

## 实验3-4

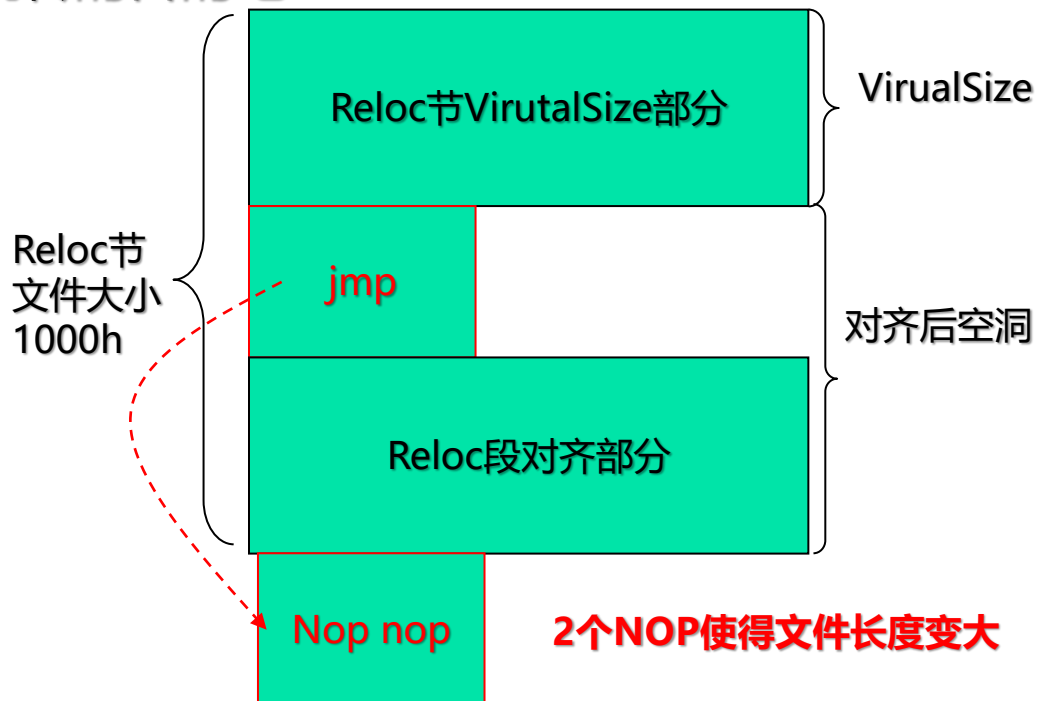
---

### 尾区段寄生之扩展区段大小

# 实验：寄生代码在最后节 文件长度变大

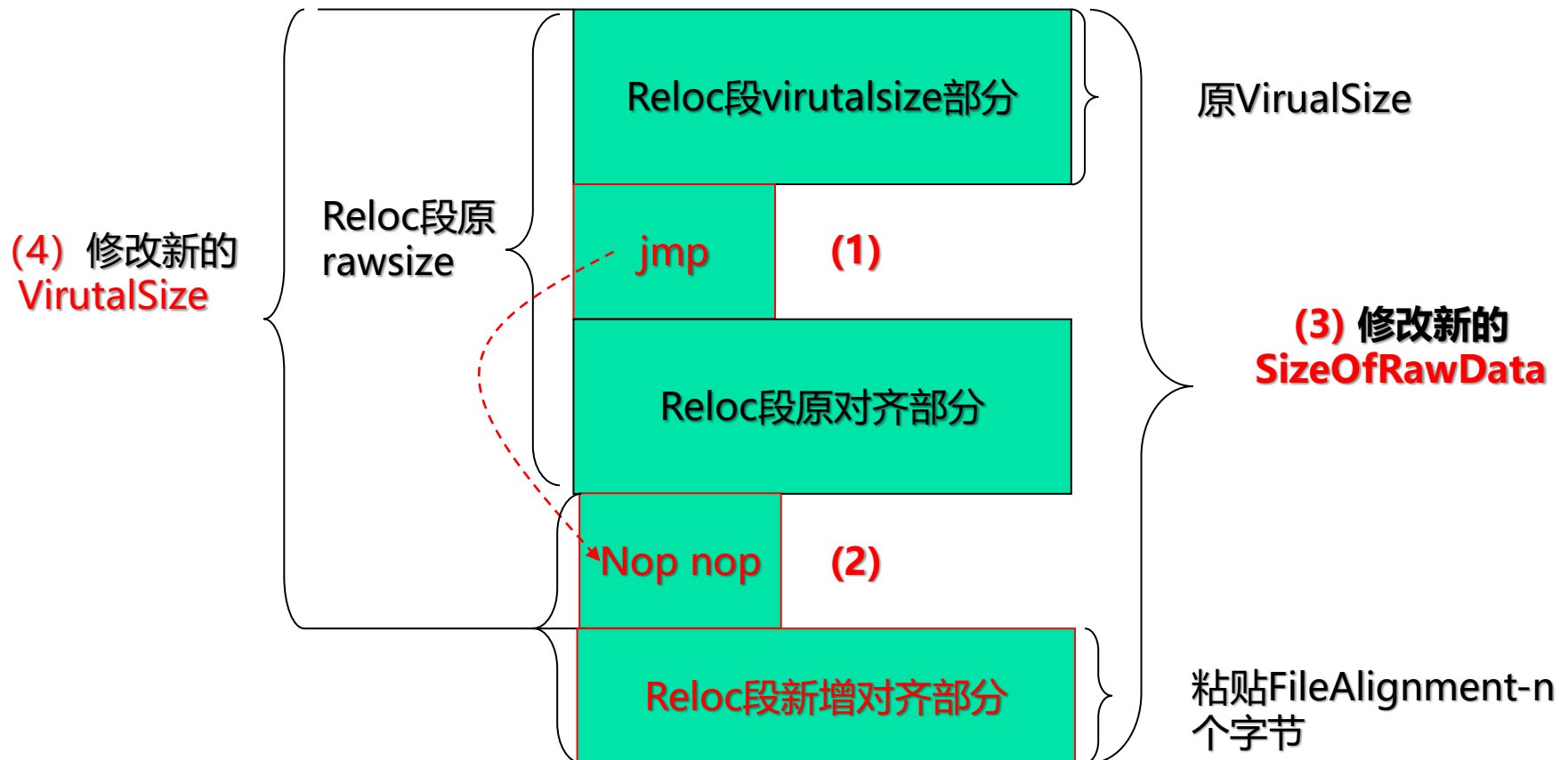
- 实验目的：该实验尝试，如果**寄生代码**大于**末节对齐后的剩余空间**，如何增加节长。
- 如何模拟这一过程？不需要真正写一个大于空洞的寄生代码，只需要修改前面的JMP指令，让其跳到节的后面（SizeOfRawData），而在节后加入若干条NOP指令，作为跳转的目的地

pFile	Data	Description
00000270	2E 72 65 6C	Name
00000274	6F 63 00 00	
00000278	00000E2A	Virtual Size
0000027C	0002C000	RVA
00000280	00001000	Size of Raw Data
00000284	0002A000	Pointer to Raw Data
00000288	00000000	Pointer to Relocations
0000028C	00000000	Pointer to Line Number
00000290	0000	Number of Relocation
00000292	0000	Number of Line Number
00000294	42000040	Characteristics
	00000040	
	02000000	



