Windows 进程管理作业

1. 简述加密壳的主要机制。

   加密壳相比压缩壳，更侧重于保护程序免受逆向分析。通常加密壳会以保护软件为目的，根据用户输入的密码用相应的加密算法对原程序进行处理，将程序代码混淆加密，以达到防破解的目的。在加壳中运用了多种防止代码逆向分析的技术，一些功能强大的加密壳可以给程序添加一些额外功能，包括限制软件使用时间，给软件添加注册功能等。

2. 简述源码级混淆的主要方法。

   （1）标识符重命名

   写代码的时候常常要求给变量、函数取有意义的名字，便于编程即将代码中的各种元素，如变量，函数，类的名字改写成无意义的名字。比如改写单个字母或者数字，又或者字母和数字的组合等，使得阅读的人无法根据名字猜测其用途。

   （2）等价表达式

   重写代码中的部分逻辑，将其变成功能上等价，但是更难理解的形式。比如将循环改成递归，精简中间变量等。

   （3）代码重排

   打乱原有代码格式。比如将多行代码挤到一行代码中。

   （4）花指令

   通过构造字节码插入程序的适当位置，使得反汇编器出错，产生无法反编译或者反编译出错的情况。

   （5）自解密

   通过对程序部分进行加密，在即将运行时代码进行自解密，然后执行解密之后的代码。

3. 简述 SEH 链结构。

   _EXCEPTION_REGISTRATION_RECORD 结构体称为异常处理器，有 Next 和 Handler 两个成员，以链表形式存在。Handler 部分作为函数指针指向异常处理函数，Next 作为结构体指针指向下一个异常处理器结构体。

4. 编写程序并生成 exe 文件，要求使用 Loadlibrary 函数和 GetProcAddress 函数调用 masm32 下的 DLL 示例，静态调用 MessageBox 函数。在 OD 中查看进程空间分布。

   代码：

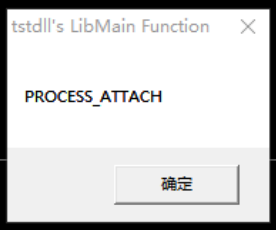
```
.386
.model flat,stdcall
option casemap:none
.stack 4096
```

```
include windows.inc
includelib user32.lib
includelib kernel32.lib
includelib masm32.lib
include user32.inc
include kernel32.inc
ExitProcess PROTO, dwExitCode:DWORD
.data
msgA db 'Cannot find this dll', 0          ;Messagebox1
msgB db 'Cannot find this function', 0     ;Messagebox2
LibName db 'tstdll.dll', 0                 ;要调用的动态链接库名称
FuncName db 'TestProc', 0                  ;调用的函数名称
hLib dd ?                                  ;存储动态链接库的句柄
TestFunc dd ?                              ;存储函数地址
.code
main proc
    invoke LoadLibrary, offset LibName                    ;调用 LoadLibrary，参数为动态链接库名称，此处为 tstdll.dll
    .if eax ==NULL
        invoke MessageBox,NULL,offset msgA, NULL, MB_OK  ;找不到 dll
    .else
        mov hLib, eax                                     ;将 eax 中的动态链接库句柄放入 hLib
        invoke GetProcAddress, hLib, offset FuncName      ;将句柄和函数名称传给 GetProcAddress
        .if eax ==NULL
            invoke MessageBox,NULL,offset msgB, NULL, MB_OK  ;找不到函数
        .else
            mov TestFunc, eax                             ;函数地址存入 TestFunc
            call [TestFunc]                               ;调用函数
        .endif
        invoke FreeLibrary,hLib
    .endif
    invoke ExitProcess, 0
main endp
    end main
```

运行截图：

hm1121.asm

```
include user32.inc
include kernel32.inc
ExitProcess PROTO, dwExitCode:DWORD

.data
msgA db 'Cannot find this dll', 0              ;Messagebox1
msgB db 'Cannot find this function', 0         ;Messagebox2
LibName db 'tstdll.dll', 0                     ;要调用的动态链接库名称
FuncName db 'TestProc', 0                      ;调用的函数名称
hLib dd ?                                      ;存储动态链接库的句柄
TestFunc dd ?                                  ;存储函数地址

.code

main proc

    invoke LoadLibrary, offset LibName                    ;调用LoadLibrary，参数为动态链接库名称，此处为t
    .if eax ==NULL
        invoke MessageBox,NULL,offset msgA, NULL, MB_OK   ;找不到dll
    .else
```

