

看雪CTF.TSRC 2018 团队赛 第二题 半加器 writeup

原创

xuenixiang 于 2018-12-05 22:04:05 发布 319 收藏

分类专栏: [CTF](#) 文章标签: [半加器](#) [writeup](#) [看雪CTF](#)

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/qq_41917908/article/details/84844384

版权

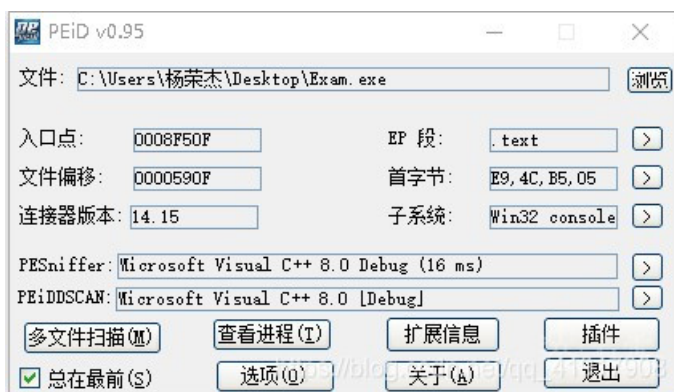


[CTF 专栏收录该内容](#)

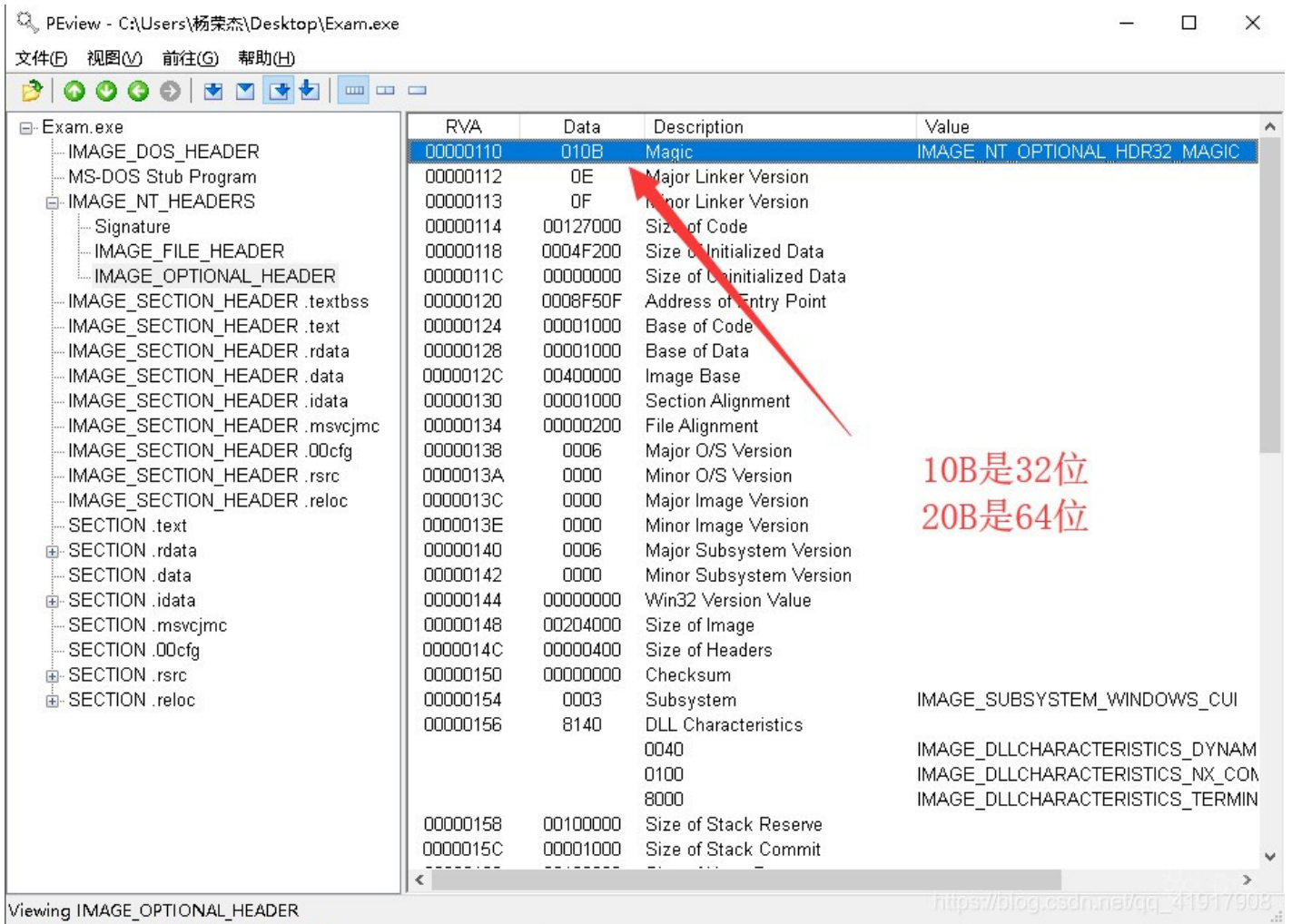
2 篇文章 0 订阅

订阅专栏

首先查壳, 发现没有壳



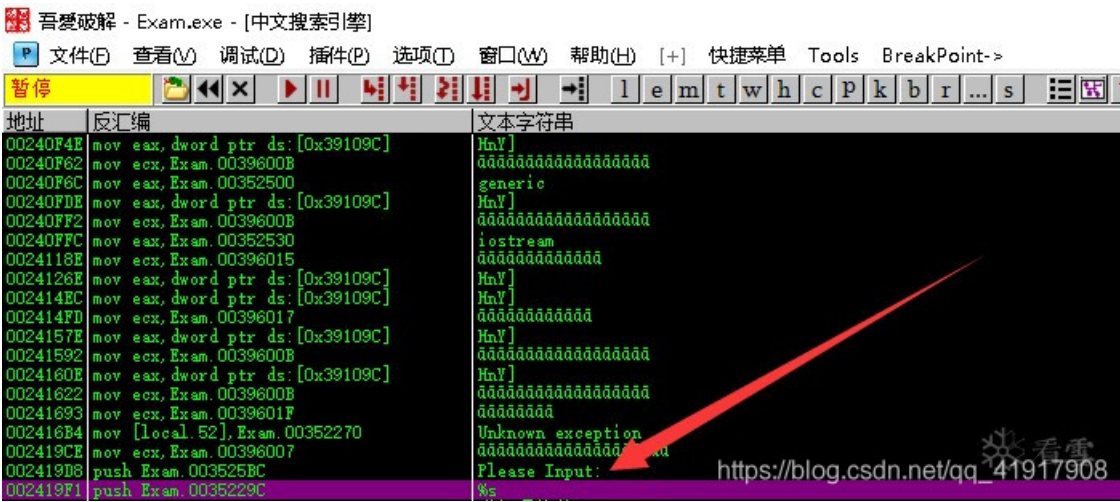
然后根据PE看一下是不是32位程序, 这个是32位程序, 所以可以用OD来分析 (希望OD能早日开发出支持64位的版本)



