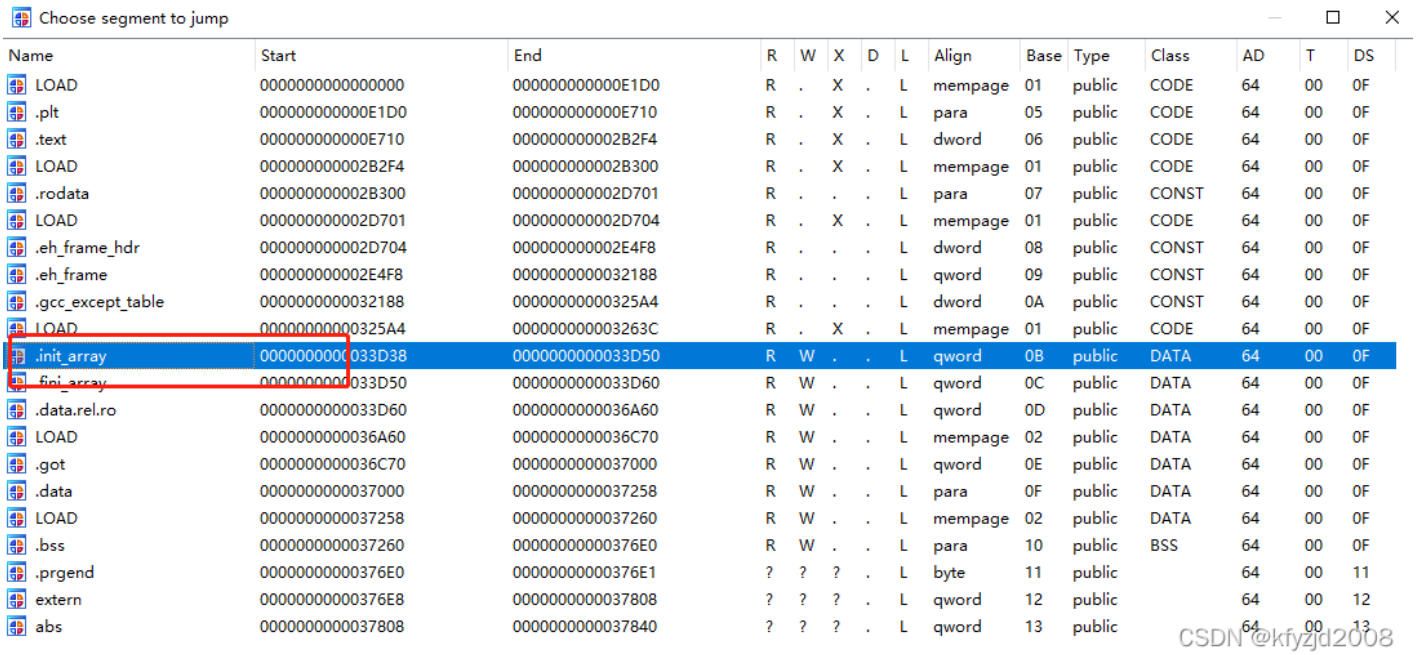
二、知识点：

ARM64中才会出现数组形加密字符串。

三、课程步骤：

将.so文件拖入IDA64，在导出函数中我们发现搜索不到.data或者decode。

此时我们回到IDA-viewA窗口中按下CTRL+S调出XX窗口找到.init_array



Name	Start	End	R	W	X	D	L	Align	Base	Type	Class	AD	T	DS
LOAD	0000000000000000	000000000000E1D0	R	.	X	.	L	mempage	01	public	CODE	64	00	0F
.plt	000000000000E1D0	000000000000E710	R	.	X	.	L	para	05	public	CODE	64	00	0F
.text	000000000000E710	000000000002B2F4	R	.	X	.	L	dword	06	public	CODE	64	00	0F
LOAD	000000000002B2F4	000000000002B300	R	.	X	.	L	mempage	01	public	CODE	64	00	0F
.rodata	000000000002B300	000000000002D701	R	.	.	.	L	para	07	public	CONST	64	00	0F
LOAD	000000000002D701	000000000002D704	R	.	X	.	L	mempage	01	public	CODE	64	00	0F
.eh_frame_hdr	000000000002D704	000000000002E4F8	R	.	.	.	L	dword	08	public	CONST	64	00	0F
.eh_frame	000000000002E4F8	0000000000032188	R	.	.	.	L	qword	09	public	CONST	64	00	0F
.gcc_except_table	0000000000032188	00000000000325A4	R	.	.	.	L	dword	0A	public	CONST	64	00	0F
LOAD	00000000000325A4	000000000003263C	R	.	X	.	L	mempage	01	public	CODE	64	00	0F
.init_array	0000000000033D38	0000000000033D50	R	W	.	.	L	qword	0B	public	DATA	64	00	0F
.fini_array	0000000000033D50	0000000000033D60	R	W	.	.	L	qword	0C	public	DATA	64	00	0F
.data.rel.ro	0000000000033D60	0000000000036A60	R	W	.	.	L	qword	0D	public	DATA	64	00	0F
LOAD	0000000000036A60	0000000000036C70	R	W	.	.	L	mempage	02	public	DATA	64	00	0F
.got	0000000000036C70	0000000000037000	R	W	.	.	L	qword	0E	public	DATA	64	00	0F
.data	0000000000037000	0000000000037258	R	W	.	.	L	para	0F	public	DATA	64	00	0F
