Software Security and Reverse Engineering

What is reverse engineering?

Today the market of software is covered by an incredible number of protected applications, which don't allow you to use all features of programs if you aren't a registered user of these. Reverse engineering is simply the art of removing protection from programs also known as “cracking”.

In Some other words cracking is described as follows: - “When you create a program you engineer it, in fact you build the executable from the source-code. The reverse engineering is simply the art of generate a source-code from an executable. Reverse engineering is used to understand how a program does an action, to bypass protection etc. Usually it's not necessary to disassemble all code of the application not only the part of the application that we are interested must be reversed. Reverse engineering used by a cracker to understand the protection scheme and to break it, so it's a very important thing in the whole world of the crack.”

In short: - "Reverse Engineering referred to a way to modify a program such that it behaves as the way a reverse engineer wish."

“Cracking is a method of making a software program function other than it was Originally intended by means of investigating the code, and, if necessary, patching It.”

A Little bit of history

Reveres egg. Most probably start with the DOS based computer games. The aim is that a player has full life and armed in the final stage of the game. So what a reverse egg. Do is just find the memory location where the life and number of weapons are store and then modify this values. They used memory-cheating tools such as game hack etc. So that they have full life and armed in the last stage of the program. But in today’s world with the advent of the shareware concept more and more software author releases the shareware versions. Hence with this reverse engineering become more tedious, more complex, and trickier.

Today to protect the software a programmer use various kind of technique, some of them are old, bad repetitive techniques but some are new. We will discuss them in next section.

Various Protection schemas

Following are the most commonly used schemas

1) Hard coded serial
2) Serial number, name protection
3) Nag screen
4) Time trial
5) Dongle (hardware protection)
6) Commercial protection

7) Other (cd rom check, keyfiles, disabled function etc.)

Let’s study this in detail

1) **Hard coded serial**: -This is the simple protection as compared to other. In this kind of protection we have to enter only a serial number and this serial number is same for all users. Serial numbers entered are compared to the original serial through an algorithm and if a user entered correct serial then the software registered.

2) **Serial number - name protection**:-In this kind of protection we have to enter a name and a serial number. Then our serial no is compared with the original serial, no which is derived from our name using some algorithm. This protection is some time easy and some times hard, based on the algorithm a programmer use. Example of this type protection is most widely used software "WinZip."

3) **Nag screen** :-In this kind of protection a screen come each time a user start the application, to remained such that how many days are left or your software are unregistered or any other message. This is a littlie hard to remove. And most of the newcomers found it difficult as a new programmer to understand pointers (i.e. –WinZip).

But if a reverse has enough knowledge of windows API then he can easily remove the nag screen.

4) **Time Trial**: - According to +ORC This kind of protection has any of following protection or combination of following protection schema: -

a) To a predetermined amount of days, say 30 days, starting with the first day of installation. This is referred as "CINDERELLA protection".
b) To a predetermined period of time (ending at a specific fixed date) independently from the start date... 'BEST_BEFORE a given date' protection.
c) To a predetermined amount of minutes and/or seconds each time you fire them... 'COUNTDOWN' TIME PROTECTIONS' example of this kind of programs are some games and audio video player which allows an unregistered user to play game for some amount of time say 5 minutes etc.
d) To a predetermined amount of 'times' you use them, say 30 times. Strictly speaking these protections are not 'time' dependent. But they depend only on thing "HOW MANY TIMES YOU EXICUTE THEM"

5) **Dongle Protection**: - this kind of protection is supposed to be toughest protection to crack. This protection is consisting of an EPROM, which was connected with a port on computer. The program which is protected by this is first cheeks the presence of this and then cheeks that the program is registered or not all though it implementation is too hard and hence this kind of protection is not very widely used. This is used in Big Protected shareware’s. This protection is used by a I/O LPT port (hardware) You will need the registration Card attached To your PC's parallel port Or other in order to make The program fully work, otherwise it will be Expired after xxDays / xxUses /rippled or it won’t work at all. Dongles such as: HASP / Sentinel are most commonly used. Dongles uses DLLs/VxD to check the "is registered"

Dongle API is also used for some checks.