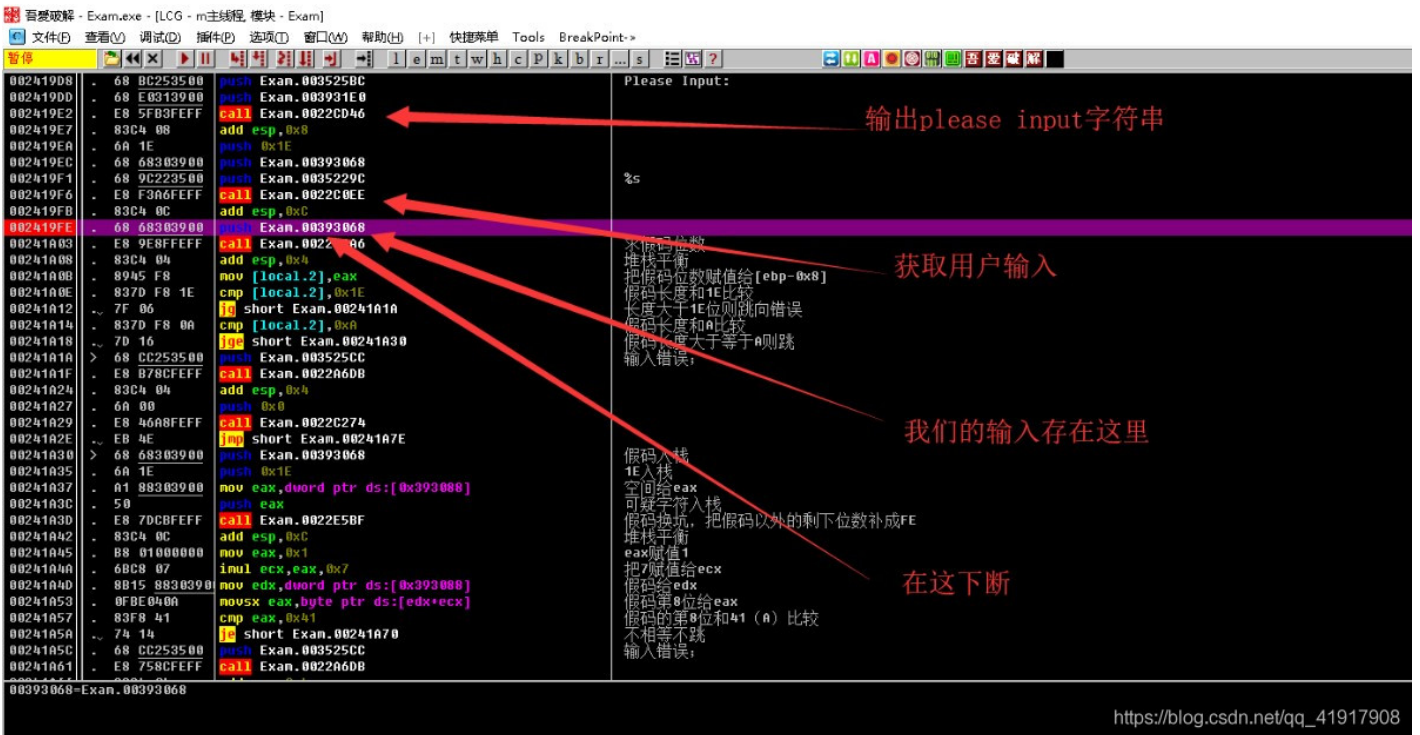
根据程序特征我们可以根据 Please Input 这个字符串定位到关键点



直接载入OD，查找对应字符串



002419FE |. 68 68303900 push Exam.00393068 在此下断



通过回溯可以看到我们目前所在的call（弄清楚自己现在所处位置）

吾爱破解 - Exam.exe - [LCG - 主线程, 模块 - Exam]

文件(F) 查看(V) 调试(D) 插件(P) 选项(O) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint->

暂停

0028A85C	-	833A 00	cmp dword ptr ds:[edx],0x0	
0028A85F	-	74 21	je short Exam.0028A882	
0028A861	-	8B45 DC	mov eax,dword ptr ss:[ebp-0x24]	Exam.00394D4C
0028A864	-	50	push eax	Exam.00394D4C
0028A865	-	E8 2725FAFF	call Exam.0022CD91	
0028A86A	-	83C4 04	add esp,0x4	
0028A86D	-	0FB6C8	movzx ecx,al	
0028A870	-	85C9	test ecx,ecx	
0028A872	-	74 0E	je short Exam.0028A882	
0028A874	-	8B55 DC	mov edx,dword ptr ss:[ebp-0x24]	Exam.00394D4C
0028A877	-	8B02	mov eax,dword ptr ds:[edx]	
0028A879	-	50	push eax	Exam.00394D4C
0028A87A	-	E8 3507FAFF	call Exam.0022AFB4	
0028A87F	-	83C4 04	add esp,0x4	
0028A882	>	E8 49010000	call Exam.0028A9D0	验证生成算法call
0028A887	-	8945 D4	mov dword ptr ss:[ebp-0x2C],eax	Exam.00394D4C
0028A88A	-	E8 6A26FAFF	call Exam.0022CE99	检查PE是否正常
0028A88F	-	0FB6C8	movzx ecx,al	
0028A892	-	85C9	test ecx,ecx	
0028A894	-	75 09	jnz short Exam.0028A89F	
0028A896	-	8B55 D4	mov edx,dword ptr ss:[ebp-0x2C]	
0028A899	-	52	push edx	Exam.00394D4C
0028A89A	-	E8 8A37FAFF	call Exam.0022E029	验证比较真码call
0028A89F	>	0FB645 E7	movzx eax,byte ptr ss:[ebp-0x19]	
0028A8A3	-	85C0	test eax,eax	Exam.00394D4C
0028A8A5	-	75 05	jnz short Exam.0028A8AC	
0028A8A7	-	E8 4BFBF9FF	call Exam.0022A3F7	退出call
0028A8AC	>	6A 00	push 0x0	
0028A8AE	-	6A 01	push 0x1	
0028A8B0	-	E8 2536FAFF	call Exam.0022DEDA	

https://blog.csdn.net/qq_41917908

运行程序，输入假码xuenixiang，会断在下图位置

吾爱破解 - Exam.exe - [LCG - 主线程, 模块 - Exam]

文件(F) 查看(V) 调试(D) 插件(P) 选项(O) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint->

