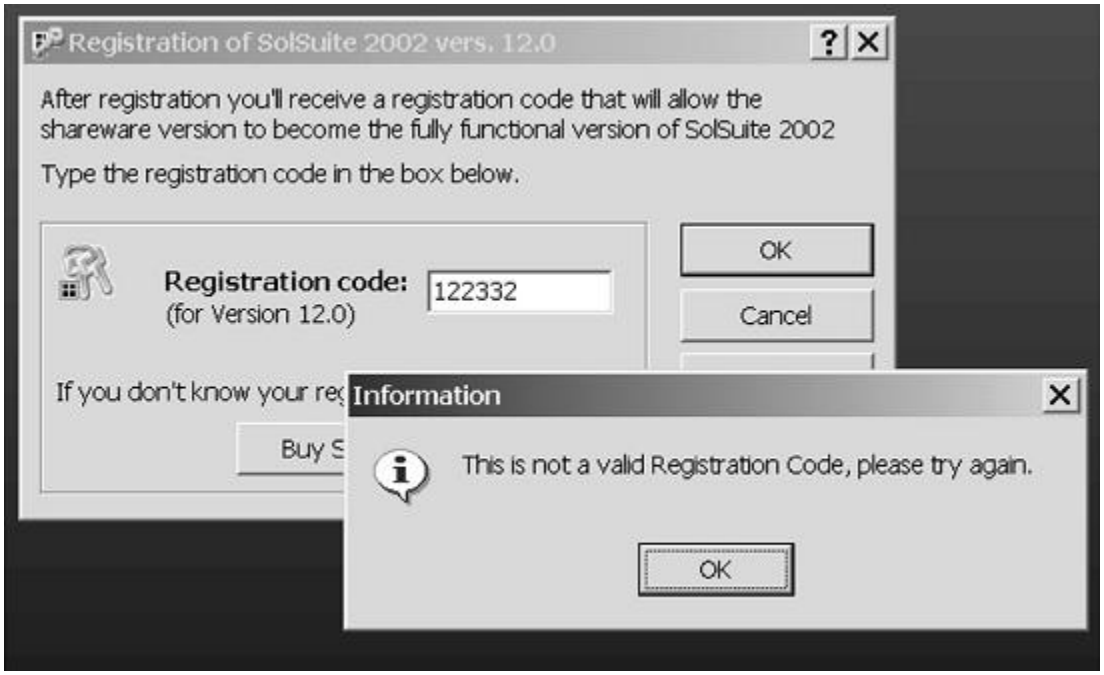
The Basic Types of Software of Protection

1. Registration – Number (Serial – Number) Protection :

Những chương trình sử dụng loại này yêu cầu người sử dụng nhập vào một registration - number để đăng ký. Có nhiều loại registration – number ,bao gồm :
Registraion – number luôn luôn giống nhau.
Registration – number thay đổi dựa vào các thông tin nhập vào như tên công ty,tên người sử dụng v.v…
Registration – number thay đổi dựa vào máy của người dùng.
Registration – number được bảo vệ trong các chương trình được viết bằng Visua Basic hoặc Borland Delphi.
Registration – number được kiểm tra online.

Registration – number Không Đổi :

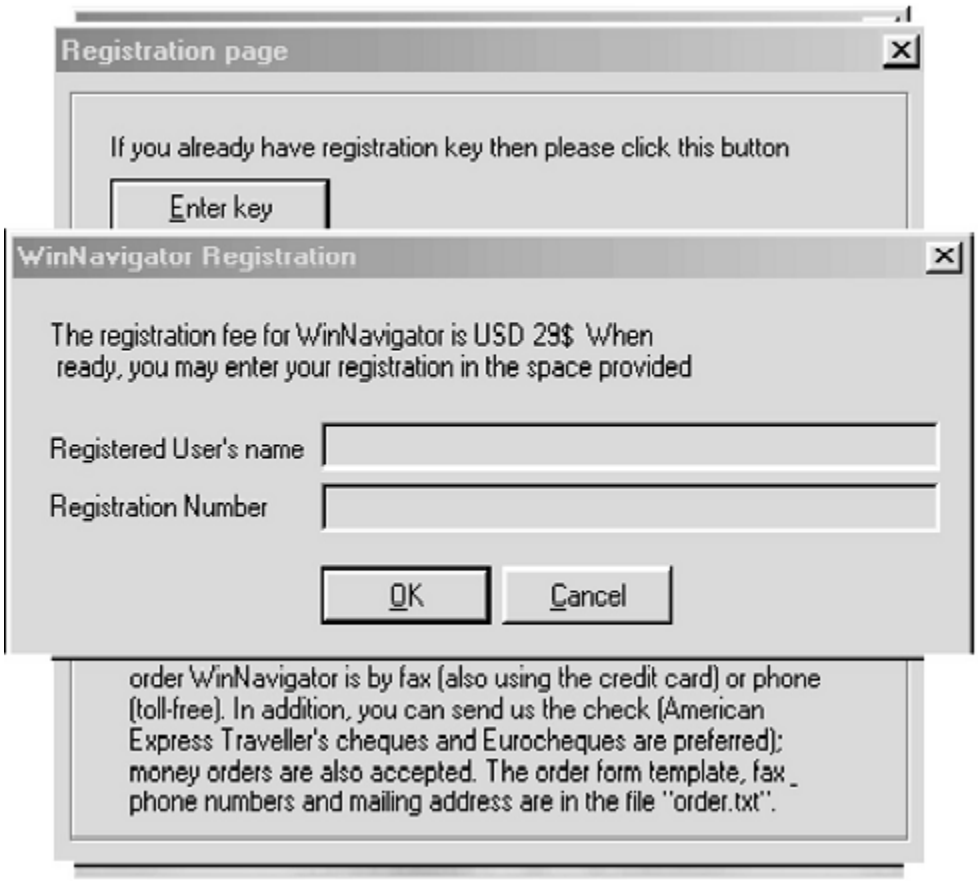
Một chương trình được bảo vệ bằng phương pháp này yêu cầu người dùng nhập vào một số đăng ký như hình bên :



