九、软件保护技术

复习

- **演示版保护技术**
- **软件加壳与脱壳**
- **大量现成的保护方案**
  - 基于软件的加密壳保护
  - 基于硬件的加密锁保护

防范算法求逆 – 基本概念

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- **软件注册码**
  - 用户向软件作者提交用户码U，申请注册
  - 软件作者计算出注册码R=f(U)，将结果返回合法用户
  - 用户在软件注册界面输入U和R
  - 软件通过验证F(U,R)的值是否合法来判定用户的合法性

防范算法求逆 – 基本概念

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 用户码U：用于区别用户身份
- 注册码R：用于验证用户身份
- 注册机：R=f(U)中的f称为注册机。掌握了注册机，就有能力针对任何用户码计算出相应的注册码
- 验证函数：F(U,R)称为验证函数。软件使用验证函数来验证注册码的合法性，即：当且仅当R=f(U)时，F(U,R)是合法值
- 算法求逆：解密者通过验证函数F推导注册机f的过程称为算法求逆。因此，验证函数F的构造是非常重要的

防范算法求逆 – 基本概念

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- **问题**
  - 验证函数与注册机没有本质区别，即 $F(U,R)=f(U)-R$
  - 验证函数自身包含注册机
- **改进**
  - 先求出f的反函数f^{-1}，使$U=f^{-1}(R)$
  - 令$F(U,R)=f^{-1}(R)-U$
  - 软件本身不包含注册机

防范算法求逆 – 堡垒战术

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- MD5算法和RSA算法非常适合在软件注册算法中使用
- 但MD5算法也不适合直接作为注册机使用
- 可以通过以下方法使用MD5算法
  - 设注册机f为R=f(U)
  - 设MD5(a)=b
  - 令验证函数F为F(U,R)=MD5[f^{-1}(R)-U+a]
- 缺陷
  - 因为MD5完全不可逆，所以a必须为常数
  - 一旦解密者获得了一对合法的U、R，就可以跟踪得到合法的a、b值
  - 进而推导出注册机

防范算法求逆 – 堡垒战术

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- RSA算法可以很容易地应用在软件保护中
  - 使用$R=U^d \bmod n$作为注册机
  - 使用$U=R^e \bmod n$作为验证函数
- **潜在风险**
  - RSA算法本身足够坚固，但使用者往往直接采用第三方公用代码，这些代码中可能含有漏洞
  - RSA算法的使用者往往不了解算法的细节，可能因错误使用RSA算法而在不知不觉中遭遇非常规手段的攻击
  - 某些函数库在生成随机素数时采用了伪随机数产生器即在完全相同的初始条件下会产生完全相同的随机数序列
  - RSA算法中存在若干由某些特殊素数构造而成的“弱密钥”。某些函数库在生成随机素数时没有淘汰这些特殊素数，导致RSA的防线在数论高手面前崩溃

防范算法求逆 – 游击战术

化整为零

- 软件保护中的“游击战术”就是将验证函数F分解成不同的多个F_i，然后尽可能将这些F_i隐藏到程序中去
- 通过任意一个F_i的验证只是使注册码合法的必要条件，而非充分条件，真正合法的注册码能够通过所有F_i的验证
- 即使解密者找到了F_i中的任意一个或多个，只要不能将一网打尽，就无法一睹F的全貌，也无法进行算法求逆

例如

- 将R切分成多段R_i
- 构造不同的f算法，使$R_i=f_i(U)$
- 令$F_i=f_i^{-1}$

防范算法求逆 – 游击战术

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 例如，可以让F_1使用MD5算法，让F_2使用RSA算法，让F_3使用自定义算法
- 在用户输入注册码后仅使用F_1进行验证，并将注册码以密文形式写入自定义格式的数据文件中，如果验证通过，就显示注册成功的提示
- 将另外两个验证函数藏起来，只有在使用者执行特定的操作时才调用它们
- 例如，在用户进行存档操作或使用某些高级功能的时候将注册码读出，再次进行验证
- 一旦有验证函数发现注册码非法，就清除注册码并将软件恢复为未注册状态