Example of programs, which uses this kind of protection, included some version of CAD etc.

6) **Commercial protection**: - Most of the software programmer don't want to spend there precious time in deciding which kind of protection they used to protect there software. Because they think that instead of the spending there time on designing the security algorithm of there programs, why not they spend time to improving the functionality of there program??? And here comes the concept of commercial protection. Today some software company's designs only security algorithm for various software. Also they provide general software, which converts fully functional software in to unregistered version and after paying the registration.

This software gets converted back in to the fully functional registered software after entering the registration details. some of the companies which uses commercial protection for there software are macromedia, Symantec etc and some companies which provides this type of protection are preview systems (vbox protection) etc..

Although this kind of protection has high security because they are professionally designed but they also have some disadvantages. One major disadvantage is that "if a person cracks only one program which is protected using this protection, then he has cracked the entire program which uses this kind of protection!!!!".

For example if a cracker has cracked the flash mx (which is protected by vbox) then he was able to crack easily all the macromedia software such as dream waver mx etc., because all these programs are based on only one kind of protection!

And in the real world there is no protection, which is still uncracked.

7) **Other protections**: - There are many other techniques which are used to protect software. These are generally used in computer games. Such as cd rom protection, disabled function etc. I think most of computer user are familiar with this protection and already seen this kind of protection. For example: - If a user doesn't have cd for a particular game then he cannot be able to play the game directly from hard disk. Because when one runs the program then the program checks for the cdrom.

Also some protection schemes have disabled functions such as you cannot save your work or you cannot use any particular function etc.

So I hope now you understand all the protection schemas, which used to protect software. Ok let's study how reverse engineering is done. The first thing to keep in mind that cracker always works with the disassembly and they are familiar with the windows API.

Now all of us computer user knows that computer only understands binary nothing else. So first we create a program and then compile it now what compiler does is check for syntax, any error and then he generate the .obj file. As in high level language some function are prewritten which are stored in library file hence after this we used linker which links the programs with the library file and then after linking we get an exe file hence exe file we use is nothing but the collection of instruction in binary formats.

Now to reverse engineer there are different tools available.

TOOLS OF THE TRADE

The popularity of Windows and the ease of creating programs for this platform have lead to the development of thousands of shareware programs. Crackers usually work with the assembly code, reverse engineering it, and have an excellent grasp of the Windows APIs as well.

There is no one particular method to crack a program. Depending upon the program and the kind of protection it has, crackers employ different techniques to get into the program. But there are some common tools that crackers employ to start cracking the program. These programs are perfectly legal and useful by themselves.
They are: -

1) Debugger
2) Dissembler
3) Hex-editor
4) Unpacker
5) File Analyzers
6) Registry monitor
7) File monitor

This is the tools, which a cracker used to reverse engineer any software. Let we have take a detail look on them.