LOAD	0000000000037258	0000000000037260	R	W	.	.	L	mempage	02	public	DATA	64	00	0F
.bss	0000000000037260	00000000000376E0	R	W	.	.	L	para	10	public	BSS	64	00	0F
.prgend	00000000000376E0	00000000000376E1	?	?	?	.	L	byte	11	public		64	00	11
extern	00000000000376E8	0000000000037808	?	?	?	.	L	qword	12	public		64	00	12
abs	0000000000037808	0000000000037840	?	?	?	.	L	qword	13	public		64	00	13

CSDN @kfyzjd2008

```
.init_array:0000000000033D38
.init_array:0000000000033D38 ; Segment type: Pure data
.init_array:0000000000033D38                 AREA .init_array, DATA, ALIGN=3
.init_array:0000000000033D38                 ; ORG 0x33D38
.init_array:0000000000033D38 off_33D38       DCQ std__string___4921590060622252445
.init_array:0000000000033D38                                         ; DATA XREF: LOAD:0000000000000088↑o
.init_array:0000000000033D38                                         ; LOAD:00000000000001D8↑o
.init_array:0000000000033D40                 DCQ std__string___16272379671879213405
.init_array:0000000000033D48                 DCQ std__string___19739608168597033O5
.init_array:0000000000033D48 ; .init_array   ends
.init_array:0000000000033D48
.fini_array:0000000000033D50 ; ELF Termination Function Table
.fini_array:0000000000033D50 ; ===========================================================================
.fini_array:0000000000033D50
.fini_array:0000000000033D50 ; Segment type: Pure data
.fini_array:0000000000033D50                 AREA .fini_array, DATA, ALIGN=3
.fini_array:0000000000033D50                 ; ORG 0x33D50
.fini_array:0000000000033D50                 DCQ j_nullsub_2
```