暂停

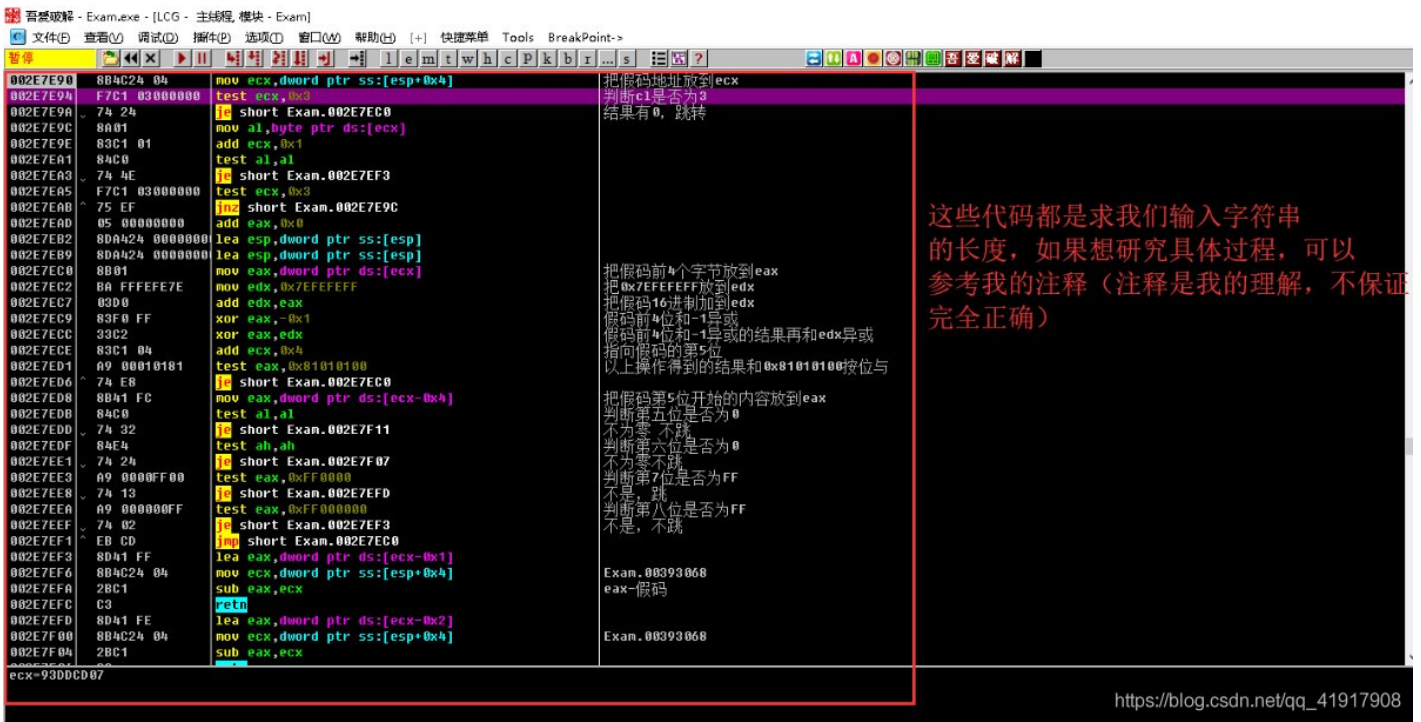
00241A03	-	E8 9E8FEFF	call Exam.0022A9A6	ASCII "xuenixiang"
00241A08	-	83C4 04	add esp,0x4	求假码位数
00241A0B	-	8945 F8	mov [local.2],eax	维持平衡
00241A0E	-	837D F8 1E	cmp [local.2],0x1E	把假码位数赋值给[ebp-0x8]
00241A12	-	7F 06	jb short Exam.00241A1A	假码长度和1E比较
00241A14	-	837D F8 0A	cmp [local.2],0xA	长度大于1E位则跳向错误
00241A18	-	7D 16	jge short Exam.00241A30	假码长度和0xA比较
00241A1A	>	68 CC253500	push Exam.003525CC	假码长度大于等于0A则跳
00241A1F	-	E8 B78CFE9F	call Exam.0022A6D8	输入错误;
00241A24	-	83C4 04	add esp,0x4	
00241A27	-	6A 00	push 0x0	
00241A29	-	E8 46A8FE9F	call Exam.0022C274	这个call是求我们输入字符串的位数的
00241A2E	-	E8 4E	jmp short Exam.00241A7E	
00241A30	>	68 68303900	push Exam.00393068	假码入栈
00241A35	-	6A 1E	push 0x1E	1E入栈
00241A37	-	A1 88303900	mov eax,dword ptr ds:[0x393088]	空间给eax
00241A3C	-	50	push eax	可穿字符串
00241A3D	-	E8 7DCBFE9F	call Exam.0022E58F	假码换坑，把假码以外的剩下位数补成PE
00241A42	-	83C4 0C	add esp,0xC	堆栈平衡
00241A45	-	B8 01000000	mov eax,0x1	eax赋值1
00241A4A	-	68C8 07	imul ecx,eax,0x7	把7赋值给ecx
00241A4D	-	8B15 88303900	mov edx,dword ptr ds:[0x393088]	假码给edx
00241A53	-	0FB60400	movsx eax,byte ptr ds:[edx+ecx]	假码第8位给eax
00241A57	-	83F8 41	cmp eax,0x41	假码的第8位和41(A)比较
00241A5A	-	74 14	je short Exam.00241A70	不相等不跳
00241A5C	-	68 CC253500	push Exam.003525CC	输入错误;
00241A61	-	E8 758CFE9F	call Exam.0022A6D8	
00241A66	-	83C4 04	add esp,0x4	
00241A69	-	6A 00	push 0x0	
00241A6B	-	E8 04A8FE9F	call Exam.0022C274	假码地址给eax
00241A70	>	A1 88303900	mov eax,dword ptr ds:[0x393088]	假码入栈
00241A75	-	50	push eax	假码的每一位和1E异或
00241A76	-	E8 29B9FE9F	call Exam.0022D3A4	
00241A7B	-	83C4 04	add esp,0x4	
00241A7E	>	33C0	xor eax,eax	
00241A80	-	5F	pop edi	

Exam.<ModuleEntryPoint>

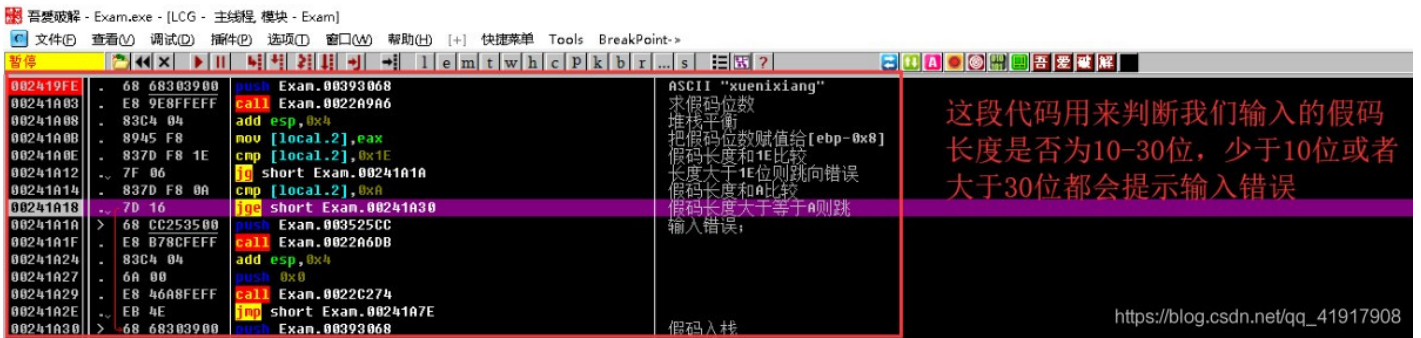
00393068=Exam.00393068 (ASCII "xuenixiang")			
地址	HEX 数据	ASCII	
001A1000	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0077F804
001A1010	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0022F50F
001A1020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		Exam.<ModuleEntryPoint>
001A1030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		Exam.<ModuleEntryPoint>
001A1040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		004F1000
001A1050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0077F800
001A1060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		CCCCCCCC
001A1070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0077F804
001A1080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		CCCCCCCC
001A1090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0077F808
001A10A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		CCCCCCCC
001A10B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0077F80C
001A10C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		CCCCCCCC
001A10D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0077F810
001A10E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		CCCCCCCC
001A10F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0077F814
001A1100	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		CCCCCCCC
001A1110	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0077F818
001A1120	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		CCCCCCCC
001A1130	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0077F81C
001A1140	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		CCCCCCCC
001A1150	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0077F820
001A1160	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		CCCCCCCC
001A1170	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0077F824
001A1180	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		CCCCCCCC
001A1190	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0077F828
001A11A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		CCCCCCCC
001A11B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0077F82C
001A11C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		CCCCCCCC
001A11D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0077F830
001A11E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		CCCCCCCC
001A11F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0077F834
001A1200	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		CCCCCCCC