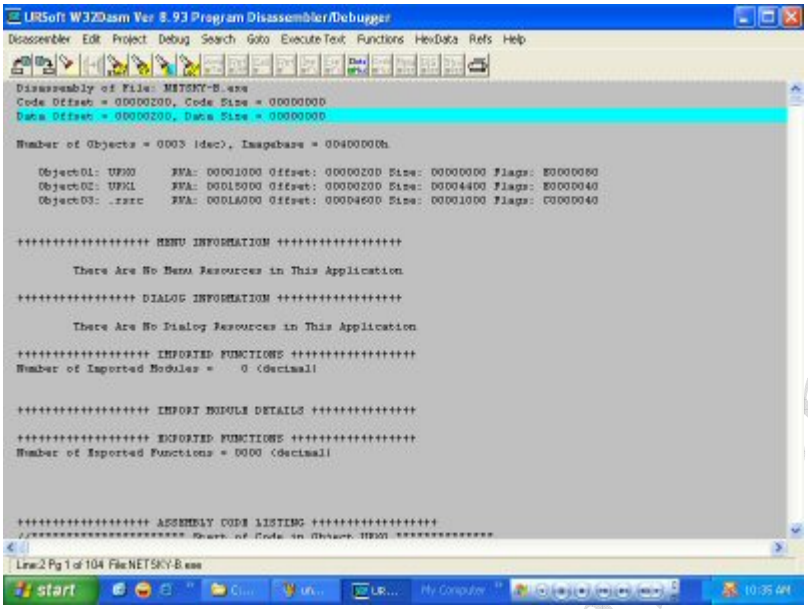
1) **Debugger**: -all of us know that debugger is a utility to debug the program. A programmer use debugger to find bugs in their program. Debugger is only tool by which we can trace/break a function or code live. There are many debuggers available in the market. We all know how to debug any program, first we put a breakpoint on the required statement and then we run the program. When this instruction is near to be executed the program stops and we can see values! This thing is directly related with cracking. Generally software programmer uses windows API function to get the serial number or to create nag screen or dialog boxes. Now if a debugger support breakpoint on execution of ape then a cracker easily set a breakpoint on API such as "getwindowtexta" and then after tracing only some lines of code he can easily find the algorithm to used the generate key and the key itself!!!

There are many debuggers available in the market but one of the most popular and a powerful debugger is SOFTICE from NUMEGA CORPORATION. This debugger is so powerful that earlier version of this debugger used to crack himself!!!! Almost all the cracker in this world is using this debugger. So after seeing its misuse Numega Corporation has kept some restriction on the sale of this great debugger and a buyer must show that he will not use this debugger for illegal activities. But cracked copy of this debugger is freely available on the net. This is a system level debugger, which works directly between a computer's hardware and windows. We cannot load this debugger within windows. We must load this debugger before windows loads in to the memory. It can monitor every process, threads silently in memory until we call it up using hotkeys. It allow us to patch memory at runtime (not permanently and hence we have to use hex editor.) viewing the contents of the register, contains at memory address etc.

2) **Disassembler**: - As an executable file is in binary format so a normal user cannot understand the instruction in this file. Also any exe or executable is generally in PE format (which is a standard format for exe file, decided by the committee of software companies like MICROSOFT, IBM, and AT&T. For more about exe search any virus related site or /simply search your favorite search engines.) Hence a cracker first disassemble the program .now a Disassembler converts the binary file in its equitant assembly language instruction's most of program is written in high level language hence size of the disassembly goes in millions (or even larger) of lines and hence it is not possible for any cracker to understand this code. And hence cracker generally looking for strings in this disassembly such as; -"your 30 day trial period has expired." Or "the serial no you entered is not valid!!!" Etc.

Then they trace the assembly code some lines and simply reverse the jumps. (For example one to jump) so that control did not come on this string and go to the statement such as "thanks for registration!!!"(We will see later how this can be done but currently this info is enough for you..)

Now there are many dissembler available. But two of them, which are most commonly used, are WIN32DASM and IDA .IDA is a powerful debugger then WIN32DASM and used for advanced cracking. But WIN32DASM is most widely used debugger by newcomer and intermediate crackers. This debugger allows you to disassemble any file which is in PE format, we can save disassembly .it can tell us which function is imported, which function is exported, we can execute jump, call, find string data reference and dialog reference easily and many more facilities it provides like we can executes the exe file, step in to it, step over and blah, blah.