Tiện lợi của phương pháp bảo vệ này so với các kĩ thuật bảo vệ Registration – number khác là correct Registration- number không cần lưu trong bộ nhớ rồi đem so sánh với Registration- number được nhập vào mà cả hai sẽ được XORed hoặc tính toán rồi lấy hai kết quả đó để so sánh với nhau. Dĩ nhiên là các lập trình viên có thể sử dụng các phép tính phức tạp hoặc mã hóa nhiều sections của chương trình để làm cho các crackers khó mà tìm được một Registration-number chính xác.

Registration-number Thay Đổi Dựa Vào Thông Tin Nhập Vào :

Đây là một protection được sử dụng phổ biến. Với loại protection này thì trước khi nhập vào Registration-number bạn phải nhập tên, công ty, hoặc các thông tin khác và một Correct Registration-number sẽ thay đổi dựa vào các thông tin bạn nhập vào



Nếu bạn nhập vào Registration-number mà không đúng với thông tin đã nhập thì sự đăng ký đó sẽ không thành công.

Registration Number Thay Đổi Dựa Vào Máy Của Người Sử Dụng:

Đây là một protection rất khó chịu cho các crackers, bởi vì chương trình được đăng ký ở máy này nhưng khó có thể đăng ký ở máy khác bởi vì Registration-number sẽ thay đổi dựa vào các thông tin của ổ cứng…

Đối với loại protection này thì ta phải tìm ra cho được đoạn code dùng để phát hiện các thông tin của máy tính như ổ cứng… và đoạn code kiểm tra tính hợp lệ của Registratio-number nhập vào.

Registration Number Protection in Visual Basic Programs

VB4

Đối với các chương trình viết bằng VB4 thì một cracker có kinh nghiệm chỉ cần chưa đầy 5 phút là có thể tìm ra được Correct Registration-number bởi vì các chương trình được viết bằng VB4 đều sử dụng cùng một thư viện VB40016.DLL (VB40032.DLL) để so sánh Correct Registration-number với Registration-number nhập vào

Translated by luCiFer

Thậm chí khi một chương trình được viết bằng VB4 dùng phương pháp khác để so sánh với Registration-number được nhập vào thì ta cũng có thể dễ dàng tìm ra Correct Registration-number trong bộ nhớ bởi vì một Correct Registration-number luôn được đặt gần với fake Registration-number,do đó chỉ cần tìm ra fake Registration-number là ta có thể tìm được Correct Registration-number.
Tiện lợi của các chương trình được viết bằng VB4 là khó trace trong SoftIce bởi vì code của nó không được compiled ra mã máy (machine code) mà được compiled thành các mã giả (pseudo-instructions). Các mã giả này chỉ được thực thi khi chương trình bắt đầu chạy.

How VB4 Programs Are Cracked

Đối với các chương trình VB4 16-bit, thì phải tìm trong memory và trong VB40016.DLL đoạn sau : 8BF88EC21EC5760E33C0F3A674051BC01DFFFF rồi đặt breakpoint vào nơi tìm ra đoạn code :

```
Prefix  Instruction
.
.
.
8BF8        mov di,ax
8EC2        mov es,dx
1E          push ds
C5760E      lds si,[bp+0E]
33C0        xor ax,ax
F3A6        repz cmpsb
7405        jz 2667
1BC0        sbb ax,ax
1DFFFF      sbb ax,ffff
.
.
.
```

Sau trace tới repz cmpsb (dùng để so sánh giữa Correct Registration-number và fake Registration-number) rồi nhìn vào địa chỉ es:di và ds:si để tìm ra Correct Registration-number.

