Crashtest's tutorial #1.

Second version

First Version : 09/08/1998. 490 Lines.
Second Version : 09/10/1998. +750 lines

In the next version : the REPZ instructions...

0) Introduction

I don't want to write another tut, in which you see somebody taking a prog on the internet
like winzip, puts a breakpoint, tells you : yeah , the protection is here, change that jnz
to jz, and you cracked it.

when ppl ask for tutorials, I usually send them to http://www.fravia.org/
but some came back and said : I just follow the tut, I press Enter when it says 'press enter'
but I don't see how I could do it myself.
So IF YOU'RE ONE OF THEM, THIS TUTORIAL IS FOR YOU !!!!
Even if this tutorial gets longer, read it !

this tut is not supposed to be perfect (as I'm not perfect myself), and it's not exclusive
things. ASM is the same for everybody.

So let's start from scratch:

First thing about cracking : take an ASM book. not a big one.
you don't have to know have to make cosinus calculus in asm to start cracking.
so buy a book, not ASM for dummies, but everybody has started with a small ASM book in his
drawer.

Second thing : get winice work on your computer. get it anywhere you want.
ask on #cracking4newbies, for example.
Install it. restart your computer.
if you don't know what changed, press CTRL+D. you'll swap to winice, who is running
'behind' windows. press CTRL+D to swap back to the buggy OS.

you should get W32dasm89, too, but using it too quickly can bring you to be just a lamer
cracker, who's looking for schemes, and just does 1 bytes patching on conditional jump.
you think more when starting with winice. (that's how I started btw, with just one tut)

So first things about necessary knowledge (knowledge is power !)

1) Basic things about ASM:

1.1) The registers:

ax, bx, cx, dx, ah, al, eax... what's all this stuff

a register is just a "special number" stored in the processor.
that can be used for anything :
storing a number : al=ffh ==> 255 in decimal.
storing a char :   al=41h ==> the code 41h=65 is the code of 'A'
storing an adress in memory.

etc...

I don't want to explain you what is a bit. read your book for that.

UNDERSTAND that the processor just take a register as a number.
if it means a char, or an adress, the processor just takes it indifferently.
the instructions makes the difference.

we use decimal, the processor hexa, that's all. there is just a difference for the
programmer/cracker. the processor takes it has it comes.

eax is a 32bits register (a 'doubleword', or 'double')
the lower part of eax is ax, a 16 bits register (a 'word)
ax can be decomposed in his higher and lower part, ah and al. (that are 8 bits register)
(ah and al are 'bytes'.)

so have a look
for example :

eax=654f65de
here higher part of eax=645f
you can't access to it directly
you have to do a division, for example, to access it easily
the lower part of eax is ax : 65de (=26078)
similarily, ah=65, al = de

they are not independant. modifying eax modifies al, ah, etc... and
modifying al changes ax and eax.
what's interesting in that ? well, you can be intersted in using the whole register
to make faster operation with huge numbers, and when you just need a boolean test (yes/no)
only al is needed.

NOTE : this his the consequences of the evolution of the PC.
on 8086 to 80286, eax didn't exist.

2nd note : what is said here about ax is the same for bx.
for basic operations (like moving, comparison), their role is similar.
but don't think the role of ds or eip is the same

all the registers are :
32 bits : eax, ebx, ecx, edx, eip, edi, esi, ebp, esp.
16 bits : cs, ds, ss, es, fs ,gs
the flag register. (no name)
we'll see the flag register later.

important register for starting cracking :
most : eax, ebx, ecx, edx, flag register,
less : edi, esi
even less : ebp, esp, cs, ds, ss, es, eip
not at all : fs, gs.

don't forget you shouldn't modify any of this register when in winice. especially cs,ds...
except if you sure you found the protection

1.2) the operations (or commands) in ASM

1.2.1) the MOV : move instruction

```
before  : eax=1304EF
MOV EAX,0101FF
after   : eax=0101FF
```

the basic operation
the type is MOV TO, FROM
the FROM isn't modified of course.

Main use :
MOV [a register], [a number]
 Like shown before : mov EDX,0123DF. the number must be of the same type of the register
 if you see a mov EDX,01, in fact it's a mov EDX,0000001
MOV [a register], [a register]
 mov EAX,ECX. no problem. as before, you can have a mov EAX,CL or mov DH,CX

More difficult :
MOV [a register], [[a register]]
 mov eax, [ecx] : it means : the processor looks at the adress stored in ecx, goes to
 this adress, and read the adress as a number, that will be stored in eax

and the opposite :
MOV [[a register]], [a register]/[a number]
 the processor puts the register contents/the number at the address written in the
 first register.