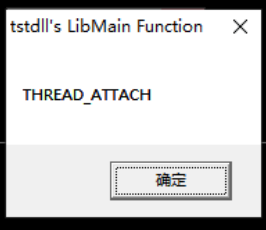


hm1121.asm

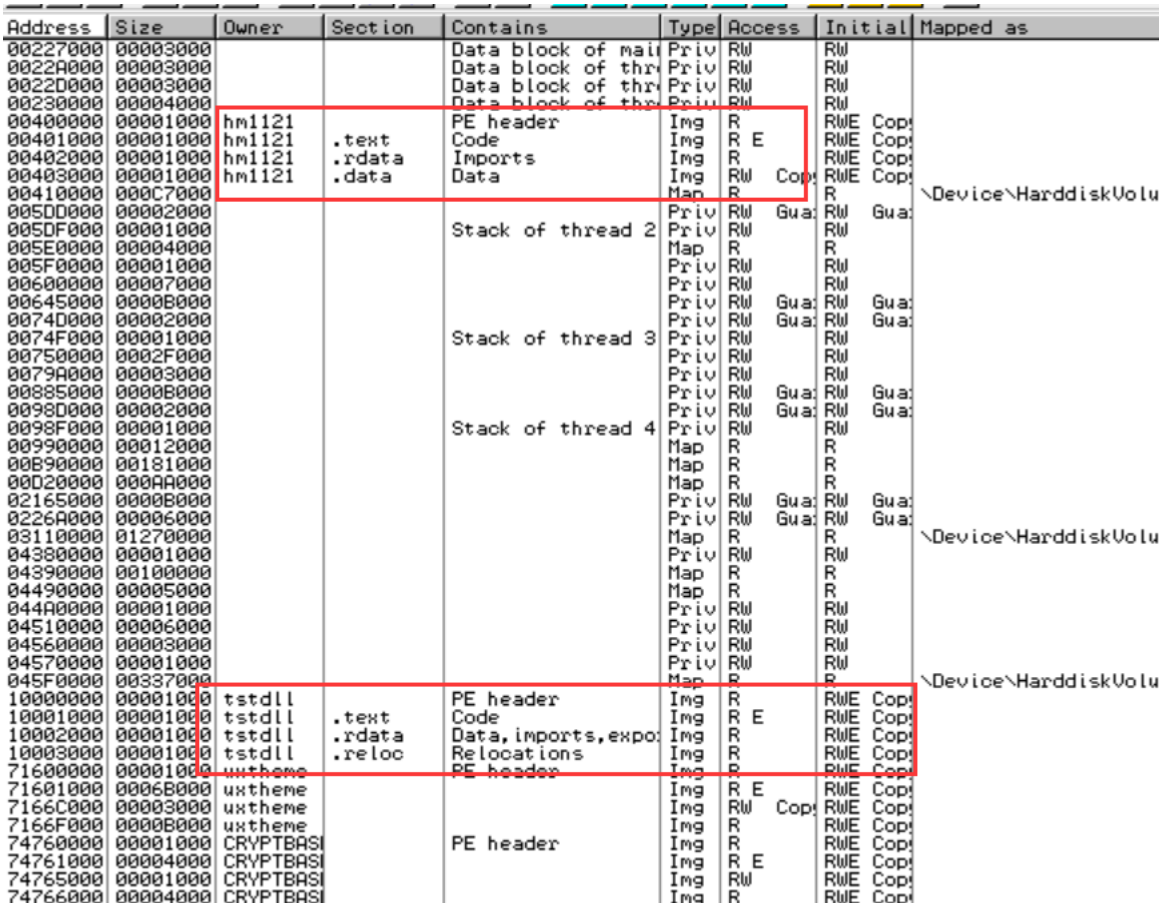
```
include user32.inc
include kernel32.inc
ExitProcess PROTO, dwExitCode:DWORD

.data
msgA db 'Cannot find this dll', 0              ;Messagebox1
msgB db 'Cannot find this function', 0         ;Messagebox2
LibName db 'tstdll.dll', 0                     ;要调用的动态链接库名称
FuncName db 'TestProc', 0                      ;调用的函数名称
hLib dd ?                                      ;存储动态链接库的句柄
TestFunc dd ?                                  ;存储函数地址

.code

main proc

    invoke LoadLibrary, offset LibName                    ;调用LoadLibrary，参数为动态链接库名称，此
```