https://blog.csdn.net/qq_41917908

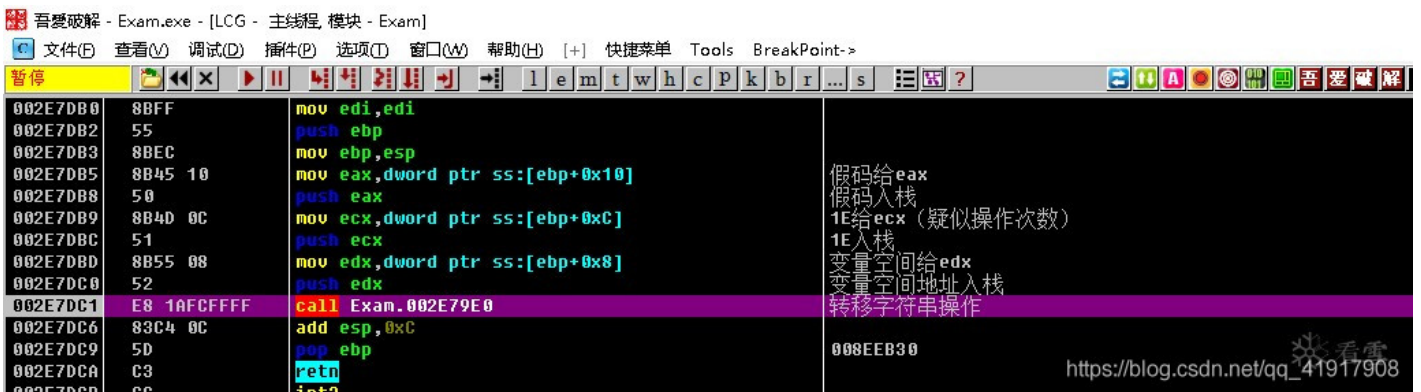
我们进00241A03这个call来看看



判断我们输入的字符串长度是否在10-30位之间



我们进入这个call 0022E5BF 看看



继续进2E7DC1这个call

吾爱破解 - Exam.exe - [LCG - 主线程, 模块 - Exam]

文件(F) 查看(V) 调试(D) 插件(P) 选项(O) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint->

002E79E8	837D 08 00	cmp dword ptr ss:[ebp+0x8],0x0	
002E79EC	74 0F	jbe short Exam.002E79FD	1E和0比较 不大于则跳
002E79EE	837D 0C 00	cmp dword ptr ss:[ebp+0xC],0x0	
002E79F2	76 09	jbe short Exam.002E79FD	
002E79F4	C745 F0 010000	mov dword ptr ss:[ebp-0x10],0x1	
002E79FB	EB 07	jmp short Exam.002E7A04	
002E79FD	C745 F0 000000	mov dword ptr ss:[ebp-0x10],0x0	
002E7A04	8B45 F0	mov eax,dword ptr ss:[ebp-0x10]	Exam.002E7DC6 Exam.00393068
002E7A07	8945 EC	mov dword ptr ss:[ebp-0x14],eax	
002E7A0A	837D EC 00	cmp dword ptr ss:[ebp-0x14],0x0	
002E7A0E	75 23	jnz short Exam.002E7A33	
002E7A10	68 10B03600	push Exam.0036B010	((destination) != NULL && ((size_in_elements) > 0
002E7A15	68 3CBE3500	push Exam.0035BE3C	%1s
002E7A1A	6A 00	push 0x0	
002E7A1C	6A 43	push 0x43	
002E7A1E	68 90B03600	push Exam.0036B090	minkernel\crt\ucrt\inc\corecrt_internal_string_templates.h
002E7A23	6A 02	push 0x2	
002E7A25	E8 CF36F4FF	call Exam.0022B0F9	
002E7A2A	83C4 18	add esp,0x18	
002E7A2D	83F8 01	cmp eax,0x1	
002E7A30	75 01	jnz short Exam.002E7A33	
002E7A32	CC	int3	
002E7A33	837D EC 00	cmp dword ptr ss:[ebp-0x14],0x0	
002E7A37	75 30	jnz short Exam.002E7A69	
002E7A39	E8 7946F4FF	call Exam.0022C0B7	
002E7A3E	C700 16000000	mov dword ptr ds:[eax],0x16	
002E7A44	6A 00	push 0x0	
002E7A46	6A 43	push 0x43	
002E7A48	68 90B03600	push Exam.0036B090	minkernel\crt\ucrt\inc\corecrt_internal_string_templates.h
002E7A4D	68 20BE3600	push Exam.0036BE20	common_tscopy_s
002E7A52	68 10B03600	push Exam.0036B010	((destination) != NULL && ((size_in_elements) > 0
002E7A57	E8 3C7EF4FF	call Exam.0022F898	
002E7A5C	83C4 14	add esp,0x14	
002E7A5F	B8 16000000	mov eax,0x16	
002E7A64	E9 78020000	jmp Exam.002E7CE1	
002E7A69	837D 10 00	cmp dword ptr ss:[ebp+0x10],0x0	xuenixiang和0比较,不为零则跳
002E7A6D	0F85 D2000000	jnz Exam.002E7A45	

edi=0077F99C, (ASCII "读")

https://blog.csdn.net/qq_41917908

往下单步跟, 到达这里可以看到转移假码的过程

吾爱破解 - Exam.exe - [LCG - 主线程, 模块 - Exam]