CSDN @kfyzjd2008

进入.initarray中可以看到三个函数，我们进入第一个，然后F5

```
int8x16_t std::string_::4921590060622252445()
{
  int8x16_t v0; // q0
  int8x16_t v1; // q1
  int8x16_t v2; // q2
  int8x16_t v3; // q4
  int8x16_t result; // q0

  byte_37030 ^= 0xC6u;
  byte_37033 ^= 0xC6u;
  byte_37034 ^= 0xC6u;
  byte_37035 ^= 0xC6u;
  byte_37036 ^= 0xC6u;
  byte_37037 ^= 0xC6u;
  v0.n128_u64[0] = 0xC6C6C6C6C6C6C6C6LL;
  v0.n128_u64[1] = 0xC6C6C6C6C6C6C6C6LL;
  stru_37010[0] = veorq_s8(stru_37010[0], v0);
  stru_37010[1] = veorq_s8(stru_37010[1], v0);
  byte_37031 ^= 0xC6u;
  byte_37032 ^= 0xC6u;
  byte_37038 ^= 0xC6u;
  byte_37039 ^= 0xC6u;
  byte_3703A ^= 0xC6u;
  byte_3703B ^= 0xC6u;
  byte_3703C ^= 0xC6u;
  byte_3703D ^= 0xC6u;
  byte_3703E ^= 0xC6u;
  byte_37040 ^= 0x94u;
  byte_37041 ^= 0x94u;
  byte_37042 ^= 0x94u;
  byte_37043 ^= 0x94u;
  byte_37044 ^= 0x94u;
  v1.n128_u64[0] = 0x3131313131313131LL;
  v1.n128_u64[1] = 0x3131313131313131LL;
  byte_37065 ^= 0x31u;
  xmmword_37050 = (__int128)veorq_s8((int8x16_t)xmmword_37050, v1);
  byte_37060 ^= 0x31u;
  byte_37061 ^= 0x31u;
  byte_37062 ^= 0x31u;
  byte_37063 ^= 0x31u;
  byte_37064 ^= 0x31u;
  byte_37066 ^= 0x31u;
  byte_37067 ^= 0x31u;
```

我们要讲得

同上篇文章

CSDN @kfyzjd2008

```
stru_37010[0] = veorq_s8(stru_37010[0], v0);
```

这种的只有在arm64中才会出现，如果是arm32只会出现我们上节课讲的那种。

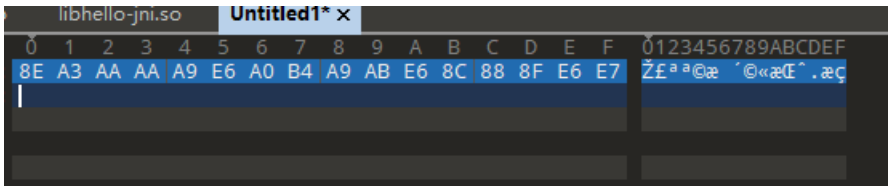
此处和v0进行异或运算，我们要看v0的值是多少。

```
.text:000000000000ECF4                STRB            W17, [X8,#(byte_37033 - 0x37010)]
.text:000000000000ECF8                STRB            W0, [X8,#(byte_37034 - 0x37010)]
.text:000000000000ECFC                LDRB            W17, [X14]
.text:000000000000ED00                STRB            W1, [X8,#(byte_37035 - 0x37010)]
.text:000000000000ED04                LDRB            W0, [X14,#(byte_37041 - 0x37040)]
.text:000000000000ED08                STRB            W2, [X8,#(byte_37036 - 0x37010)]
.text:000000000000ED0C                LDRB            W1, [X14,#(byte_37042 - 0x37040)]
.text:000000000000ED10                STRB            W3, [X8,#(byte_37037 - 0x37010)]
.text:000000000000ED14                LDRB            W2, [X14,#(byte_37043 - 0x37040)]
.text:000000000000ED18                LDRB            W3, [X14,#(byte_37044 - 0x37040)]
.text:000000000000ED1C                MOVI            V0.16B, #0xC6
.text:000000000000ED20                ADRL            X9, xmmword_37050
.text:000000000000ED28                EOR             V5.16B, V5.16B, V0.16B
.text:000000000000ED2C                EOR             V6.16B, V6.16B, V0.16B
.text:000000000000ED30                EOR             W15, W15, W13
.text:000000000000ED34                EOR             W16, W16, W13
.text:000000000000ED38                EOR             W4, W4, W13
.text:000000000000ED3C                EOR             W5, W5, W13
.text:000000000000ED40                EOR             W6, W6, W13
```