防范算法求逆 – 游击战术

- 虚虚实实

设 $R_a = 3U$，$R_b = 5U$，$R_c = 7U$，则

$$F_a = 7R_a + 11R_b + 5R_c - 111U$$
$$F_b = 11R_a + 7R_b + 3R_c - 89U$$
$$F_c = 5R_a + 3R_b + 11R_c - 107U$$

- 不管是数论、代数、线性代数、几何、解析几何，还是微积分、概率论，都可以作为软件保护的武器

防范算法求逆 – 游击战术

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

战略转移

- 每一个验证函数都必须访问注册码，而注册码的来源只有一个
- 解密者会跟踪程序从注册界面读取注册码的过程，并监控存放注册码的内存地址
- 一旦验证函数访问这个地址，就会泄露行踪
- 应对的办法就是进行大规模转移，即软件必须不停地给注册码“搬家”，而且“搬家”的方法要多样化

防范算法求逆 – 游击战术

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 内存复制。这种常规做法容易被解密者通过内存监视断点识破
- 写入注册表或文件，然后将代码读入另一个内存地址。这种方法会被解密者的注册表和文件监视工具识破
- 同时将注册码复制到多个地址，让解密者无法确定哪一个地址才是注册码的“新家”
- 在反复使用某个函数“搬家”后，突然使用另一个前半部分代码与该函数相同而后半部分代码与该函数不同的函数“搬家”
- 反复使用以上方法

抵御静态分析

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 静态分析是指使用通过反汇编得到的程序清单来分析程序流程

- 反静态分析技术主要从扰乱汇编代码的可读性入手，例如使用大量的花指令、将提示信息隐藏等

抵御静态分析 – 花指令

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 反汇编过程中存在几个关键问题，其中之一是数据与代码的区分问题。反汇编算法必须对汇编指令长度、多种多样的间接跳转实现形式进行适当的处理，从而保证反汇编结果的正确性

工具名	反汇编算法
OllyDbg	Linear Sweep/Recursive Traversal
SoftICE	Linear Sweep
WinDBG	Linear Sweep
W32Dasm	Linear Sweep
IDA Pro	Recursive Traversal

抵御静态分析 – 花指令

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 线性扫描算法的技术含量不高，反汇编工具将整个模块中的每一条指令都反汇编成汇编指令，将遇到的机器码都作为代码处理，没有对所反汇编的内容进行任何判断

- 线性扫描算法无法正确地将代码和数据分开，数据也被当成代码来解码，从而导致反汇编出现错误。这种错误将影响对下一条指令的正确识别，会使整个反汇编都出现错误

抵御静态分析 – 花指令

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 递归行进算法按照代码可能的执行顺序来反汇编程序，对每条可能的路径进行扫描

- 当解码出分支指令后，反汇编工具就将这个地址记录下来，并分别反汇编各个分支中的指令。这种算法比较灵活，可以避免将代码中的数据作为指令来解码

抵御静态分析 – 花指令

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 巧妙构造代码和数据，在指令流中插入很多“数据垃圾”，干扰反汇编软件的判断，使它错误地确定指令的起始位置，这类代码数据称为花指令

- 不同的机器指令包含的字节数并不相同，有的是单字节指令，有的是多字节指令。对多字节指令，反汇编软件需要确定指令的第1个字节的起始位置，也就是操作码的位置，才能正确地反汇编这条指令，否则，它就可能被反汇编成另外一条指令了

抵御静态分析 – 花指令

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

```
start_:
    xor  eax,eax
    test eax,eax
    jz   label1
    jnz  label1
    db   0E8h              ;注意这里
label1:
    xor   eax,3
    add   eax,4
    xor   eax,5
    ret
```