Hầu hết các chương trình VB4 32-bit sử dụng hàm MultiByteToWideChar trong thư viện VB40032.DLL để so sánh hai chuỗi.Việc cần làm là đặt breakpoint tại hàm hmemcpy trước khi nhấn OK và trace cho tới khi vào VB40032.DLL rồi tìm trong memory các bytes sau :

56578B7C24108B74240C8B4C2414

Đặt breakpoint vào địa chỉ tìm được sau đó nhấn OK

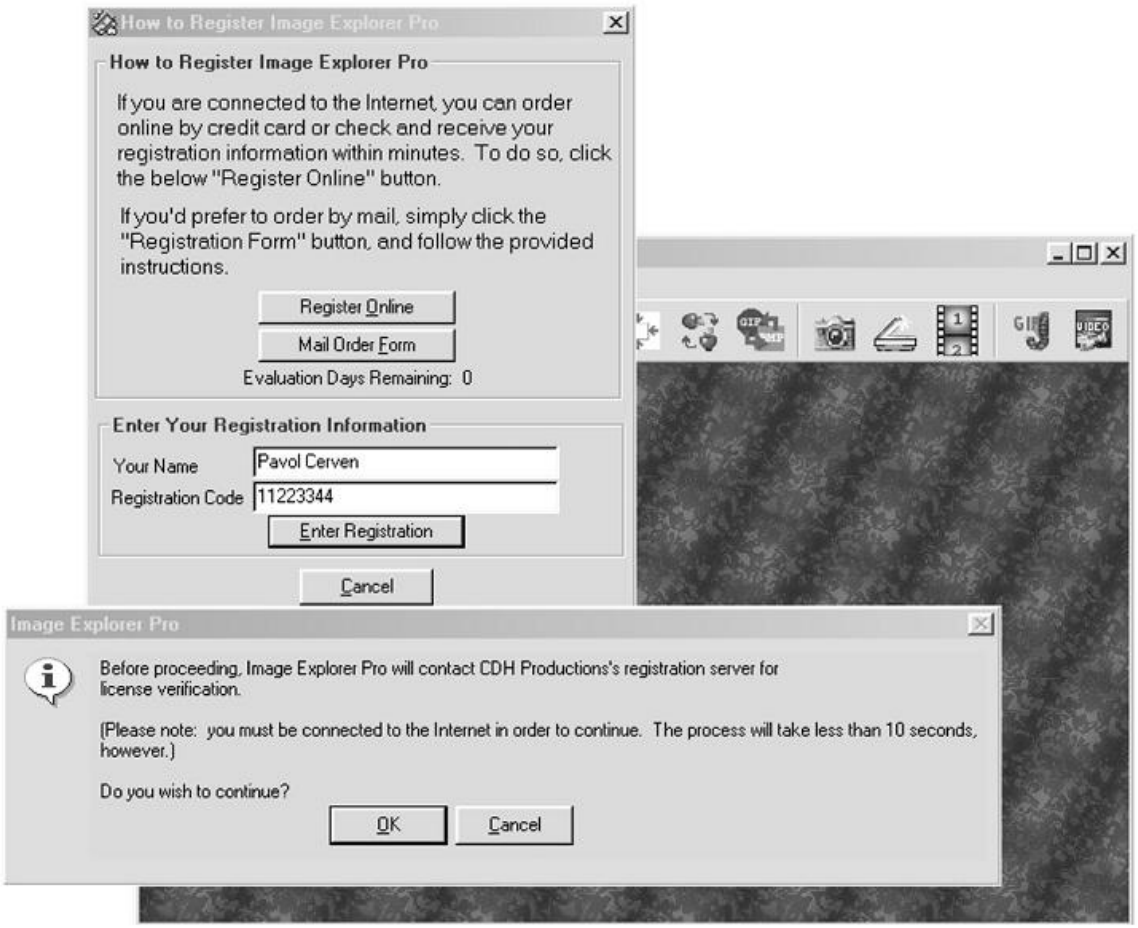
```
Prefix  Instruction
.
.
.
56                  push esi
57                  push edi
8B7C2410            mov edi,[esp+10] ;es:edi→fake Registration-number
8B74240C            mov esi,[esp+0C] ;esi→correct Registration-number
813F70006300        cmp dword ptr [edi],00630070
7527                jne 0F79B381
803E00              cmp byte ptr [esi],00
.
.
.
```

VB5 and Higher,VB5 and Higher compiled in P-code (packed code) :

Chương trình viết bằng VB5 (or P-code) thì khó crack hơn VB4. Các crackers thường không muốn crack các chương trình được viết bằng VB (hoặc Delphi) bởi vì code của nó khó đọc và khó hiểu, hơn nữa lại mất nhiều thời gian để trace. Để crack các chương trình viết bằng VB thì các crackers thường sử dụng các phương pháp phát sinh ra một Registration-number chỉ để cho một người dùng hoặc sửa code để chương trình chấp nhập các Registration-number nhập vào. Chỉ có những crackers xuất sắc mới tạo ra được keygen bởi vì code của nó rất khó hiểu.(và công cụ thường sử dùng là SmartCheck).