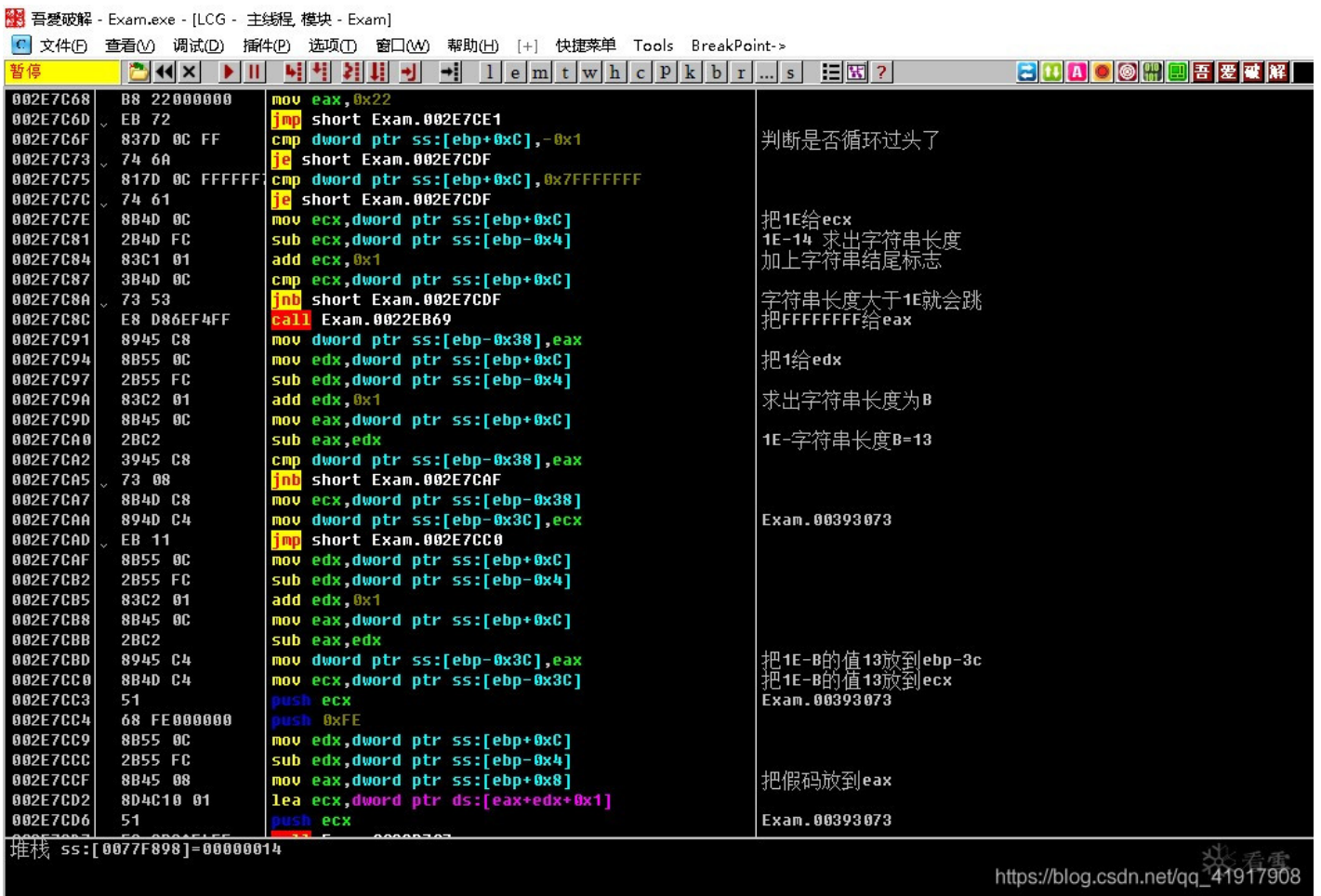
文件(F) 查看(V) 调试(D) 插件(P) 选项(O) 窗口(W) 帮助(H) [+] 快捷菜单 Tools BreakPoint->

002E7B45	8B4D 08	mov ecx,dword ptr ss:[ebp+0x8]	
002E7B48	894D F8	mov dword ptr ss:[ebp-0x8],ecx	把假码给edx
002E7B4B	8B55 10	mov edx,dword ptr ss:[ebp+0x10]	假码给ebp-c
002E7B4E	8955 F4	mov dword ptr ss:[ebp-0xC],edx	1E给eax
002E7B51	8B45 0C	mov eax,dword ptr ss:[ebp+0xC]	1E给ebp-4
002E7B54	8945 FC	mov dword ptr ss:[ebp-0x4],eax	接下来这段是把xuenixiang放到另一个位置
002E7B57	8B4D F8	mov ecx,dword ptr ss:[ebp-0x8]	假码给edx
002E7B5A	8B55 F4	mov edx,dword ptr ss:[ebp-0xC]	假码第一位给al
002E7B5D	8A02	mov al,byte ptr ds:[edx]	假码第一位给ecx的地址
002E7B5F	8B01	mov byte ptr ds:[ecx],al	假码第一位给ecx的地址再给ecx, 屁话
002E7B61	8B4D F8	mov ecx,dword ptr ss:[ebp-0x8]	假码第一位给edx
002E7B64	0FBE11	movsx edx,byte ptr ds:[ecx]	假码第一位给ebp-40
002E7B67	8955 C0	mov dword ptr ss:[ebp-0x40],edx	假码第一位地址给eax
002E7B6A	8B45 F8	mov eax,dword ptr ss:[ebp-0x8]	向后移位, 指向*后一位
002E7B6D	83C0 01	add eax,0x1	现在指向*的后一位
002E7B70	8945 F8	mov dword ptr ss:[ebp-0x8],eax	
002E7B73	8B4D F4	mov ecx,dword ptr ss:[ebp-0xC]	
002E7B76	83C1 01	add ecx,0x1	
002E7B79	894D F4	mov dword ptr ss:[ebp-0xC],ecx	把uenixiang给ebp-c
002E7B7C	837D C0 00	cmp dword ptr ss:[ebp-0x40],0x0	判断是否为字符串结尾
002E7B80	74 0D	jbe short Exam.002E7B8F	不为零, 不跳
002E7B82	8B55 FC	mov edx,dword ptr ss:[ebp-0x4]	1E给edx
002E7B85	83EA 01	sub edx,0x1	循环次数减一
002E7B88	8955 FC	mov dword ptr ss:[ebp-0x4],edx	把10给ebp-4
002E7B8B	74 02	jbe short Exam.002E7B8F	
002E7B8D	EB C8	jmp short Exam.002E7B57	
002E7B8F	837D FC 00	cmp dword ptr ss:[ebp-0x4],0x0	判断循环了多少次
002E7B93	0F85 D6000000	jnz Exam.002E7C6F	
002E7B99	8B45 08	mov eax,dword ptr ss:[ebp+0x8]	
002E7B9C	C600 00	mov byte ptr ds:[eax],0x0	
002E7B9F	837D 0C FF	cmp dword ptr ss:[ebp+0xC],-0x1	
002E7BA3	74 4B	jbe short Exam.002E7BF0	
002E7BA5	817D 0C FFFFFFFF	cmp dword ptr ss:[ebp+0xC],0xFFFFFFFF	
002E7BAC	74 42	jbe short Exam.002E7BF0	
002E7BAE	837D 0C 01	cmp dword ptr ss:[ebp+0xC],0x1	
002E7BB2	76 3C	jbe short Exam.002E7BF0	
002E7BB4	E8 B06FF4FF	call Exam.0022EB69	