抵御静态分析 – 花指令

- Linear Sweep式反汇编软件是逐行反汇编的，代码中的垃圾数据“0E8h”干扰了其工作，错误地确定了指令的起始位置，导致反汇编的一些跳转指令所跳转的位置无效

```
:00401000 33C0              xor  eax, eax
:00401002 85C0              test eax, eax
:00401004 7403              je   00401009
:00401006 7501              jne  00401009
:00401008 E883F00383        call 83440090
:0040100D C00483F0          rol byte ptr [ebx+4*eax], F0
:00401011 05C3000000        add eax, 000000C3
```

抵御静态分析 – 花指令

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 在递归行进算法中，一个十分重要的假设是：对任意一条控制转移指令，都能确定其后继（即转移）的目的地址

```
start_:
    xor     eax,eax
    test    eax,eax
    jz      label1
    jnz     label0                  ;指向无效的跳转指令
label0:
        db  0E8h
label1:
    xor     eax,3
    add     eax,4
    xor     eax,5
    ret
end start_
```

抵御静态分析 – SMC

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

```
;-=-=-=-解密数据-=-=-=-=-
push    DataLen
push    offset EnData
call    DecryptFunc

EnData  BYTE    0ECh,01Bh,054h,076h,064h,064h,066h,074h,074h,001h


        BYTE    04Ah,021h,06Dh,070h,077h,066h,021h,075h,069h,06Ah
        BYTE    074h,021h,068h,062h,06Eh,066h,022h,001h,060h,0E9h
        BYTE    001h,001h,001h,001h,05Eh,082h,0EEh,023h,011h,041h
        BYTE    001h,06Bh,001h,08Eh,086h,003h,011h,041h,001h,051h
        BYTE    08Eh,086h,00Bh,011h,041h,001h,051h,06Bh,001h,000h
        BYTE    0D8h,08Eh,0BEh,001h,011h,041h,001h,0BAh,04Eh,001h
        BYTE    001h,001h,034h,0C1h,0FDh,0F4h,0ABh

DataLen     EQU $ - offset EnData       ;$
lm_exit:    invoke  ExitProcess,0
```

抵御静态分析 – SMC

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

```
CodeBegin:   jmp      loc_begin
szTitle      BYTE'Success',0
szMsg        BYTE'I love this game!',0
loc_begin:
    ;-------从栈中得到 MessageBoxA 函数的地址--------
    pop      edi
    ;-------计算运行时的地址与编译地址之差--------
    call     loc_next
loc_next:
    pop      ebp
    sub      ebp,offset loc_next
    ;-------调用 MessageBoxA 函数显示信息--------
    push     MB_OK
    lea      eax,[ebp + szTitle]
    push     eax
    lea      eax,[ebp + szMsg]
    push     eax
    push     NULL
    call     edi
    ;-------把自身代码清零--------
    lea      edi,[ebp + CodeBegin]
    mov      ecx,CodeLen
    xor      eax,eax
    cld
    rep      stosb
CodeEnd:
```

抵御静态分析 – SMC

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- “SMC”是“Self-Modifying Code”的缩写
- 利用SMC技术的这个特点，在设计加密方案时，可以把代码以加密形式保存在可执行文件中，然后在程序执行时动态解密，这样可以有效地“对付”静态分析
- 如果要了解被加密代码的功能，就只能动态跟踪或者分析出解密函数的位置并编写程序来解密这些代码

抵御静态分析 – SMC

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 可以在解密函数中把代码的解密与自身保护及反单步跟踪、反断点跟踪结合起来
- 可以利用SMC技术设计出多层嵌套加密的代码。在第1层代码中解密第2层代码，在第2层代码中解密第3层代码，依此类推
- 可以设计一个比较复杂的解密函数。针对在最外层的解密函数，还可以把它分散在程序中的多个地方，使其隐蔽性更强