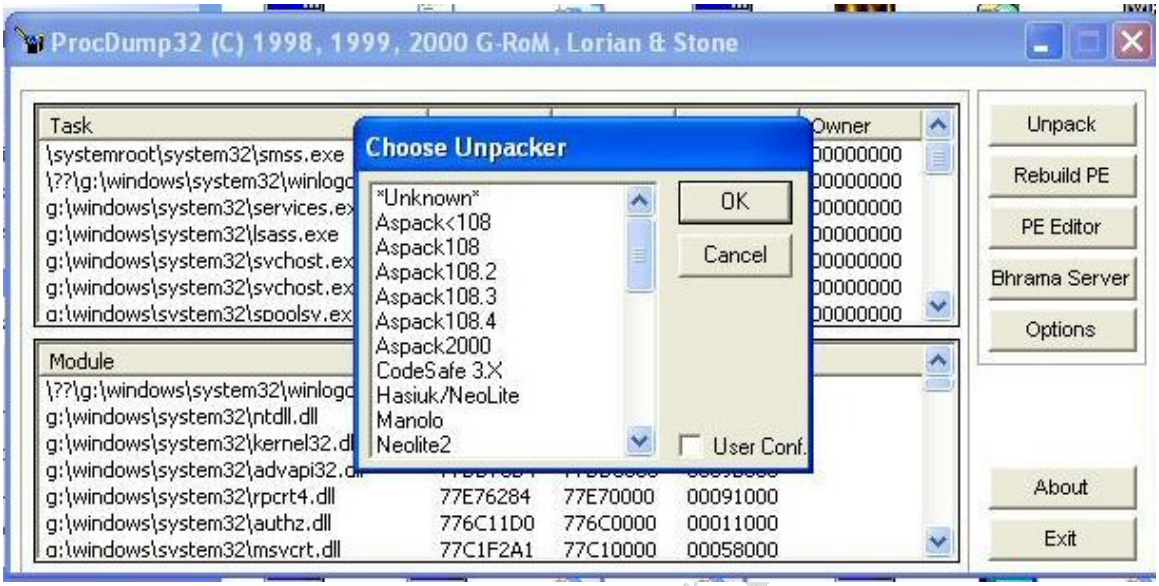


3) **Hex Editor**: -as I mention above that softice can change the value at memory location only at the run time. Now this is not useful or not a good cracking if we have to change the value each time we run the program. Therefore we use hex editors. A hex editor allows us to change the contents of any file in hex format. It displays the contents of the file in hex format. We can simply have to change the value at memory location which we find using softice. Now there are a lot of hex editor available such as ultredit, biew, hiew and a lot (I think many c, c++ programmers has developed it).

But the most popular among these is HIEW. Which stands for “Hacker's vIEW". This little program offers a lot of facilities such as editing in hex or ASCII format, searching any string in hex or ASCII format. There is another good facility which makes it different from others is that, it offers you to write the assembly code and it can automatically convert this code in to equitant hex format. This is helpful for the crackers who don’t know equitant hex value of assembly instruction. (For example: - if we have to change the jump to nope at any memory location then after pressing F7 key then we can only write nope and it will automatically convert it to its hexequilant which is 90.) There are other hex editors also but it is the most widely used.