Registration Number Được Kiểm Tra Online:

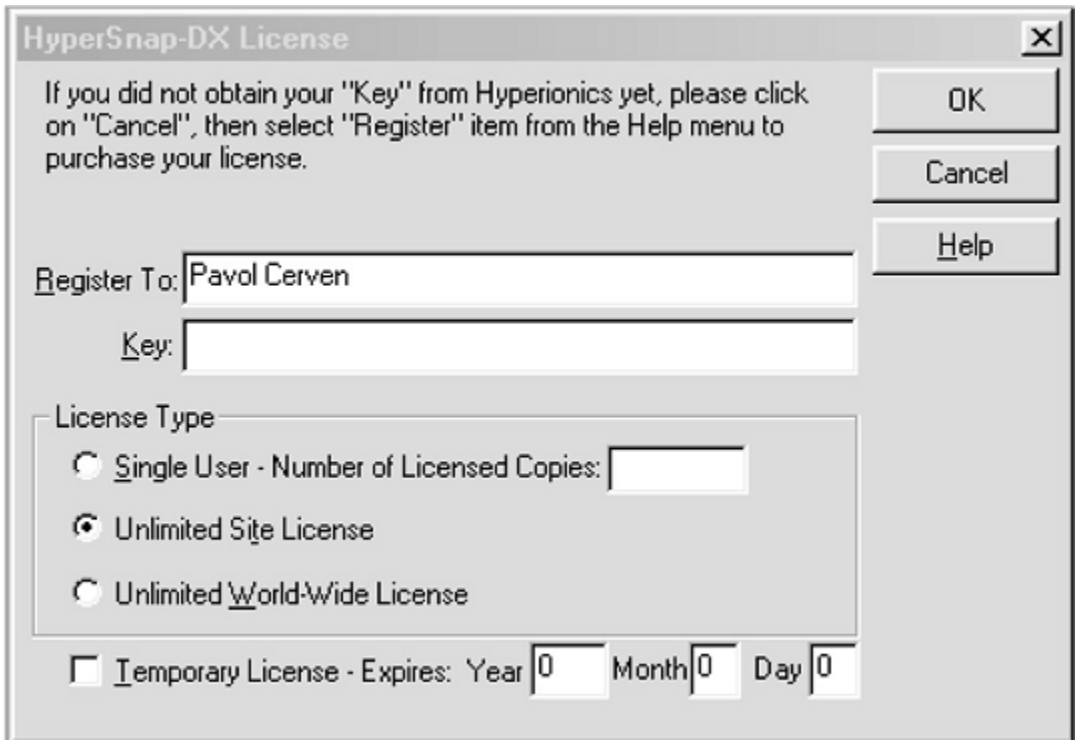
Một vài phần mềm dùng Internet để kiểm tra các Registration-number



Khi mà các Registration-number được nhập vào thì nó sẽ được gửi tới server thông qua Internet. Server sẽ kiểm tra và trả về một giá trị cho biết Registration-number đó có đúng không. Chương trình sẽ xử lý giá trị này và quyết định chương trình có được đăng ký hay không. Phần lớn các protection này rất đơn giản, một cracker có kinh nghiệm có thể loại bỏ chúng một cách dễ dàng.

2. Time-Limited Programs :

Đây là loại chương trình sẽ không cho sử dụng nếu hết hạn dùng thử.
Tuy nhiên protection này không hiệu quả lắm bởi vì một khi giới Time-Litmits mà bị cracked thì ta có thể sử dụng toàn bộ chức năng của chương trình.



Time-Limits được thực thi bằng nhiều cách:

- Time-Limit bị gỡ bỏ khi ta nhập đúng Registration-number.
- Time-Limit bị gỡ bỏ khi tìm thấy Registration file.
- Time-Limit không thể bị gỡ bỏ, người dùng chỉ còn lựa chọn duy nhất là mua bản đầy đủ mà không bị giới hạn thời gian.
- Time-Limit dựa vào số lần mà chương trình đã chạy.Khi đạt tới số lần qui định thì chương trình sẽ không cho sử dụng nữa.