Note : you'll understand all this before when you're in winice, with the register window
 disabled: every time you'll execute such an instruction, winice will automatically show you
 the adress pointed by the register between brackets.

DEEPER EXPLANATION :
before :
EDX=0345FD
EAX=1039456
at the adress in memory DS:EDX=DS:0345FD, you'll see 0304569 if you read it as a intel number
(the DS and the 'intel number thing' will be explained later)

so if you do a
mov EAX,[EDX]

after you'll have
EAX=0304569.
get it ? (I hope so...)

Note : you can see a mov EAX, [EDX+10]. it means
 take the adress shown in the number equal to (EDX+10).

You must understand that : EDX,ECX,etc... are 32 bits number
it means they can go from 0 to FFFFFFFF. (4294967295)

1.2.2) the CMP operation

the operation of comparison which is linked to the jumps comparison (see later)

CMP [a register], [a number],CMP [a register], [a register],CMP [a register], [[a register]]
it's easily understandable if you understood all the MOV operation.
note that none of the parameters (first or second are modified)
as before, cmp eax,ch is impossible. the same type is required on both 'side'.

So we'll start directly with the next operation to understand better the use of the CMP

1.2.3) the jumps operation

1.2.3.1) the basic jump : JMP

```
JMP [a register]
JMP [a number]
```

the next operation that will be executed is at the adress CS:[register] or CS:[Number].
(like 'DS:', 'CS:' will be explained (perhaps) later)

1.2.3.2) the conditional jumps : JNZ,JZ,JG,...

This are the most important for cracking : the test if your serial is good or not, for example.
do you understand ?

well: the scheme is like this
 a comparison
 the 'result' is stored in the FLAG REGISTER
 A conditional jump

First : the flag register is a register without name.
 all of his bits are taken separately. they all have a name. you can see them in winice under
 ESI=.
 o d i s z a p c you can count : 8 bits. it's a 8 bit register.
 if one of them is in uppercase (Z, not z), it means the flag is on.

So what about the flag register.
Imagine a simple protection : just one serial. it must be 123456 (01E240 in hex)
EAX contains the number you entered.
there will be a comparison with 01E240, like this

```
CMP EAX,01E240
JNZ xxxxx
```

understand : the computer execute the cmp operation. flags will be changed. especially ZF/Z
 the Zero Flag.
in fact, the processor will do a substract. if the result is zero (it means EAX=01240), the zero
flag will be on. if not, it will be changed to off.

JNZ means Jump if Not Zero : jump if the Z flag is off
JZ means Jump if Zero : jump if the Z flag is off.

but you'll don't have to think about the flag : just look at the value in eax, and
JNZ will actually jump if eax is different from 01E240.
(I hope my explanations are enough)

perhaps don't think too much about the flag. you can see Winice tell you if the jump will be
really effective. swithc Z to z to see the change (there are CMP ... JNZ everywhere).

other important jumps :
JG,JGE ... : Jump if greater, if greater or equal... (easy to understand. just read your book)

1.2.4) the call operation

used for calling a set of operation, then come back.
basically, the next operation will be at the address pointed by the operand
(in front of the call), and it will come back with a RET.

example :

```
address  operations   steps
001 :    MOV AX,10      1
002 :    CALL 004       2
003 :    JMP 006        5
004 :    MOV AX,9       3
005 :    RET            4
006 :    MOV BX,AX      6
```

it's a strange structure, just to explain you its interest.

1.2.5) other operations

ADD, MULT, etc...
I talked about the basic instructions, go on with your book.
take w32dasm, disasm any small Window EXE, and look at the instructions you should understand.
for example : ADD, INC, DEC, MULT, REPZ,REPNZ,TEST,push, pop,jumps,
and logical operation : AND, OR,XOR, NOT
I think I'll explain push and pop anyway later.

End of this first part : don't forget to read your book, and/or other tuts.
try to imagine how things work. and when you know how to use winice, explore any prog.

2) WinICE

What is it ? the best debugger on earth for Win95/98 (didn't test it on 98)
The best, the great WinICE.
How to use it

2.1) start.

Install it, press Ctrl+d (you should be in winice then), press Ctrl+F1 for a better view, type
'faults off' enter
'wl' enter.
F2

Ok
now you're in winice.
you see all the register in the upper part. with the register flags, as I told you, under ESI=.
you see the memory dump under the register. it's just the content of the memory, at any adress
you see the code under the memory dump. it's where the instructions to be executed are shown.
the highlighted instruction is the current one.
and under the code, the place where you enter orders.