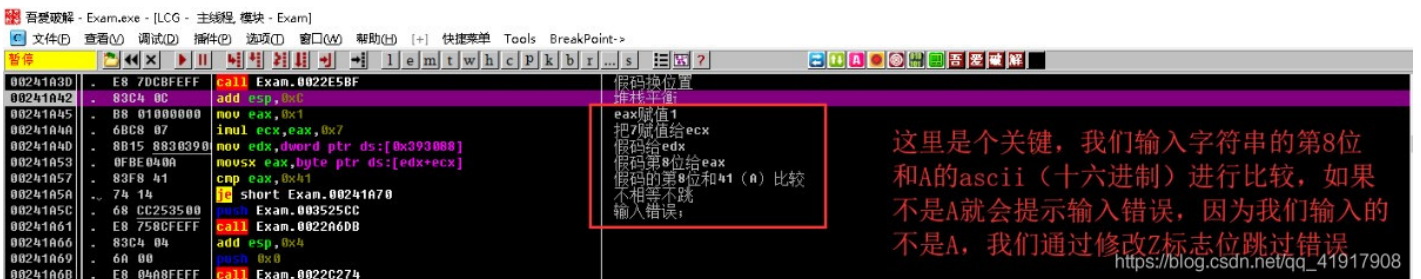
堆栈 ss:[0077F8A4]=008EEB30
ecx=0000001E

https://blog.csdn.net/qq_41917908

这段是在验证假码的长度和假码最高限制1E(十进制的30)的关系



我们出这2个call，看到这个地方，是可以确定flag第8位是A



00241A76处的这个call是整个程序的算法部分，它把我们输入假码的每一位都和1F异或，然后把我们输入的原有的字符串替换



进CALL看看，可以看到异或的具体算法


```

00230B00 > 55      push ebp
00230B01 . 8BEC   mov  ebp,esp
00230B03 . 81EC CC000000 sub  esp,0xCCC
00230B09 . 53     push ebx
00230B0A . 56     push esi
00230B0B . 57     push edi
00230B0C . 80BD 34FFFFFF lea  edi,[local.51]
00230B0E . B9 33000000 mov  ecx,0x33
00230B0F . B8 CCCCCCCC mov  eax,0xCCCCCCCC
00230B10 . F3:AB  rep  stos dword ptr es:[edi]
00230B11 . B9 07603900 mov  ecx,Exam.00396007
00230B12 . E8 0CFBFEFF call Exam.0022D7B4
00230B13 . B8 01000000 mov  eax,0x1
00230B14 . 68C8 07   imul ecx,eax,0x7
00230B15 . 8D55 08   mov  edx,[arg.1]
00230B16 . C60408 23 mov  byte ptr ds:[edx+ecx],0x23
00230B17 . C745 F8 0000 mov  [local.2],0x0
00230B18 . EB 09     jmp  short Exam.00230C19
00230B19 > 8B45 F8   mov  eax,[local.2]
00230B1A . 83C8 01   add  eax,0x1
00230B1B . 8945 F8   mov  [local.2],eax
00230B1C > 8B45 08   mov  eax,[arg.1]
00230B1D . 50     push  eax
00230B1E . E8 84CDFE FF call  Exam.0022A9A6
00230B20 . 83C4 04   add  esp,0x4
00230B21 . 3945 F8   cmp  [local.2],eax
00230B22 . 73 16     jnb  short Exam.00230C40
00230B23 . 8B45 08   mov  eax,[arg.1]
00230B24 . 0345 F8   add  eax,[local.2]
00230B25 . 0FBE 08   movsx ecx,byte ptr ds:[eax]
00230B26 . 83F1 1F   xor  ecx,0x1F
00230B27 . 8D55 08   mov  edx,[arg.1]
00230B28 . 0355 F8   add  edx,[local.2]
00230B29 . 8B0A     mov  byte ptr ds:[edx],cl
00230B2A . EB D0     jmp  short Exam.00230C10
00230B2B > 8B45 08   mov  eax,[arg.1]
00230B2C . 5F     pop  edi

```

Exan.<ModuleEntryPoint>

把33给ecx

把0x23 (#) 给假码第8位

这个部分是每一位和1F异或的过程

求假码长度为A

A和假码长度比较，小于所以不跳

假码给eax

把第一位*给ecx

第一位*和1F异或

假码给edx

把假码第一位*和1F异或的结果给假码第一位

把假码的每一位和1F异或然后再赋值给自己

处理后的假码给eax

https://blog.csdn.net/qq_41917908

注意这个地方，程序会将第八位无条件的换成#号，也就是说我们第八位不管是不是41 (A)，都会先被替换为#再参与下面的异或运算

```

00230B00 > 55      push ebp
00230B01 . 8BEC   mov  ebp,esp
00230B03 . 81EC CC000000 sub  esp,0xCCC
00230B09 . 53     push ebx
00230B0A . 56     push esi
00230B0B . 57     push edi
00230B0C . 80BD 34FFFFFF lea  edi,[local.51]
00230B0E . B9 33000000 mov  ecx,0x33
00230B0F . B8 CCCCCCCC mov  eax,0xCCCCCCCC
00230B10 . F3:AB  rep  stos dword ptr es:[edi]
00230B11 . B9 07603900 mov  ecx,Exam.00396007
00230B12 . E8 0CFBFEFF call Exam.0022D7B4
00230B13 . B8 01000000 mov  eax,0x1
00230B14 . 68C8 07   imul ecx,eax,0x7
00230B15 . 8D55 08   mov  edx,[arg.1]
00230B16 . C60408 23 mov  byte ptr ds:[edx+ecx],0x23
00230B17 . C745 F8 0000 mov  [local.2],0x0
00230B18 . EB 09     jmp  short Exam.00230C19
00230B19 > 8B45 F8   mov  eax,[local.2]
00230B1A . 83C8 01   add  eax,0x1
00230B1B . 8945 F8   mov  [local.2],eax
00230B1C > 8B45 08   mov  eax,[arg.1]
00230B1D . 50     push  eax
00230B1E . E8 84CDFE FF call  Exam.0022A9A6
00230B20 . 83C4 04   add  esp,0x4
00230B21 . 3945 F8   cmp  [local.2],eax
00230B22 . 73 16     jnb  short Exam.00230C40
00230B23 . 8B45 08   mov  eax,[arg.1]
00230B24 . 0345 F8   add  eax,[local.2]
00230B25 . 0FBE 08   movsx ecx,byte ptr ds:[eax]
00230B26 . 83F1 1F   xor  ecx,0x1F
00230B27 . 8D55 08   mov  edx,[arg.1]
00230B28 . 0355 F8   add  edx,[local.2]
00230B29 . 8B0A     mov  byte ptr ds:[edx],cl
00230B2A . EB D0     jmp  short Exam.00230C10
00230B2B > 8B45 08   mov  eax,[arg.1]
00230B2C . 5F     pop  edi

```

寄存器 (MMX)

EAX 00000000

ECX 008EB30 ASCII "jzqvgv<qx"

EDX 7CFF7770

EBX 004F1000

ESP 0077F99C ASCII "jzqvw"

EBP 0077F99C ASCII "jzqvw"

ESI 0022F50F Exam.<ModuleEntryPoint>

EDI 0022F50F Exam.<ModuleEntryPoint>

EIP 00241A92 Exam.00241A92

C 0 ES 002B 32位 0(FFFFFFFF)

P 1 CS 0023 32位 0(FFFFFFFF)

A 0 SS 002B 32位 0(FFFFFFFF)

Z 1 DS 002B 32位 0(FFFFFFFF)

S 0 FS 0053 32位 4F4000(FFF)

T 0 GS 002B 32位 0(FFFFFFFF)

D 0

0 0 LastErr ERROR_SUCCESS (00000000)

EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)