Time-Limit bị gỡ bỏ khi ta nhập đúng Registration-number :

Protection này cũng tương tự như các protection đã trình bày, có cái khác là nếu Correct Registration-number không được nhập vào, chương trình sẽ không được đăng ký và nó sẽ không chạy sau khi hết thời hạn dùng thử.

Time-Limit bị gỡ bỏ khi tìm thấy Registration Key File (.REG) :

Đây là một protection tuyệt vời mà ít được sử dụng. Nó sẽ gửi Registration file thông qua Internet, trong file này có thể chứa một phần lớn mã của chương trình dùng để unblock Time limit. Nhiều chương trình antivirus sử dụng loại protection này

Time-Limit không thể bị gỡ bỏ; người dùng phải mua bản đầy đủ không bị giới hạn thời gian :

Các chương trình Demos thường sử dụng loại protection này,loại này không có nhập vào Registration-code. Do đó khi hết hạn sử dụng chỉ có cách là mua chương trình.

Tim-Limit dựa vào số lần mà ta đã khởi động chương trình :

Loại này cũng giống như các loại Time-Limit khác. Tuy nhiên thay vì kiểm tra số ngày qui định thì nó kiểm tra số lần mà ta đã chạy chương trình.

3. Registration Key File (Key File) Protection :

Protection này sẽ tạo ra một file, thường là nằm cùng thư mục cài đặt. Chương trình sẽ kiểm tra nội dung bên trong file và nếu đúng thì chương trình được đăng kí, nếu sai hoặc tìm không thấy file thì chương trình sẽ không được đăng ký hoặc không chạy. Trong file này có thể chứa thông tin về người dùng hoặc các constants cho việc giải mã các phần bị mã hóa.

Có hai loại Registration file Protection:

- Một vài chức năng của chương trình sẽ bị khóa nếu không key file.
- Chương trình sẽ giới hạn thời gian sử dụng nếu không có key file.

Một vài chức năng của chương trình sẽ bị khóa nếu không có Registration key file.

Đây là một loại protection khá tốt.Tuy nhiên, giống như các loại protection khác nó có thể bị gỡ bỏ. Khi sử dụng loại này thì một số chức năng của chương trình sẽ bị giới hạn nếu không có correct key file. Ngay khi một correct key file được đặt vào thư mục cài đặt thì các chức năng của chương trình sẽ phục hồi.
Có nhiều cách để thực thi loại protection này. Tệ nhất và dễ bị gỡ bỏ nhất là sử dụng một routine để kiểm tra có correct key file hay không, nếu tìm thấy thì các chức năng bị giới hạn sẽ không bị giới hạn nữa. Thường thì các Key file loại này được mã hóa gây không ít khó khăn cho các crackers

Đây là một loại protection tốt bởi vì nó khó bị gỡ bỏ nhưng lại gây không ít khó khăn cho người dùng không có kết nối Internet.
Loại protection này có một điều bất lợi là chỉ cần một người có được correct key file và share cho những người khác thì chương trình sẽ chạy trên tất cả các máy có key file đó.

Chương trình sẽ giới hạn thời gian sử dụng nếu không có Registration key file.

Phần lớn các công ty chuyên về antivirus đều sử dụng loại này. Chương trình sẽ thể hiện như một chương trình Unregistered và Time-Limited nếu không có key file.