2.1.1) important function keys.

basic :

F1 = Help. like typing 'help' in the command line
F2 = switch on off register windows. useful when you see a mov ax,[cx] and when you want to quickly see what's on cx adress.

F4 = see the output (windows) . any key to swith back to WI. note : the execution is halted
     when pressing F4 : it's just 'a snapshote'.
F5 = like pressing ctrl+D, it switches back to W95. the execution is resumed, with WI running
     behind of course.
F6 = when you're not in the dump memory windows, switches between code and command windows.

'advanced' function keys :
F7 = go to the code window, put the cursor on an instruction under the current (highlighted one)
     instruction, press F7 : it means 'goto here'. all the instructions are executed until this
     line is reached.
F8 = trace into : means one step instruction, including go 'into' a call instruction (see F10)
F9 = in the code windows : set a breakpoint on the instruction on/off.
F10 = step over : means one step instruction, skipping call. (see F8)
F11 = special. we'll see later.
F12 = special. we'll see later.

don't forget to use the help command.
try to use it before saying that you don't understand.

2.1.2) important commands

r = makes it possible to modify registers
d [an adress] = makes the memory dump show you the [adress] adress (:)).
e = edit memory
s = search in memory
 if you want to search 'Your name' in memory type
 s 0 l FFFFFFFF 'Your name'
 don't forget the zero, the l (L) and the eight 'F' before to do a full search

u can use the mouse too.

2.2) serious things.

2.2.1) breakpoints.

a breakpoint is something that will say to winice : stop here. and w8t

example for explanation : u enter a name, when you want to stop the program when it reads
the name you entered (to see what's he's doing later)
then type
bpx getdlgitemtexta (for example)
it will make WI stop when this windows command will be executed.

the first thing to do for cracking is entering the program, as closest as the "serialcheck"
as possible.
so the breakpoint is the starting point.

they are different kinds of breakpoints
most useful ones :
BPX : breakpoint on execution. when a adress get executed
BPM : breakpoint on memory. when an adress is read or written.

2.2.2) many important notes

2.2.2.1) in order not to loose time in analysing Kernel32.dll

when u use bpx [the name of a windows function]
 you'll stop at the beginning of this function (perhaps in the Kernel32.dll).

unless you want to analyse the whole windows, press F12. it will makes you out of the windows DLL. see F11 and F12 later.

2.2.2.2) what program are you analysing ?

in winice, at the bottom, on the left, you see the task you're studying.
if you see 'the shareware I want to crack', it's good.
if you see 'explorer' or 'kernel32' perhaps you made a mistake.

imagine. several tasks are running. explorer, winamp, AVP antivirus, etc... (i don't know)
if you press Ctrl+d at anytime, perhaps winamp was currently executed, etc...
so you should know that in fact you're not in your 'tobecracked' program.
good advice : close as many apps as possible. winice will break on any task if you asked to break on a too basic function. it won't make a difference between the target and winamp, for example.

moreover, when your task is calling a windows functions, the instructions executed will belong to a windows DLL. you can see the "current file you're in" between the code and the command windows.

to conclude, if you see kernel32/user/user32/gdi/VMM between the code and the commands windows you're not currently in front of a code you're likely to analyse (if you're reading this tut).
so when you enter such a dll, try to get out as soon as possible (press F10, not F8 !)
similarily, when you're in your prog, you see a
call USER!Dlgitemtexta
press F10 to go over it.

Hint : unload as many apps as possible. they could make WI break, at unwanted points.

2.2.2.3) about the export list

I talked about call USER!Dlgitemtexta. what is it ?
if you see softice initialisation settings in the symbol loader
there is the export dll section : what is it ?

first : a dll like the exported one is a set of function called by programs (for example for reading a registry key)
if you don't put a dll like kernel32.dll in the export list, look what it can do

not in the export list	in the export list
call xxxxxxxx	call kernel32!hmemcpy

here hmemcpy is a function contained in the kernel32.dll file.
so if you don't put a dll like this in the export list, perhaps you'll execute a call, and land in the middle of an unuseful dll. you should have pressed F10 to skip the call.
but if you did put it, winice will recognize the call as a call to this function
and will show you the name instead of an adress.

I'm sure you'll see what's happening.

the task is still your shareware (in the bottom right) but a function of kernel32dll is called so the current file showed in the code (between the command and the codewindows) is kernel32
.