抵御静态分析 – 信息隐藏

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 大多数软件在设计时都采用了人机对话方式
- 所谓人机对话，是指软件在运行过程中在需要由用户选择的地方显示相应的提示信息并等待用户按键选择，在执行一段程序之后显示反映该段程序运行后状态的提示信息（是正常运行，还是出现错误）或者显示提示用户进行下一步工作的帮助信息
- 解密者可以根据这些提示信息迅速找到核心代码。基于对安全性的考虑，需要对这些敏感信息进行隐藏处理

抵御静态分析 – 信息隐藏

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

```
if condition then
   showmessage(0, 'You see me!', 'You see me!', 0);
```

```
:00401110 85C0                  test eax, eax                    ;判断
:00401112 755B                  jne 0040116F
:00401114 50                    push eax
* Possible StringData Ref from Data Obj ->"You see me!"
:00401115 6838504000            push 00405038
* Possible StringData Ref from Data Obj ->"You see me!"
:0040111A 6838504000            push 00405038
:0040111F 50                    push eax
* Reference To: USER32.MessageBoxA, Ord:01BEh
:00401120 FF15A0404000          Call dword ptr [004040A0]
```

```
:00401110 85C0                  test eax, eax                    ;判断
:00401112 755B                  jne 0040116F
:00401114 50                    push eax
* Possible StringData Ref from Data Obj ->"Tbx-"
:00401115 6838504000            push 00405038
* Possible StringData Ref from Data Obj ->" Tbx-"
:0040111A 6838504000            push 00405038
:0040111F 50                    push eax
* Reference To: USER32.MessageBoxA, Ord:01BEh
:00401120 FF15A0404000          Call dword ptr [004040A0]
```

抵御静态分析 – 信息隐藏

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 对一些关键信息进行隐藏处理，可以在一定程度上增加静态反编译的难度
- 例如，对“软件已经过期，请购买”等数据进行隐藏处理，可以有效防止解密者根据这些信息利用静态反汇编快速找到程序判断点进行破解
- 信息隐藏的实现思路很多，例如将要显示的字符加密存放，在需要时将其解密并显示出来

抵御静态分析 – 多态变形

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 病毒的世界里充斥着多态（Polymorphic)、变形（Metamorphic)和混淆（Ohfucscation)
- 多态是第一种对杀毒软件造成严重威胁的技术
  - 一个多态病毒在触发时由解密模块进行解密，在感染时由加密模块进行加密并感染，但其加密模块在每一次的感染中会有所修改
  - 一个仔细设计的多态病毒在每一次感染中没有一个部分是相同的
  - 使得使用病毒特征码进行侦测变得困难
- 变形是为了避免被杀毒软件通过模拟环境或“蜜罐”系统而被查杀。变形病毒在每一次的感染都完全将其自身改写

抵御静态分析 – 多态变形

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 多态
  - 多态引擎往往意味着将代码用某种算法编码（算法可能是即时生成的，也可能是密码学算法）
  - 引擎会生成一个充满干扰指令的解码器，以便在运行时对代码进行解码
- 变形
  - 变形则是把一段代码重新编码。虽然仍然使用x86指令集，但是像“add eax, 5”这样的指令可能被换成等价的5个“inc eax”指令，而且可以用jmp指令打乱代码的顺序。此类变换会使代码迅速膨胀
- 混淆通常以多态变形引擎的垃圾生成器出现，一般是指一些花指令和无用的代码。有些时候，变形也叫作混淆

文件完整性检验

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 在软件保护方案中，建议增加对软件自身完整性的检查，以防止解密者修改程序，进而达到破解的目的

- 文件完整性检验的原理就是文件发布时用散列函数计算文件的散列值，并将此值放在某处，以后文件每次运行时，重新计算文件的散列值，并与原散列值进行比较，以判断文件是否被修改了