4. Hardware-Key (Dongle) Protection

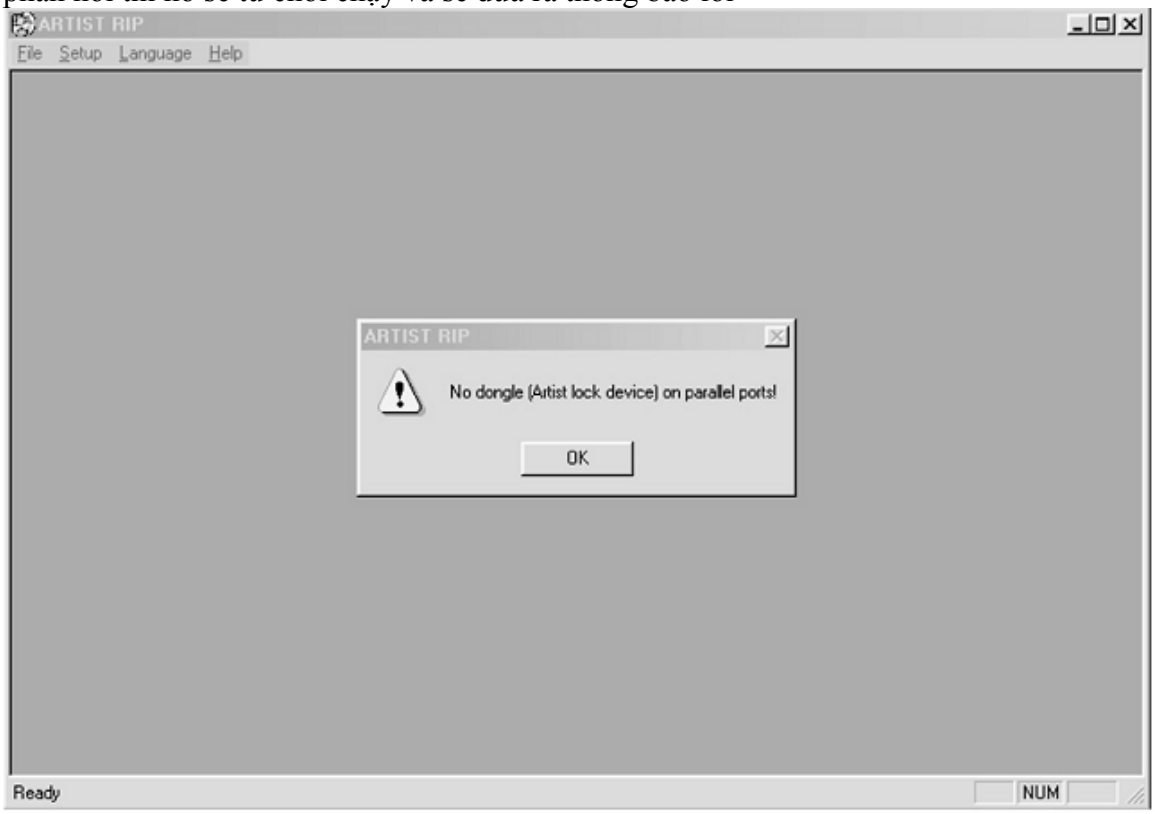
Bảo vệ phần mềm với Hardware-Key là một loại khác với khá đặc biệt bởi vì một copy-protection gọi là *dongle* phải được kết nối với một cổng I/O trên máy tính và phải hiện diện để chương trình chạy.
Có hai loại Hardware-Key Protection :

- Chương trình sẽ không thể chạy nếu không có Hardware-Key.
- Một vài chức năng của chương trình sẽ bị giới hạn nếu không có Hardware-Key.

Translated by luCiFer

Chương trình không thể chạy nếu không có Hardware-Key :

Hầu hết các Hardware-Key rất đơn giản. Chương trình gởi dữ liệu tới cổng nơi mà Hardware-Key được cho là ở đó và nó đợi phản hồi. Nếu chương trình không nhận được phản hồi thì nó sẽ từ chối chạy và sẽ đưa ra thông báo lỗi



Có nhiều cách để tiếp cận nơi kiểm tra sự hiện diện của Hardware-Key và một cách trong số này là dùng API hook. Nó sẽ kiểm tra sự hiện diên của key trong việc gọi tới hàm API. Một vài Hardware-Key cao cấp còn sử dụng chính drivers của nó.

Một cái không hay của việc sử dụng Hardware-Key là bởi vì một *dongle* phải được cung cấp cho mỗi bản sao của chương trình nên giá của nó sẽ tăng lên. Do đó các Hardware-Key chỉ sử dụng cho các chương mắc tiền và trình không phải là shareware.

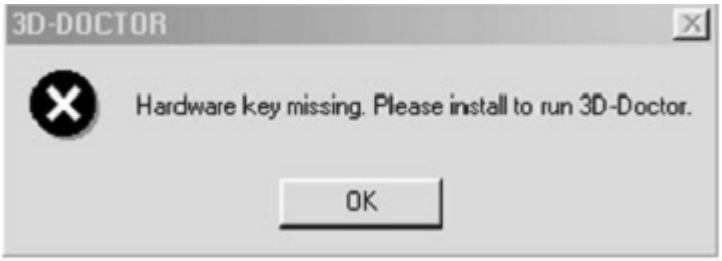
Một vài chức năng của chương trình sẽ bị giới hạn nếu không có Hardware-Key :