2.2.2.4) About the symbol loader

except for the export dll list, the symbol loader is useless for a beginner.

Prophecy adds that it's useful anyway, if you want to break when the program starts loading.
for example, when a second check is carried out when restarting (classic : the prog says "good serial, I will restart now", and then , SECOND CHECK)

open the symbol loader
go to file=>open. select the EXE of the prog
then press CTRL−L
symbol loader will break on entry point of prog

2.2.2.5) Explanation about F11 and F12

when do I use them ?
F11 : you set a breakpoint on a function (getdlgitemtexta for example) and winice is just at the beginning of this function. as you don't want to study it (I think), Press F11.
You'll go out of the function directly. but don't press any other key like F8 or F10.
directly press F11.
F12 : you are in the middle of a function you don't want to study.
example : if you set a breakpoint on hmemcpy, when winice wil break,press F11 (as I said before). but since Hmemcpy as been called by a Windows DLL (usually), and you want to get out of it, press F12.
in other words, press F12 to do the same as F11, but this one works when you're not just at the beginning of a function

What do they do really ?
F11 : when a function is called, the info for coming back (with a RET) to the instruction after the CALL command are stored in memory. if nothing is changed, WI can use them to go to this instruction
F12 : this key tells WI to step over (not going into calls) all instructions until it founds a RET.

F11 Example:
you typed BPX GETDLGITEMTEXTA

the code is like this

```
xxxxx
.......
call XXXX!GETDLGITEMTEXTA
```

6 "crackers" have tried to crack Keytext, making useless keymakers, that passed only the first check.

The new trend in software protection is to change the check from a version to another, but with more checks added.
example : use a 3.1 generated serial in version 3.2 of getright, and you'll have a message later telling you this serial is pirated.
I say "trend" because it's used more and more. (that's nice, it means more job !
only Winzip hasn't changed since version 4 (if you know a winzip 3 send me plz))
for example
MP3Wolf, XWolf, ...Wolf, Getright, ADC, Keytext, BulletprofFTP, CDRWin,Virtual Turntable,etc...

3.3) more help : some real algorithm

Well, you wonder ... is all that rubbish, or is it true ?
I give you some REAL algorithm, taken from REAL shareware.

Ultraedit32 (old version, 2.XX I think)
N=name length
S=sum of all the name's characters code
Serial=64*S*S+N*N

CDWizzard 4
(you enter a name, it gives a key, then enter the password)
Password=11/7*key (rounded, since word are used)

WinPGP (1.XX version, I think)
S=sum of all the name's characters code
serial = (S+32)*196618

Jigsaw 2

```
Serial:=51659;
for (all name's chars) do Serial=2*Serial+Nth_char_code;
```

serial for 'A' is 37847, for 'B' is 37848

3.4) Which Breakpoints ?

Reading text :
 classic
 GETDLGITEMTEXT (16 bit EXE−target)
 GETWINDOWTEXT ("" """)
 GETDLGITEMTEXTA (32 bit EXE−target)
 GETWINDOWTEXTA ("" "")
if it doesn't work, use this one ! HMEMCPY

Messagebox
MESSAGEBOX (16b)
MESSAGEBOXA
MESSAGEBOXEXA
MESSAGEBEEP

of course, it's a shortlist here. but good for starting.

4) Last word

Don't forget the first crack is long... VERY LONG...
but when you finished (especially something that looks magic like a Keymaker...)
You'll feel... proud, and happy... (like shouting , bouncing, dancing, singing maria carey (joke)
and don't forget : keymaker (when possible of course) are better than simple cracks.

BTW : If cracking made ppl sing maria carey, I would have already stopped !

5) about this tutorial

I thank all the people that helped me for this tutorial, and all the people that
tought me how to crack.
Hi to all of my close or far friends, in no particular order :
Cafein,Prophecy,aLC, Toclean, Skates, infamous, docm, hayras,kwai_lo
Knight , Freeman, Blade, Psylogene, Pepsifan, snowbull, Watt Willy, Kosh/blacklord
SIRAX, Ferkiller, Bisoux, Kitt, Ti, Zyglute ....

AND DON'T FORGET :
Monica Lewinsky, Boris Eltsin, Benyamin netanyahou, and ryutaro hashimoto....
Edith piaf, Misoguchi, Bergman , bill gates, barbie, ken, ryu, etc...

```
(BTW : I think the last ones are mistakes...)


Crashtest [tNO]
```