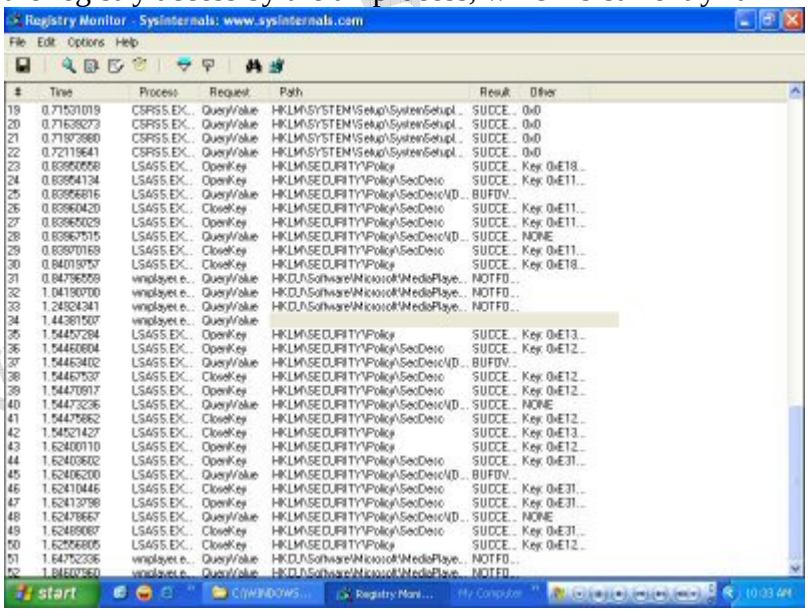
4) **Unpacker/PE Editor**: - sometimes programmers used file compressor such as UPX, ASPACK to minimize the size of the program. This is called a file packer. Now what a packer do is using any algorithm he reduce he size of the file and append it code in to the exe file and at run time, first the code of the unpacker is executed and after that it decompress or unpack the program in memory. Since the program we have to crack is unpacked in the memory only hence a cracker cannot simply disassembles and patch the program. User can only patch it runtime. Therefore to un-pack the exe file permanently we use unpacked. Which unpack the exe file and we can store this unpack file to the disk. If a program is using a packer then its exe header will changed. There are various techniques available to manually unpack the exe by modifying the exe header but those are high level techniques and don’t want to discuss them here because I think most of the

newsiest find difficult to understand it. The most widely used unpacker is procdump. This software has ability to unpack different kind of packer stand-alone. It also allows changing or viewing the header of exe files.

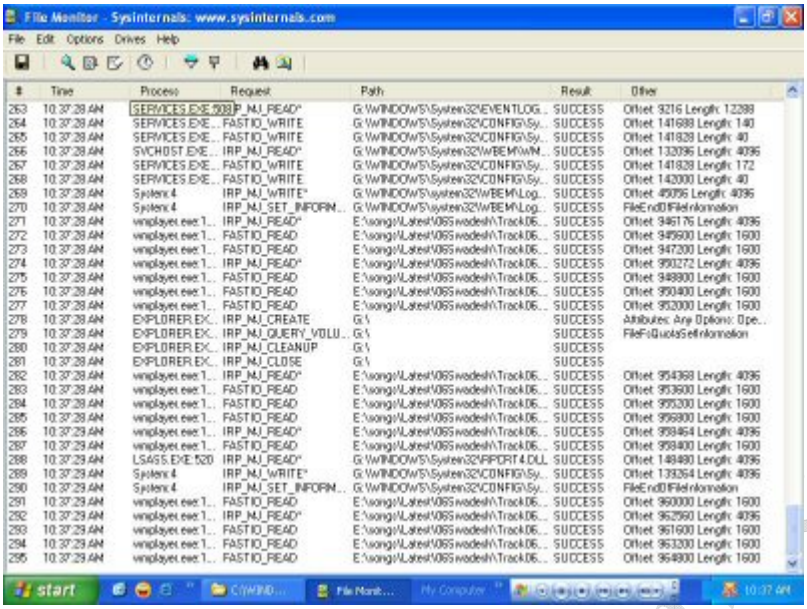


5) **File Analyzers**: - To identify which packer is used to pack file cracker uses this kind of programs. By using this, a cracker can know which compiler or packer is used to protect the shareware. This software simply works on signature byte. With the help of this you can find what compiler or in which language the program has written. There are many this kind of program are available such as file inspector, File Info etc.

6) **Registry monitor**: -Some program uses registry keys to store their registration information. Hence, 'Registry Monitor' is a software which works in background and traps all the registry access by the all process, which is currently running.



7) **File monitor**: -some program also uses key file or they have there security algorithm in different file and hence file monitor is use to see which application is using what file.



Bypassing the protection

How programs are Reverse engineered

In my pervious article I discussed about the different protection schema and tools used for cracking. In this article I show u how cracker past all these protections.