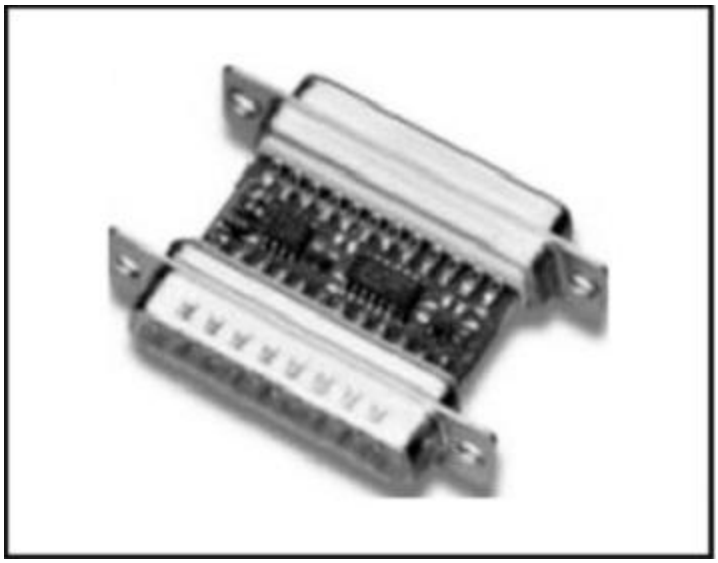
Nguyên lý của loại protection này rất đơn giản – khi không có Hardware-Key nào được kết nối thì một vài chức năng quan trọng của chương trình sẽ không hoạt động

Khi mà một Hardware-Key được kết nối thì các chức năng đó sẽ phục hồi. Khi nhưng chức năng này được đặt trong các Hardware-Key thì đây là một loại protection chắc chắn, ngoài ra key này có thể chứa codes cho việc giải mã các chức năng ngay trong memory. Hầu như không thể gỡ bỏ protection này mà không có key, đặc biệt nó được mã hóa tốt. Tuy nhiên Hardware-Key chỉ được sử dụng để blocked hay được Unblocked thì các crackers sẽ dễ dàng gỡ bỏ protection này thậm chí không dùng tới Hardware-Key.

HASP Hardware-Keys

Dòng HASP của các Hardware-Keys (hình dưới) của Aladdin Knowledge Systems cung cấp nhiều lựa chọn. HASP cài đặt các drivers của nó khi phần mềm được cài đặt và phần mềm này chỉ được sử dụng sau khi giao tiếp với Hardware-Key. HASP có các drivers đặt biệt cho DOS, Windows 9x/2000/NT/XP, và Mac OS X.





Để biết các lệnh Call được thực thi ở đâu trong HASP , nhìn vào lệnh cmp bh,32 :

```
HASPCall :
cmp bh,32                    ;Test to see whether this is a ;higher HASP service
jb Jum
mov esi,dword ptr [ebp+28]
mov eax,dword ptr [esi]

Jum:
mov esi,dword ptr [ebp+20]
mov esi,dword ptr [esi]
push ebp
call Hasp ()                 ;the basic Hasp service is called here
pop ebx
mov edi,dword ptr [ebp+1c]
mov dword ptr [edi] , eax    ;save the return value
mov edi , dword ptr [ebp + 20]
mov dword ptr [edi] , ebx    ;save the return value
mov edi , dword ptr [ebp + 24]
mov dword ptr [edi] , ecx
mov edi , dword ptr [ebp + 28]
mov dword ptr [edi] , edx    ;save the return value
…
```

Một dịch vụ HASP cơ bản luôn có cùng một lệnh Call. Chương trình quyết định dựa vào các tham số mà nó được gọi và khi đó nó quyết định dịch vụ HASP nào sẽ gọi.
Nếu bạn nhìn vào một cấp cao hơn trong mã chương trình thì bạn sẽ thấy đoạn mã như sau :

```
…
push eax
```

```
push ecx
push 000047FE              ;password1
push 000015C9              ;password2
push ebx                   ;lptport
push edi
push 00000003              ;HASP service number 3
mov [esp + 38],0000001A    ;address
call HASPCall;call the ReadWord() service
…
```

Trong trường hợp này, hàm HASP no.3 được gọi : ReadWord(). Tất cả các dịch vụ khác cũng được gọi theo cách này. Sau đây là các hàm quan trọng và thường được sử dụng nhất :

Function No.1 : IsHasp()

Đây là hàm luôn được gọi đầu tiên để kiểm tra xem có Hardware key kèm theo không. Chỉ đơn giản thay đổi giá trị trả về của dịch vụ này sẽ không thể vượt qua HASP.

Input Values :
BH = 01
BL = LTP port

Return Values :
EAX = 0 ; Không tìm thấy Hardware key
EAX = 1 ; Tìm thấy Hardware key

Hàm No.2 : HashCode()

Đây là hàm được gọi ngay sau khi IsHasp() được gọi, password1 và password2 là mật mã mà được sử dụng để giao tiếp với Hardware key. Seed code sẽ định nghĩa các giá trị trả về.

Input Values :
BH = 02
BL = LPT port
EAX = seed code
ECX = password1
EDX = password2

Return Values :
EAX = code1
EBX = code2
ECX = code3
EDX = code4

Function No.3 : ReadWord()

Đây là hàm dùng để đọc dword (a word) từ vùng nhớ của HASP. Địa chỉ nơi mà việc đọc được thực thi sẽ được lưu trong EDI. Để tìm ra địa chỉ bạn phải nhân nó với 2.