MM0 0000 0000 0000 0000

MM1 0000 0000 0000 0000

MM2 0000 0000 0000 0000

MM3 0000 0000 0000 0000

MM4 0000 0000 0000 0000

MM5 0000 0000 0000 0000

MM6 0000 0000 0000 0000

MM7 0000 0000 0000 0000

https://blog.csdn.net/qq_41917908

异或的结果保存在ecx

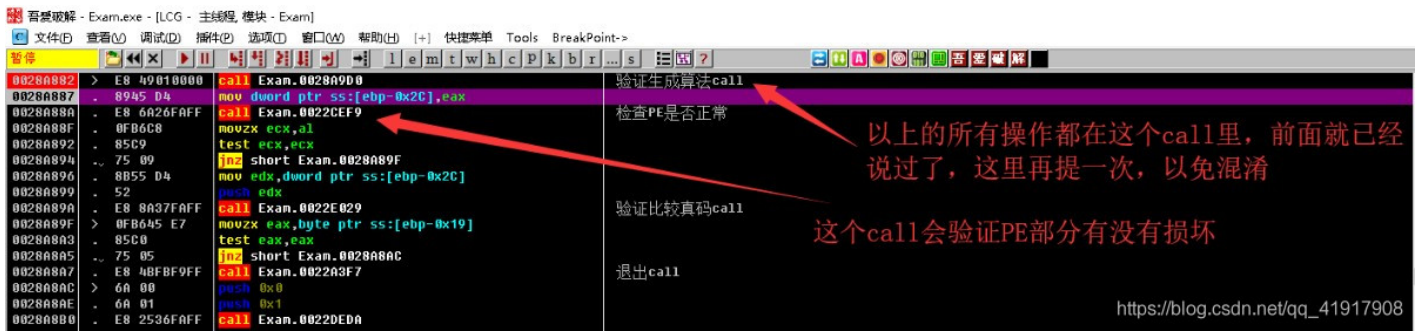
```

寄存器 (MMX)
EAX 00000000
ECX 008EB30 ASCII "jzqvgv<qx"
EDX 7CFF7770
EBX 004F1000
ESP 0077F99C ASCII "jzqvw"
EBP 0077F99C ASCII "jzqvw"
ESI 0022F50F Exam.<ModuleEntryPoint>
EDI 0022F50F Exam.<ModuleEntryPoint>
EIP 00241A92 Exam.00241A92
C 0 ES 002B 32位 0(FFFFFFFF)
P 1 CS 0023 32位 0(FFFFFFFF)
A 0 SS 002B 32位 0(FFFFFFFF)
Z 1 DS 002B 32位 0(FFFFFFFF)
S 0 FS 0053 32位 4F4000(FFF)
T 0 GS 002B 32位 0(FFFFFFFF)
D 0
0 0 LastErr ERROR_SUCCESS (00000000)
EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)
MM0 0000 0000 0000 0000
MM1 0000 0000 0000 0000
MM2 0000 0000 0000 0000
MM3 0000 0000 0000 0000
MM4 0000 0000 0000 0000
MM5 0000 0000 0000 0000
MM6 0000 0000 0000 0000
MM7 0000 0000 0000 0000
https://blog.csdn.net/qq_41917908

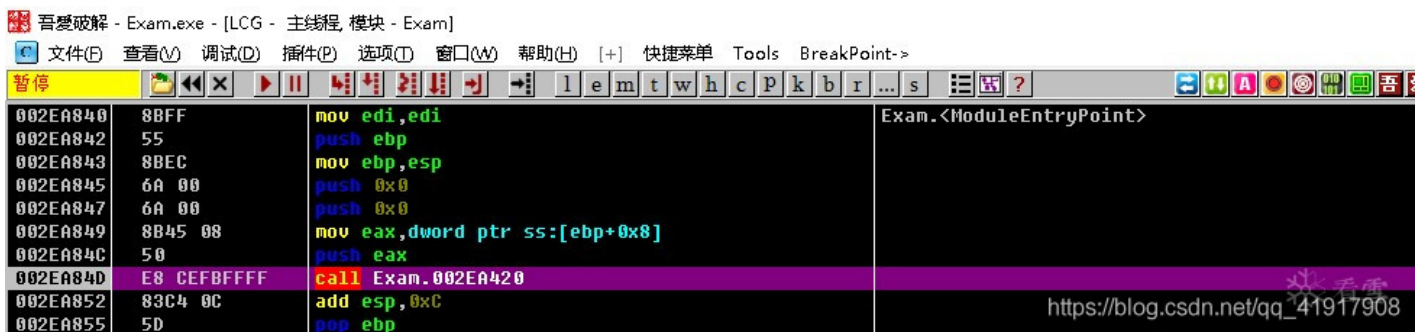
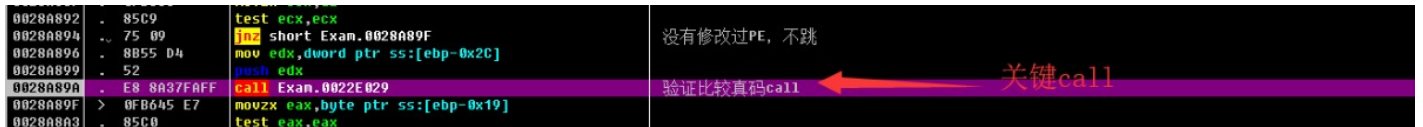
```

异或结果

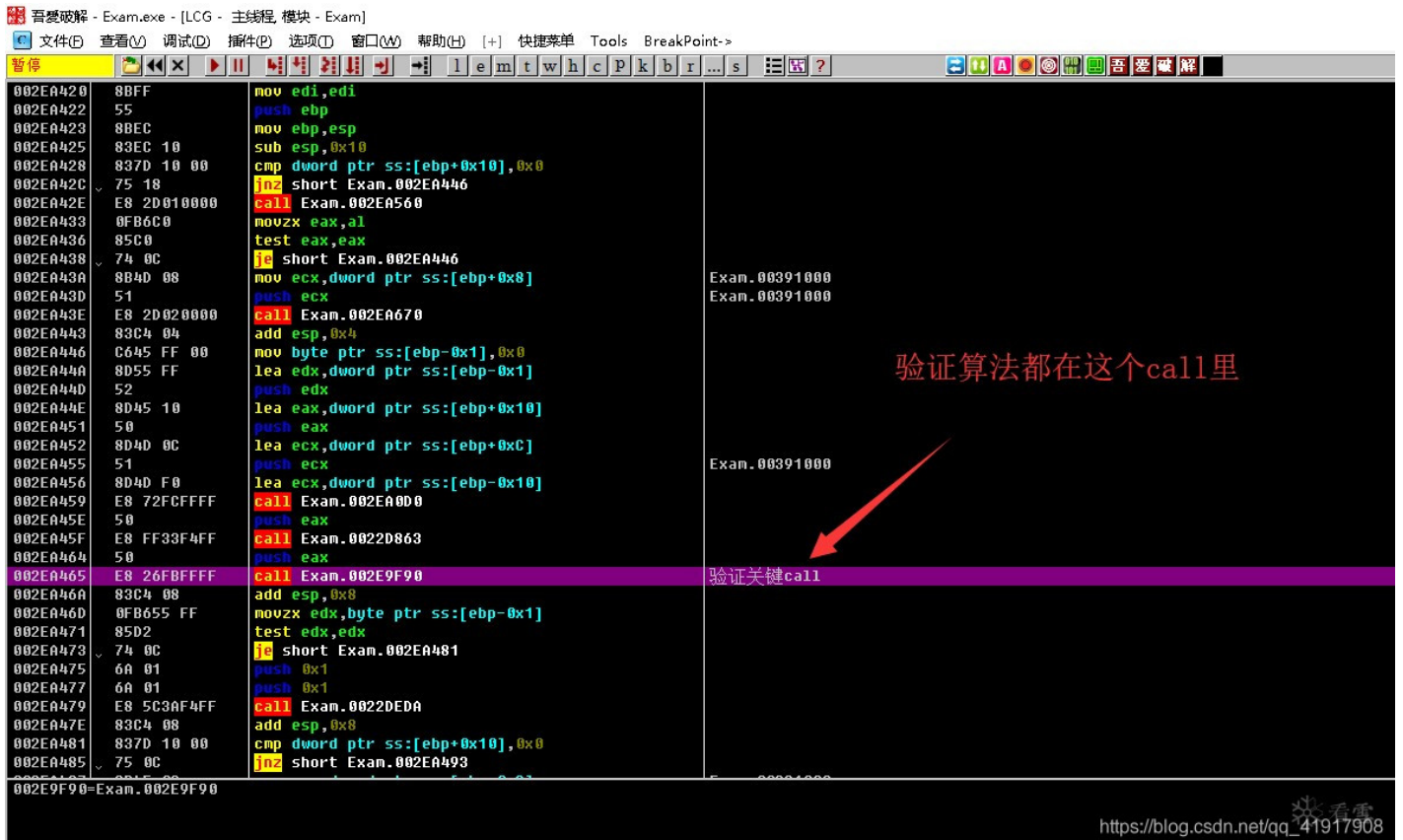
0028A88A 这个call是检查PE是否正常，是否被修改过（会检测PE签名，NT头，等重要的地方）