There are different ways to crack. These approaches are determinate from different knowledge, different type of cracker, different personal preferences.
An example can be more useful than thousand of words. There are three type of approaches in cracking shareware programs that need serial number to register or have nag screens. They are

1) **Serial fishing**
2) **Bypassing of the check also called patching**
3) **To make a key generator.**

The first method is simplest and fastest and can be used by normal cracker. The last one is more complex. In fact, you need to understand all the serial number check routine and then u have to code a program based on this which generates the key according to input. The advantage of this method is that the serial number can be used for further versions of the program or for different computers or for different user. So, the choice is determined from the level of knowledge, the time the cracker has and his style of cracking.

Let's have a detail look on them.

1) Serial fishing: -serial fishing is supposed to be the cleanest method to crack any program. This method is also known as live cracking because in this we find the correct serial only at run time. It means we find the serial when program is executing. Serial fishing deals with finding the correct serial and then registering software using this serial

number. In serial fishing we don't have to modify the code but simply we have to inspect or analyze the code.

In serial fishing first we enter any fake i.e. wrong serial number of our choice say 123456 .now this serial number is compared with the correct serial and hence we have to only find the memory location or register where our correct serial number is stored. The general routine in high-level language to compare the serial number is as follows: -

If (entered serial=correct serial) then
Register program (do some modification in program or store the registration information)
Message box ("successfully registered")
Else
Message box ("sorry!! Your serial number is not valid")

And in assembly the general routine is as follows: -

In assembly all the data is stored in registers or stored in any memory location. Suppose eax register store the fake serial and ebx stores the correct serial.

Now the routine is: -
100aa : Cmp eax,ebx
100bb : Jz 100xx ←jump if our serial is correct
100cc : Mov ax,yyyy
100dd: other code….
100xx : code for message box successfully registered
100yy : code…

Where 100xx is memory locations.
Here what is happening that both serial numbers is compared using the cmp instruction and if the two serials are equal then control jumps to the message that we have entered the correct serial. Otherwise controls transfer to next statement, which is 100% sure like this "you have entered a invalid serial"

Although this is not necessary that always cmp is used. But mostly it is used to compare the serial. Now the programmer uses windows APIs such as GetWindowTextA or GetDlgItemTextA to get the serial numbers.

Now as I mentioned SOFTICE allows us to set or put a breakpoint on windows API. Hence a cracker simply puts the breakpoint on such API and when after entering the serial number program breaks on this breakpoint then a cracker simply trace the disassembled code to find the correct serial. Crackers while tracing is simply search the conditional jump such as jne or jz or jae after a cmp instruction. In short they checks the routine I mentioned above and in this way a cracker can find the whole algorithm and correct key with the simple softice command such as: -
D eax
Or

? eax
Well here D eax simply display contains of the register eax in hex format. And? eax display contains of eax in ASCII format. (All these are softice commands).
And after finding correct key he can easily register the software and if he want to distribute the key for every user then he simply creates a keygenrator after analyzing the whole algorithm. Because we know that in serial number-name protection for each name there will be a different key. Some program also uses various techniques such as appending ur hard drive serial number to the end of serial and etc in this case serial number is different for each computer and hence a cracker simply writes the key generator after analyzing the whole protection schema.
By using this technique a cracker can easily defeat the first two protections I mentioned in my previous article (hard coded and name-serial number combination.)

2) Patching: - if a program is showing the nag screen and don't have any option to register then we use patching. Patching is also referred as dead cracking. Using patching is not supposed to be a good crack. most crackers avoid to use this technique until they don't have other option then this. In case of nag screen programmer simply uses the windows API such as DialogBoxParam or MessageBoxa etc. now a cracker sets the breakpoint on these API calls and run the program. Now when the program calls this function then softice pauses the execution of program and a cracker have to deal with the assembly snippets. The simple structure of calling a nag screen in high-level language is as bellow: -

```
If (program is not registered) then
   Display the nag screen
Else
   Execute the program
```

And in assembly the structure is as follows: -
Suppose that first program checks for the registration and return the value in eax register. (If eax=1 then register and eax=0 mean unregistered)
Now this compared as

```
dddd: Cmp eax,1
aaaa:  Jz xxxx
bbbb: Mov ax,02
cccc:  Call yyyy          ←this is for calling the nag screen
Xxxx: Rest of the program…
```

Here aaaa ,bbbb etc are called offset or memory locations.
So whats happening here is that first program checks that if it is registered .the registration status of program is put in to eax. Now this eax is compared with 1 if eax is one then program is registered and we don't have to show the nag screen else we have to show the nag screen.