Input Values :
BH = 03
BL = LPT port
ECX = password1
EDX = password2
EDI = address

Return Values :
ECX = status 0 – Correct , ngược lại sẽ có lỗi xảy ra (bạn có thể tìm ra mô tả của lỗi này trong các tài liệu HASP)

Function No.4 : WriteWord()
Đây là hàm ghi dword (a word) vào trong memory của HASP. Địa chỉ nơi việc ghi được thực thi sẽ được đặt tại EDI. Để tìm ra địa chỉ, bạn phải nhân nó cho 2 bởi vì nó là một word.

Input Values :
BH = 04
BL = LPT port
ECX = password1
EDX = password2
EDI = address

Return Values :

ECX = status 0 – Correct, ngược lại sẽ có lỗi xảy ra. (bạn có thể tìm ra mô tả của lỗi này trong các tài liệu HASP)

Function No.5 : HaspStatus()
Sử dụng hàm này để thu thập các thông tin của HASP như là : kích thước của vùng nhớ (memory size), kiểu HASP (HASP type), và cổng LPT (LPT port)

Input Values :
BH = 05
BL = LPT port
ECX = password1
EDX = password2

Return Values :
EAX = memory size
EBX = HASP type
ECX = LPT port

Function No.6 : HaspID()
Sử dụng hàm này để nhận biết HASP ID. EAX chứa phần thấp hơn của ID (lower ID), và EBX chứa phần cao hơn (higer ID).

Input Values :
BH = 06
BL = LPT port
ECX = password1
EDX = password2

Return Values :
EAX = ID lower
EBX = ID higher
ECX = status 0 – Correct, ngược lại sẽ có lỗi xảy ra. (bạn có thể tìm ra mô tả của lỗi này trong các tài liệu HASP)

Function No.50 : ReadBlock()
Đây là hàm dùng để đọc một vùng nhớ từ HASP. Địa chỉ nơi mà việc đọc được thực thi được chứa trong EDI. Độ dài của vùng được đọc thì được đặt trong ESI, còn địa chỉ nơi mà dữ liệu được đọc sẽ được lưu ở ES:EAX.
Để nhận ra địa chỉ hiện tại nơi mà việc đọc xảy ra khi nó được thực thi , ta nhân địa chỉ được đặt trong EDI cho 2 (bởi vì dữ liệu được đọc bởi các words)

Input Values :
BH = 50 or 32h
BL = LPT port
ECX = password1
EDX = password2
EDI = start address
ESI = data block length
ES = buffer segment
EAX = buffer offset

Return Values :
ECX = 0 – Correct, ngược lại sẽ có lỗi xảy ra. (bạn có thể tìm ra mô tả của lỗi này trong các tài liệu HASP)

Function No.51 : WriteBlock()
Đây là hàm dùng để ghi một vùng nhớ vào trong HASP. Địa chỉ nơi mà việc ghi được thực thi thì được đặt tại EDI. Chiều dài của vùng được ghi được đặt trong ESI, và địa chỉ nơi mà dữ liệu được ghi sẽ được đặt tại ES:EAX

Để nhận ra địa chỉ hiện tại nơi mà việc ghi xảy ra khi nó được thực thi , ta nhân địa chỉ được đặt trong EDI cho 2 (bởi vì dữ liệu được ghi bởi các words).

Input Values :
BH = 51 or 53
BL = LPT port
ECX = password1
EDX = password2
EDI = start address
ESI = data block strength
ES = buffer segment
EAX = buffer offset

Return Values :
ECX = 0 – Correct, ngược lại sẽ có lỗi xảy ra. (bạn có thể tìm ra mô tả của lỗi này trong các tài liệu HASP).

Thông thường, HASP còn sử dụng các hàm khác bên cạnh các hàm vừa đề cập ở trên, bao gồm các hàm :SetTime(), GetTime(), SetDate(), GetDate(), Writebyte(), Readbyte().

Sentinel Hardware keys

Sentinel cũng tương tự như HASP. Tuy nhiên nó có một điều đặt biệt là nó sẽ kiểm tra gói giao tiếp được khởi tạo chính xác hay chưa trước khi hàm API được gọi.

```
…
cmp word ptr [esi], 7242
is correct
        mov ax, 0002                ;set the error number
        pop edi
        pop esi
        ret 000c
…
```

Nếu giá trị 7242 không được tìm thấy ngay lúc đầu thì sẽ gặp lỗi “Invalid Packet”.

Ở trên thì mình vừa trình bày về các loại Protection cơ bản của phần mềm. Hy vọng các bạn thích và thấy có ích sau khi đọc. Thân