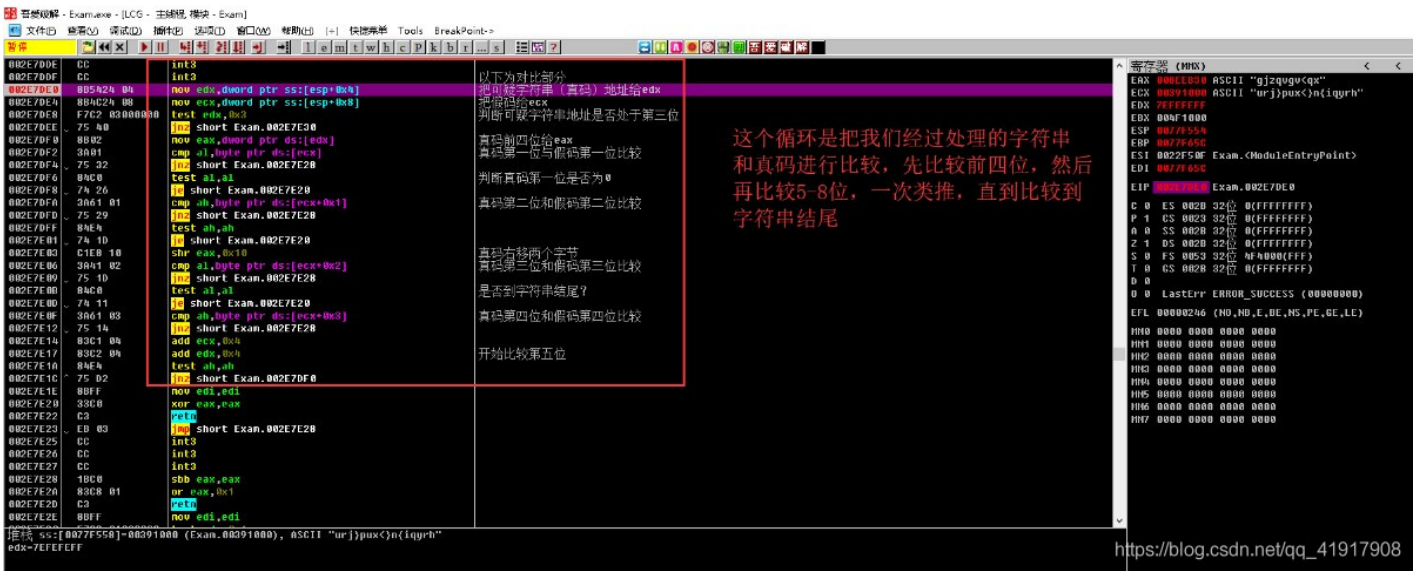
下面进28A89A这个关键call



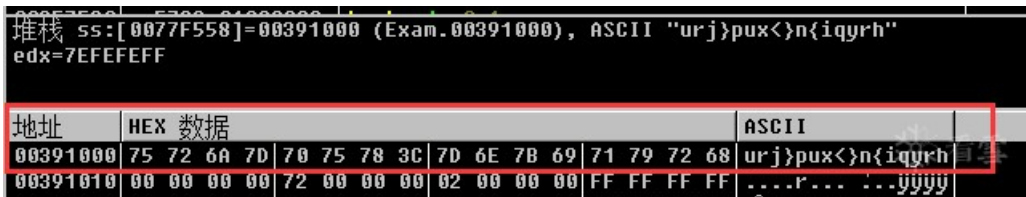
继续进call，往下跟会找到2EA465位置的这个call



进去后就是我们程序最关键的比较部分了



这里我们看到了用来比较的真码



00391000 75 72 6A 7D 70 75 78 3C 7D 6E 7B 69 71 79 72 68 urj}pux<}n{iqyrh

xor 1E

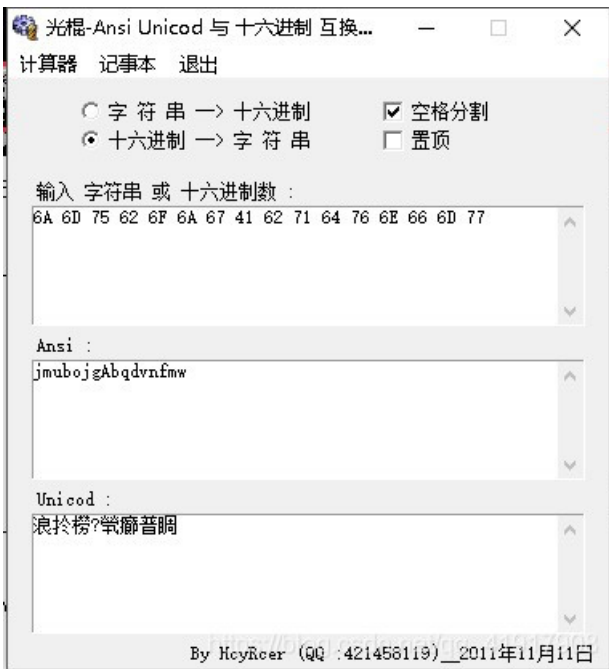
6A 6D 75 62 6F 6A 67 23 62 71 64 76 6E 66 6D 77

41

我们用计算器把 75 72 6A 7D 70 75 78 3C 7D 6E 7B 69 71 79 72 68 每一位和1E进行异或运算, 得到6A 6D 75 62 6F 6A 67 23 62 71 64 76 6E 66 6D 77

又因为我前面提到flag的第8位是41(A), 所以 flag应该是6A 6D 75 62 6F 6A 67 41 62 71 64 76 6E 66 6D 77

我们将 6A 6D 75 62 6F 6A 67 41 62 71 64 76 6E 66 6D 77 转换为ascii 得到jmubojgAbqdvnmfw



输入jmubojgAbqdvnmfw 软件提示了OK



此 writeup若有不正确的地方，欢迎大佬指正~~

2018.12.3

www.xuenixiang.com



[创作打卡挑战赛](#)
[赢取流量/现金/CSDN周边激励大奖](#)