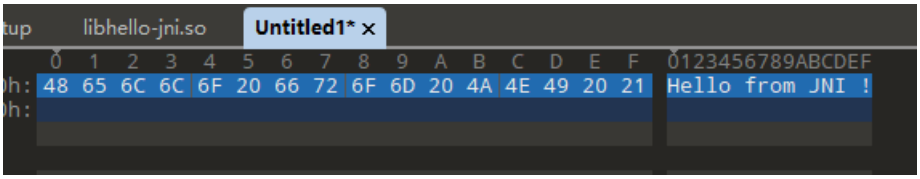
CSDN @kfyzjd2008

此处我们可以看到v0的值为0xC6

打开我们的010 Edtor，进入 stru_37010[0] 将值复制到 010中

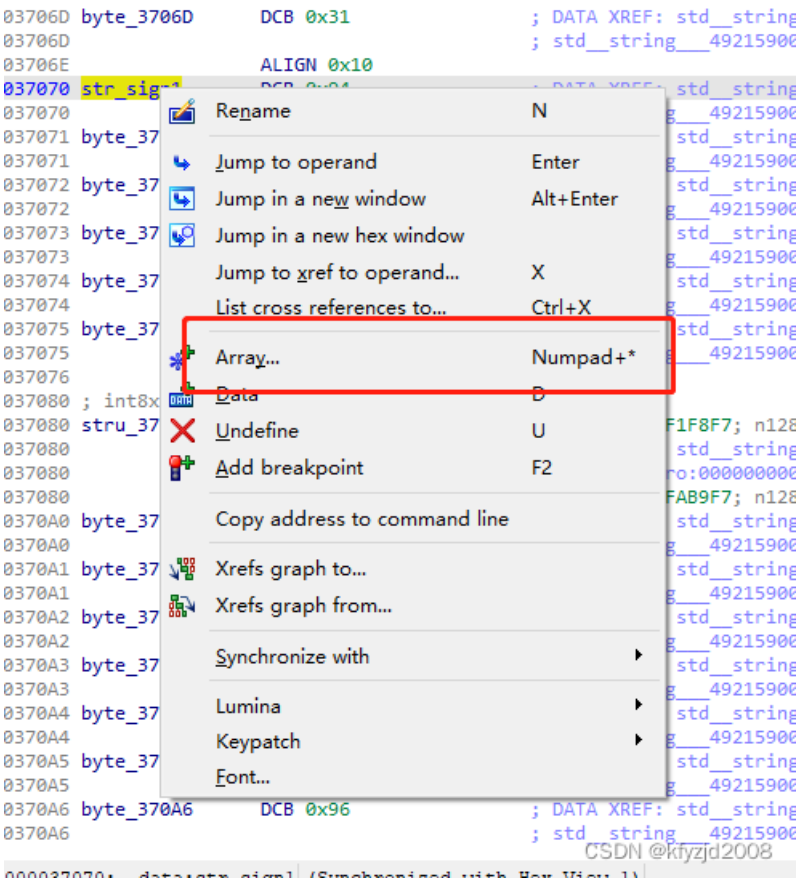


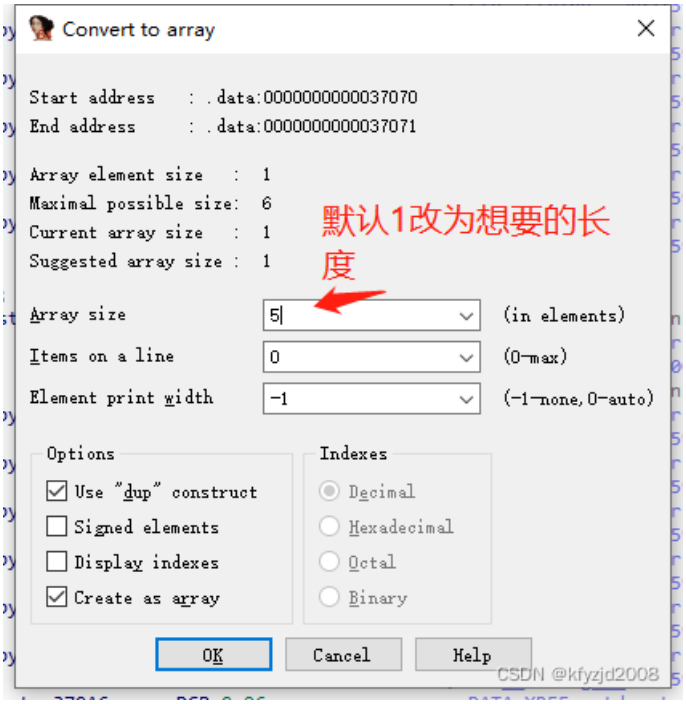
根据上节课的内容进行异或操作后结果

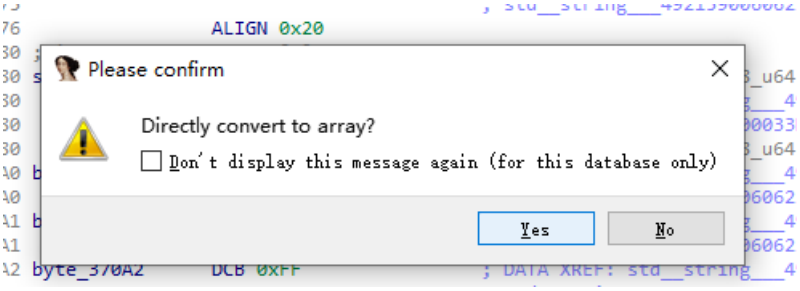


2、对于正常的字节改为数组形式。

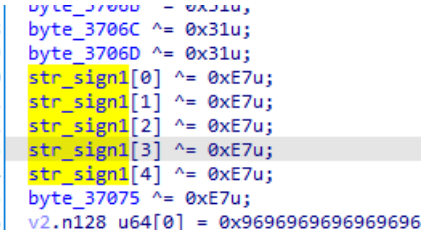
按下ESC回到它的调用处右键。


03706D byte_3706D DCB 0x31 ; DATA XREF: std__string
03706D ; std__string___49215900
03706E ALIGN 0x10
037070 str_sig1
Rename N
Jump to operand Enter
Jump in a new window Alt+Enter
Jump in a new hex window
Jump to xref to operand... X
List cross references to... Ctrl+X
Array... Numpad+*
Data D
Undefine U
Add breakpoint F2
Copy address to command line
Xrefs graph to...
Xrefs graph from...
Synchronize with
Lumina
Keypatch
Font...
0370A6 byte_370A6 DCB 0x96 ; DATA XREF: std__string
0370A6 ; std__string___49215900
CSDN @kfyzjd2008


Convert to array
Start address : .data:0000000000037070
End address : .data:0000000000037071
Array element size : 1
Maximal possible size: 6
Current array size : 1
Suggested array size : 1
默认1改为想要的长度
Array size 5 (in elements)
Items on a line 0 (0=max)
Element print width -1 (-1=none,0=auto)
Options
Use "dup" construct
Signed elements
Display indexes
Create as array
Indexes
Decimal
Hexadecimal
Octal
Binary
OK
Cancel
Help
CSDN @kfyzjd2008


ALIGN 0x20
Please confirm
Directly convert to array?
Don't display this message again (for this database only)
Yes
No
byte_370A2 DCB 0xFF ; DATA XREF: std__string

这里点击YES。



再看我们刚才的伪代码位置，就变成了数组形式。

这节课就到这里了。