## 实验的步骤

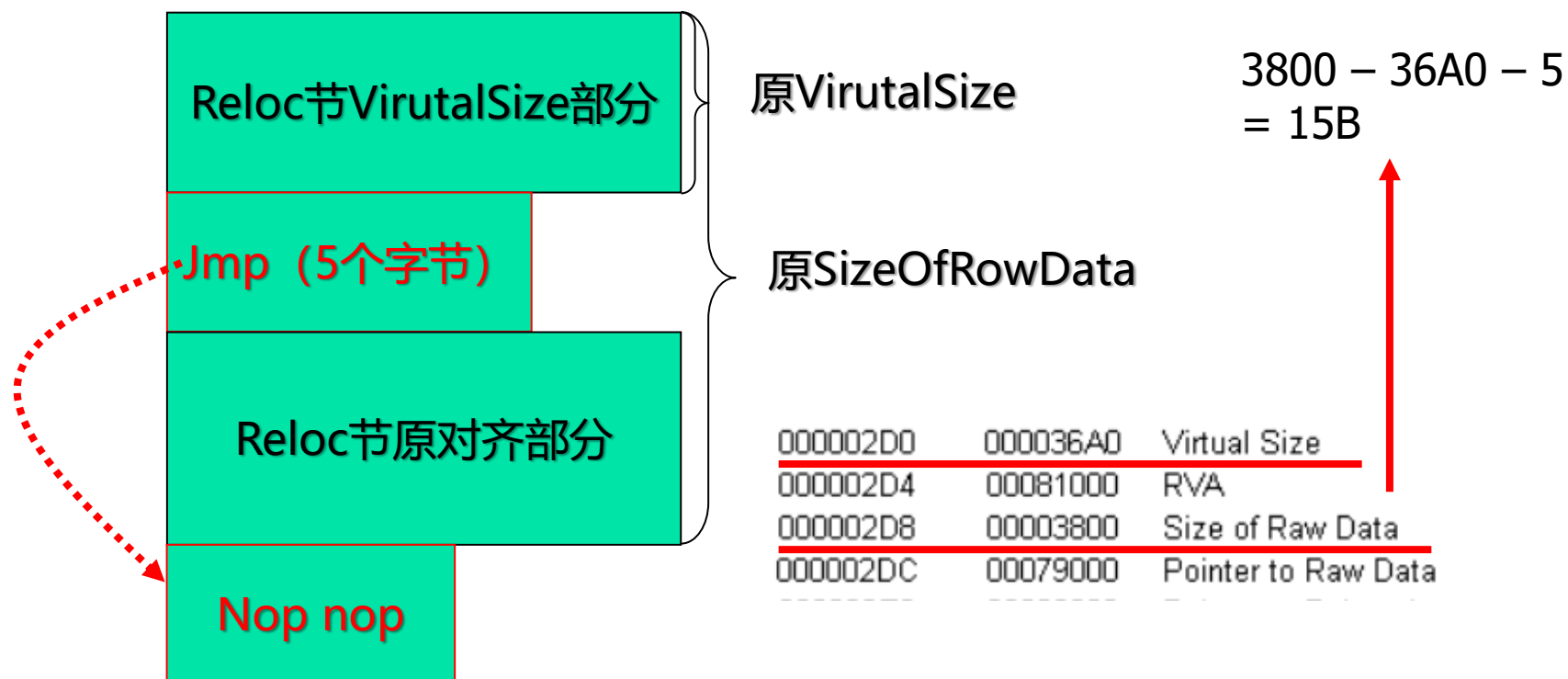
- 1) 在reloc节原VirtualSize后添加JMP xx xx xx xx, 跳到reloc节后面
- 2) 在reloc节后添加**学号最后一位个数n**的NOP指令 (这里假设n=2)
- 3) 修改reloc节头的**SizeOfRawData**, 加一个FileAlignment
- 4) 修改reloc节头的**VirtualSize**为原SizeOfRawData+n(n个nop), 现在NOP才是实际结尾
- 5) 修改可选映像头的**SizeOfImage** = (relocRVA+新VirtualSize)除以SectionAlign取上整
- 6) 在NOP后手动增加FileAlignment-n个字节, 内容不论, 为对齐后填充内容
- 7) 修改入口点RVA (AddressOfEntryPoint) 为寄生代码开始处



# 实验步骤1：计算JMP指令的位移量

JMP指令的跳转位移量

= 原SizeOfRawData - ( 原Virutalsize + JMP指令长 (5字节) )