So we have to only reverse the jump (jz to jnz). So that the nag screen does not appear. In this case we use hex editor such as hiew to patch the exe file of programs.

Patching is also used to remove the time trial protections. Suppose we have a program, which expires after 30 executions. Now it is clear that when we run the program it compares that is 30 executions are over or not. If not then it increases the number of total execution by 1 and store this value somewhere but if 30 executions are over then it shows the message that ur program has expired.

The structure is same as the nag screen: -

aaaa : cmp eax,1e ← (1E in hex=30 in decimal)
bbbb : jea xxxx ←jump if greater then or equal to 30
cccc : ax,02
dddd: call yyyy ←this is for calling the nag screen
eeee: ret ←stop execution and exit
Xxxx: Rest of the program…

Here what's the program is doing is that it comparing the number of times we use with 30 if it is equal or above then it display the message and exit. so what we do here is simply change the jea to jmp. so that program always jump irrespective of that if it is registered or not.

3)Key generator:-this technique is supposed a little harder. in this technique a cracker need to understand all the serial number check routine and understand all the conditions. such as a serial number can contain '–' symbol or size of serial number must be 11 character long or user name must not be blank etc. this techniques simply needs that a cracker must understand the assembly language very well and analyzes the code very carefully. he must be careful to analyze each line of code. because a small mistake in understanding the code can result in unexpected results.

Now lets see how crackers past the commercial protection.
Well today many of the commercial protection are using different techniques to fool the tools of cracking such as anti dissembler code. Anti softice tricks and etc. hence this protections are harder to crack for a newcomer. First lets see how this program protects the software: -

There are common dll or say binary file for all the software which uses a particular commercial protection such as the entire macromedia product uses the same protection 'vbox' and all the files related with vbox is stored in the c:\programfiles\comman\vbox directory. Now when a user runs the program then first the vbox files are executed. Which check that if program is registered or not. If program is not registered then it checks the 30 days trial period and if trial not expired then executes the program. Commercial protection included many checks so a cracker cannot easily patch the program. The most popular trend in between the cracker is that they simply BYPASS this kind of protection. it means as I mentioned that the vbox changed the header of exe file and for this reason all files related with vbox is executed before the actual exe file of program is executed. Now what a cracker does is simply find the original entry point of the exe. It means a cracker only have to find that from which point the original program

starts its execution. For this a cracker puts breakpoint on windows API such as GetProcAddress etc and then run the program. Now when program executed then first vbox code is executed and therefore vbox calls the API GetProcAddress and SOFTICE pauses the execution of program. Now a cracker have the assembly snippets. The rest is purely depends on a crackers ability and experience. After tracing some lines from the vbox files a cracker can find the original program entry point.

After finding the entry point a cracker simply modify the exe header and IAT.
So now onwards program has nothing to deal with commercial protection because cracker has bypassed the protection!!!!

For each commercial protection there is a different way to crack. The method I discussed here is only related with vbox protection.

So this are all the techniques generally used in cracking world. Nowadays there are several cracking groups specialized in reverse web scripts. There is nothing of new in this because the web pages are written in java or CGI scripts or something else. So, they can be considered as small programs. Consequently, this is only another type of crack.

The web cracker usually reverses the protection schemes of web pages creating cracked passwords, which are distributed on the web.
To end this article I would like to mention these lines of a cracker:-
“There is a crack, a crack in every thing. That is how the light gets in.”

Hope it tells the psychology of a cracker.

-Hardik Shah