hm1121.asm

```
include user32.inc
include kernel32.inc
ExitProcess PROTO, dwExitCode:DWORD

.data
msgA db 'Cannot find this dll', 0              ;Messagebox1
msgB db 'Cannot find this function', 0         ;Messagebox2
LibName db 'tstdll.dll', 0                     ;要调用的动态链接库名称
FuncName db 'TestProc', 0                      ;调用的函数名称
hLib dd ?                                      ;存储动态链接库的句柄
TestFunc dd ?                                  ;存储函数地址

.code

main proc

    invoke LoadLibrary, offset LibName                    ;调用LoadLibrary，参数为动态链接库名称，此处为tstdl
    .if eax ==NULL
```



MemoryMap



Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
00227000	00003000			Data block of mai	Priv	RW	RW	
0022A000	00003000			Data block of thr	Priv	RW	RW	
0022D000	00003000			Data block of thr	Priv	RW	RW	
00230000	00004000			Data block of thr	Priv	RW	RW	
00400000	00001000	hm1121		PE header	Img	R	RWE Cop	
00401000	00001000	hm1121	.text	Code	Img	R E	RWE Cop	
00402000	00001000	hm1121	.rdata	Imports	Img	R	RWE Cop	
00403000	00001000	hm1121	.data	Data	Img	RW Cop	RWE Cop	
00410000	000C7000				Map	R	R	\Device\HarddiskVolu
005DD000	00002000				Priv	RW Gua	RW Gua	
005DF000	00001000			Stack of thread 2	Priv	RW	RW	
005E0000	00004000				Map	R	R	
005F0000	00001000				Priv	RW	RW	
00600000	00007000				Priv	RW	RW	
00645000	0000B000				Priv	RW Gua	RW Gua	
0074D000	00002000				Priv	RW Gua	RW Gua	
0074F000	00001000			Stack of thread 3	Priv	RW	RW	
00750000	0002F000				Priv	RW	RW	
0079A000	00003000				Priv	RW	RW	
00885000	0000B000				Priv	RW Gua	RW Gua	
0098D000	00002000				Priv	RW Gua	RW Gua	
0098F000	00001000			Stack of thread 4	Priv	RW	RW	
00990000	00012000				Map	R	R	
00B90000	00181000				Map	R	R	
00D20000	000AA000				Map	R	R	
02165000	0000B000				Priv	RW Gua	RW Gua	
0226A000	00006000				Priv	RW Gua	RW Gua	
03110000	01270000				Map	R	R	\Device\HarddiskVolu
04380000	00001000				Priv	RW	RW	
04390000	00100000				Map	R	R	
04490000	00005000				Map	R	R	
044A0000	00001000				Priv	RW	RW	
04510000	00006000				Priv	RW	RW	
04560000	00003000				Priv	RW	RW	
04570000	00001000				Priv	RW	RW	
045F0000	00337000				Map	R	R	\Device\HarddiskVolu
10000000	00001000	tstdll		PE header	Img	R	RWE Cop	
10001000	00001000	tstdll	.text	Code	Img	R E	RWE Cop	
10002000	00001000	tstdll	.rdata	Data,imports,expo	Img	R	RWE Cop	
10003000	00001000	tstdll	.reloc	Relocations	Img	R	RWE Cop	
71600000	00001000	uxtheme		PE header	Img	R	RWE Cop	
71601000	0006B000	uxtheme			Img	R E	RWE Cop	
7166C000	00003000	uxtheme			Img	RW Cop	RWE Cop	
7166F000	0000B000	uxtheme			Img	R	RWE Cop	
74760000	00001000	CRYPTBASE		PE header	Img	R	RWE Cop	
74761000	00004000	CRYPTBASE			Img	R E	RWE Cop	
74765000	00001000	CRYPTBASE			Img	RW	RWE Cop	
74766000	00004000	CRYPTBASE			Img	R	RWE Cop	