文件完整性检验 - 磁盘

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- CRC算法可以对字符串进行CRC32转换，得到一个4字节的CRC32值
- 当要实现完整性校验时，
  - 首先将需要校验的文件当成一个字符串，计算该字符串的CRC32值
  - 然后在文件的某个地方储存这个CRC32值
  - 当文件运行时，对文件重新进行CRC32计算，将结果与储存的原CRC32值进行比较
  - 如果文件有改动，CRC32值就会有变化

文件完整性检验 - 磁盘

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is program canno
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be run in DOS
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode....$.......
00000080	2C	46	F0	F6	68	27	9E	A5	68	27	9E	A5	68	27	9E	A5	,F[illegible]h'[illegible]h'[illegible]h'[illegible]
00000090	80	38	94	A5	7E	27	9E	A5	EB	3B	90	A5	61	27	9E	A5	€8[illegible]~'[illegible]?[illegible]a'[illegible]
000000A0	0A	38	8D	A5	6D	27	9E	A5	68	27	9F	A5	42	27	9E	A5	.8[illegible]m'[illegible]h'[illegible]B'[illegible]
000000B0	80	38	95	A5	6C	27	9E	A5	52	69	63	68	68	27	9E	A5	€8[illegible]l'[illegible]Richh'[illegible]
000000C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000D0	50	45	00	00	4C	01	03	00	9B	A6	C0	3F	00	00	00	00	PE..L...[illegible]?....

PE 文件头（计算 CRC32 值的起始地址）

此处存放 CRC32 值

文件完整性检验 - 磁盘

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 先写一个第三方程序，用这个程序打开一个目标文件，然后计算目标文件（从PE文件头开始处到文件结束处的数据）的CRC32值，并把这个CRC32值写入PE文件头之前的空白处
- 读取自身文件从PE文件头开始的所有内容，将其储存在一个字符串中。对这个字符串进行CRC32转换，得到“原始”文件的CRC32值
- 读取自身文件先前储存的CRC32值（PE文件头的前一个字段)
- 比较两个CRC32值，如果相等，说明文件没有被修改，反之说明文件已经被修改了

文件完整性检验 - 磁盘

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

校验和

- 在PE的可选映像头（IMAGE_OPTIONAL_HEADER)里有一个Checksum字段，该字段中存储了该文件的校验和
- EXE文件的校验和可以是0，但一些重要的文件、系统DLL及驱动文件必须有一个非0校验和

```
ULONG MapFileAndCheckSumA(
  IN LPSTR Filename,                    //文件名
  OUT LPDWORD HeaderSum,                //指向 PE 文件头的 Checksum
  OUT LPDWORD new_checksum              //指向新计算出来的 Checksum
);
```

文件完整性检验 - 内存映像校验

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 磁盘文件完整性校验可以抵抗解密者直接修改磁盘文件，但对内存补丁没有效果，必须对内存关键代码数据进行校验
- 每个程序至少有一个代码区块和一个数据区块
  - 从内存映像中得到PE的相关数据，例如代码区块的RVA和内存大小等
  - 根据得到的代码区块RVA和内存大小，计算其内存数据的CRC32值
  - 读取自身文件先前储存的CRC32值（PE文件头的前一个字段)
  - 比较两个CRC32值

代码与数据结合

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 在一般的程序中，代码与数据是分开的
- 将序列号与程序代码相结合
- 在不知道正确注册码的情况下很难被破解
- 其实现原理是将特征数据（例如序列号）与程序的某些关键代码或数据联系起来，例如用序列号或其散列值对程序的关键代码或数据进行解密

代码与数据结合

- 在软件程序中有一段加密的密文C。C既可以是注册版本中的一段关键代码，也可以是使用注册版程序的某个功能所必需的数据
- 当用户输入用户名和序列号之后，计算解密用的密钥：密钥=F（用户名，序列号）
- 对密文进行解密：明文M=Decrypt（密文C，密钥）
- 给解密的代码加上异常处理代码。如果序列号不正确，产生的垃圾代码一定会导致异常；如果序列号正确，则生成代码正常，不会导致异常
- 利用某种散列算法计算出明文M的校验值：Hash（明文M）。散列算法可以采用MD5、SHA等。然后，检查校验值是否正确。如果校验值不正确，说明序列号不正确，就拒绝执行