# 实验步骤1：在文件中写入JMP指令

## ■ reloc节写入JMP跳到节后

Reloc节的文件起始PointerToRawData

文件中找到写入位置

原VirtualSize大小

Reloc节VirtualSize部分

JMP指令的文件位置

jmp

000002D0	000036A0	Virtual Size
000002D4	00081000	RVA
000002D8	00003800	Size of Raw Data
000002DC	00079000	Pointer to Raw Data

$$79000 + 36A0 = 7C6A0$$

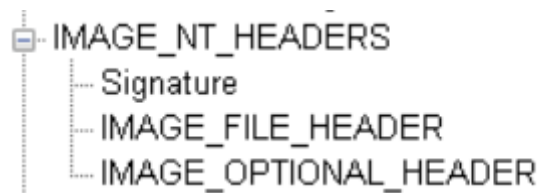
写入JMP

E9 00 00 01 5B

cloudmusic.exe\*

0007c690h:	20	3B	44	3B	48	3B	4C	3B
0007c6a0h:	e9	5B	01	00	00	00	00	00

# 实验步骤2: reloc节后添加2个NOP



000002D0	000036A0	Virtual Size
000002D4	00081000	RVA
000002D8	00003800	Size of Raw Data
000002DC	00079000	Pointer to Raw Data

文件对齐后的大小为3800，文件起始位置79000，2个NOP的写入位置为：  
 $79000 + 3800 = 7C800$

Reloc节起始文件位置  
79000

Reloc节原对齐大小  
3800

NOP写入位置  
 $79000 + 3800$



cloudmusic.exe*	
0007c800h:	90 90 00 00 00
0007c810h:	48 86 F7 0D 01
0007c820h:	01 01 31 0B 30
0007c830h:	4C 06 0A 2B 06

写入位置

写入内容

写入2个NOP

# 实验步骤3：修改节的SizeOfRawData

0000013C	00400000	Image Base
00000140	00001000	Section Alignment
00000144	00000200	File Alignment

← 查看文件对齐粒度为200

查看文件对齐粒度为200，因此

新SizeOfRawData = 原SizeOfRawData + 200 = 3800 + 200 = 3A00

pFile	Data	Description
000002C8	2E 72 65 6C	Name
000002CC	6F 63 00 00	
000002D0	000036A0	Virtual Size
000002D4	00081000	RVA
000002D8	00003800	Size of Raw Data
000002DC	00079000	Pointer to Raw Data



cloudmusic.exe*															
000002d0h:	A0	36	00	00	00	10	08	00	00	3A	00	00	00	00	00
000002e0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000002f0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

文件偏移2D8处写入00 3A 00 00

# 实验步骤4：修改节的VirtualSize

新的VirtualSize:

原SizeOfRawData + 2个NOP指令  
= 3800 + 2 = 3802

pFile	Data	Description
000002C8	2E 72 65 6C	Name
000002CC	6F 63 00 00	
000002D0	000036A0	Virtual Size
000002D4	00081000	RVA
000002D8	00003800	Size of Raw Data
000002DC	00079000	Pointer to Raw Data




cloudmusic.exe*
000002d0h: 02 38 00 00 00
000002e0h: 00 00 00 00 00
000002f0h: 00 00 00 00 00
00000300h: 00 00 00 00 00

文件偏移2D0处写入02 38 00 00



# 实验步骤5：修改文件的SizeOfImage

00000140	00001000	Section Alignment	 节对齐粒度 1000
00000144	00000200	File Alignment	
00000148	0005	Major O/S Version	
0000014A	0001	Minor O/S Version	
0000014C	0000	Major Image Version	
0000014E	0000	Minor Image Version	
00000150	0005	Major Subsystem Version	
00000152	0001	Minor Subsystem Version	
00000154	00000000	Win32 Version Value	
00000158	00085000	Size of Image	原SizeOfImage 85000

$$\begin{aligned}\text{新SizeOfImage} &= \text{ceil} [ ( \text{新VirtualSize} + \text{reloc节起始RVA} ) / \text{sectionAlignment} ] \\ &= \text{ceil} [ ( 81000 + 3802 ) / 1000 ] = 85000\end{aligned}$$