关于软件保护的若干忠告

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 尽量自行开发保护机制，不要依赖任何非自行开发的代码。在不影响效率的情况下，可以用虚拟机保护软件（例如VMProtect等）处理需要保护的核心代码
- 不要依赖壳的保护。加密壳都能被解开或脱壳，现在的许多壳转向虚拟机加密方向就是利用了这一特点。如果时间允许且有相应的技术能力，可以设计自己的加壳/压缩方法。如果采用现成的加壳工具，最好不要选择流行的工具。保护强度与流行程度成反比，越是流行的工具，就越可能由于被广泛深入地研究而有了通用的脱壳/解密办法

关于软件保护的若干忠告

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 增加对软件自身完整性的检查，包括对磁盘文件和内存映像的检查，以防止有人未经允许就修改程序，进而达到破解的目的
- 不要采用一目了然的名字来命名函数和文件，例如“IsLicensedVersion”、“key.dat”等。所有与软件加密相关的字符串都不能以明文形式直接存放在可执行文件中，这些字符串最好是动态生成的
- 给用户的提示信息越少越好，因为任何蛛丝马迹都可能导致解密者直接到达加密的核心代码处。例如，发现破解企图后，不要立即向用户发送提示信息，可以在系统的某个地方做一个记号，经过一段随机时间后使软件停止工作，或者使软件“装作”正常工作，但在所处理的数据中加入一些“垃圾”

关于软件保护的若干忠告

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

- 将注册码和安装时间记录在多个地方
- 检查注册信息和时间的代码越分散越好。不要调用同一个函数或判断同一个全局标志。如果这样做，只要修改一个地方，其他的地方就都被破解了
- 不要通过GetLocalTime()和GetSystemTime()这种众所周知的函数来获取系统时间。可以通过读取关键系统文件的修改时间来获取系统时间的相关信息
- 如果有可能，应采用联网检查注册码的方法，而且数据在网上传输时需要加密
- 编程时在软件中嵌入反跟踪代码，以提高安全性

关于软件保护的若干忠告

- 在检查注册信息时插入大量无用的运算以误导解密者，并在查出错误的注册信息后加入延时机制
- 为软件保护增加一定的随机性。例如，除了在启动时检查注册码，还可以在软件运行的某个时刻随机检查注册码。随机值还可以很好地防范那些模拟工具的解密
- 如果采用注册码的保护方式，最好是一机一码，即注册码与机器特征相关。这样，一台机器上的注册码就无法在另外一台机器上使用，从而防止注册码被散播所造成的影响。在机器号的算法上不要太迷信硬盘序列号，因为使用相关工具可以修改其值

关于软件保护的若干忠告

- 如果试用版与正式版是独立的版本，且试用版软件不具有某项功能，则不要只禁用相关菜单，而要彻底删除相关代码，使编译后的程序中根本没有相关的功能代码
- 如果软件中包含驱动程序，则最好将保护判断代码放在驱动程序中。驱动程序在访问系统资源时受到的限制比普通应用程序少得多，这也给软件设计者提供了发挥的余地
- 如果采用keyfile的保护方式，则keyfile的体积不能太小。可将其结构设计得复杂一些，在程序中的不同位置对keyfile的不同部分进行复杂的运算和检查

关于软件保护的若干忠告

- 自己设计的检查注册信息的算法不能过于简单，最好采用比较成熟的密码学算法
- 设计加密方案时应该多从解密的角度考虑，这样才能比较合理地运用各种技术。当然，任何加密方案都无法达到完美的程度，因此，在设计时要考虑其他方面的平衡。加密方案的好坏，用IT界的一句名言来描述或许很合适：一个木桶能装多少水，是由最短的那块木板决定的