在本例中，SizeOfImage没有发生改变，于是不修改

# 实验步骤6：补上文件对齐部分

- 因为在reloc节后增加了1个文件对齐粒度，因此，这部分需要在文件中进行填充，内容不限

pFile	Data	Description
000002C8	2E 72 65 6C	Name
000002CC	6F 63 00 00	
000002D0	000036A0	Virtual Size
000002D4	00081000	RVA
000002D8	00003800	Size of Raw Data
000002DC	00079000	Pointer to Raw Data

插入的位置为2个nop指令后，  
 $79000 + 3800 + 2$

需要填充到  $79000 + 3A00 = 7CA00$

```
0007c700h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0007c710h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0007c720h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0007c730h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0007c740h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0007c750h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0007c760h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0007c770h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0007c780h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0007c790h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

剪切(T)  
复制(C)  
粘贴(P)  
剪切追加(U)  
复制追加(Y)  
复制文件路径/名称(F)  
剪贴板(I)

# 实验步骤7：修改程序入口RVA

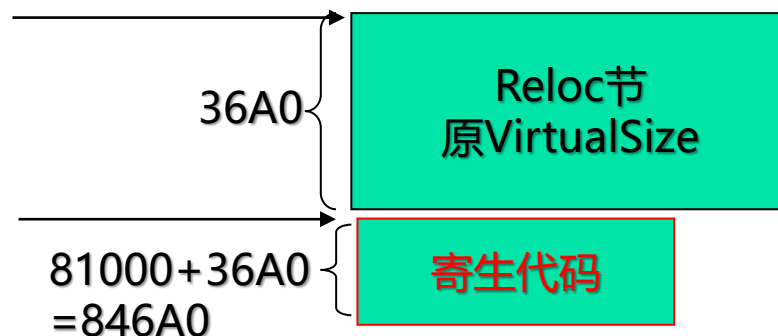
最后修改  
AddressOfEntryPoint

pFile	Data	Description
00000120	010B	Magic
00000122	0C	Major Linker Version
00000123	00	Minor Linker Version
00000124	00047400	Size of Code
00000128	00038600	Size of Initialized Data
0000012C	00000000	Size of Uninitialized Data
00000130	0002D9EE	Address of Entry Point

修改为JMP指令RVA

000002D0	000036A0	Virtual Size
000002D4	00081000	RVA
000002D8	00003800	Size of Raw Data
000002DC	00079000	Pointer to Raw Data

节起始RVA=81000



cloudmusic.exe\*

```
00000130h: A0 46 08 00 00
00000140h: 00 10 00 00 00
00000150h: 05 00 01 00 00
```

← 文件偏移130处写入 A0 46 08 00

# 在x64dbg中加载看结果

CPU - 主线程, 模块 - cloudmus		
地址	HEX 数据	反汇编
008C46A0	E9 5B010000	JMP cloudmus.008C4800
008C46A5	0000	ADD BYTE PTR DS:[EAX], AL
008C46A7	0000	ADD BYTE PTR DS:[EAX], AL
008C46A9	0000	ADD BYTE PTR DS:[EAX], AL
008C46AB	0000	ADD BYTE PTR DS:[EAX], AL
008C46AD	0000	ADD BYTE PTR DS:[EAX], AL
008C46AF	0000	ADD BYTE PTR DS:[EAX], AL
008C46B1	0000	ADD BYTE PTR DS:[EAX], AL
008C46B3	0000	ADD BYTE PTR DS:[EAX], AL

地址	HEX 数据	反汇编
008C47F4	0000	ADD BYTE PTR DS:[EAX], AL
008C47F6	0000	ADD BYTE PTR DS:[EAX], AL
008C47F8	0000	ADD BYTE PTR DS:[EAX], AL
008C47FA	0000	ADD BYTE PTR DS:[EAX], AL
008C47FC	0000	ADD BYTE PTR DS:[EAX], AL
008C47FE	0000	ADD BYTE PTR DS:[EAX], AL
008C4800	90	NOP
008C4801	90	NOP

看见模拟的超过空洞长度的病毒体（从JMP到n个NOP）都已经成功地加载到内存中！

**重复以上实验，并自行完成以下内容：**

**在n个NOP后再添加一条JMP指令跳回原入口点**