Cracking Tutorial Compilation Vol.1.

Converted to PDF

Written by Tikka Wang

```
   _|_  o |__, |__,  __      __|      __   __       |__,
    |_, | |  \ |  \ (__(  -  (__|  .  (___ (__)  .  (__(_ |  \

   - by Tikka, email:     -               - no elite ascii art-
    - akkit_is@hotmail.com -             - irc.cocytusuk.org -
     -   Software Cracking Issue 1 - Covering the basics   -
```

Introduction

This document is for covering the basics of software cracking.

Why crack software?

I personaly want to crack software because i enjoy the challenge also it feels quite
nice making a serial number for something or removing a nag screen.
Most cracking tutorials say stuff like, this is only for educational purposes and to an
extent i would say this is right but software is extremly expensive and cracked software
is distributed so easily accross the internet that its far to easy to just search for a crack
on google or use a p2p network.

What needs cracking?

 - Limited functionality (cant save/print because the menu item is greyed out)
 - Nag screen (asking to be registered every time its loaded for example)
 - Serial number (finding a serial number or creating a keygen)
 - Dongles (Autocad for example needs a dongle, an electronic key that will be
   plugged into the back of the pc)

Above i have listed the most common things to fix, there are others date/time locks etc.

Tools you may need?

```
OllyDbg/softice -  debuger       (live debuging)
win32dasm/ida   -  dissassembler (for dead listings)
UltraEdit32     -  hex editor my fav. :)
```

How do you crack?

Different protections need different techniques.
A simple nagging message box could be patched by removing the code from the program.
Assembly language uses mnemonics to make up the language,

```
        cmp     YOUR_SERIAL, REAL_SERIAL
        je      ADDRESS
```

As a software cracker, your main interest is aimed at the programming above.

```
jmp = jump to a line of code
CMP = compare, used to check real and false serials.
je  = jump if eual to.
jz  = jump if not equal to.
```

```
if yourserial = internalserial then showgoodboy()
```

each mnemonic has an op code.
example:
NOP = 90 ; nop means no operation, it is basicaly a blank space.

I have taken a snippet of dissassembled code from IDA, this is called a dead listing.

```
_text:00404A84                 pop     ecx
_text:00404A85                 mov     edi, eax
_text:00404A87                 pop     ecx
```

```
_text:00404A88                 test    edi, edi
_text:00404A8A                 push    0
_text:00404A8C                 jz      short loc_0_404A9A
_text:00404A8E                 push    offset aFullStealthEnabled ; "Full Stealth Enabled"
_text:00404A93                 push    offset aThanksForRegisteringFeelFreeToVisitOurWebSiteAtHttp ;
"Thanks for registering!  Feel free to v"...
_text:00404A98                 jmp     short loc_0_404AA4
_text:00404A9A ; ---------------------------------------------------------------------------
_text:00404A9A
_text:00404A9A loc_0_404A9A:                           ; CODE XREF: sub_0_404A3D+4F#j
_text:00404A9A                 push    offset aRegistrationInvalid ; "Registration Invalid"
_text:00404A9F                 push    offset
aTheRegistrationInformationYouEnteredDoesNotMatchOurData ; "The registration information you entere"...
_text:00404AA4
```

This is what you are up against, confusing shit eh?

i said above that one way to crack would be to NOP out code, its alright for nags and make some programs registered easily.

```
   test    edi, edi
   jz      Jump to bad registration if edi doesnt = 00
   push    offset aRegistered ; "Registered"
   Say thank you because registration was ok
```

if you changed the JZ intruction for NOP's you would effectivly remove the serial checking, whatever serial you entered in to the program would register the program.
some programs will enter the CORRECT serial in to the registry and when you restart it will stay registered other times it will restart as unregistered.
this would be because some programs will store the FAKE serial you entered into the program and our patch only effects the registration routine not the start up code for checking for the registry key.

```
   _|_  o |__, |__,  __       __|       __   __          |__,
    |_, | |¯ \ |¯ \ (__(  -  (__|  .  (___ (__)  .  (__(_ |¯ \

   - by Tikka, email:     -              - no elite ascii art-
    - akkit_is@hotmail.com -            - irc.cocytusuk.org -
     -   Software Cracking Issue 2 - eat, sleep and dream. -
```

Introduction

Since ive been getting back into the swing of things, ive found myself to have a few
strange dreams, one was solving a cracking problem in my sleep but couldnt remmember
when i woke up and as for tonights episode..

This evening at bout 9:40pm i wake up and think omfg, i have some odd dreams.
i tell henson of my dream and thought it would be a laugh to add to my tutorial section
as i havent writen my next tutorial.
here it is.

```
<Akkit0r> beep
<henson> beep beep
<Akkit0r> ;/
<Akkit0r> im tired
<Akkit0r> been asleep
<Akkit0r> ;/
<Akkit0r> i hve really strange dreams
<Akkit0r> ;/
<henson> lol
<Akkit0r> i do really
<Akkit0r> right one was
<Akkit0r> we tried backing up a car to the endge of a gravil cliff
<Akkit0r> (like u get a boat in and our water)
<Akkit0r> and it was summin to do with letting soemthing/one out
<Akkit0r> and being able to get the car back up
<Akkit0r> now we put like (my idea) tempery fencing
<Akkit0r> but its like
<Akkit0r> beach wind breaks
<Akkit0r> with workman metal poles stuck in the ground
<Akkit0r> u`
<Akkit0r> |
<Akkit0r> |
<Akkit0r> |
<Akkit0r> |
<Akkit0r> like so
<Akkit0r> but i dont remember that being tested
<Akkit0r> but
<Akkit0r> scot from (tv series)emerdale somehow got hurt at that time
<Akkit0r> and died
<Akkit0r> but
<Akkit0r> they managed to keep his brain alive,
<Akkit0r> inside a fucking debugger
<henson> lol
<Akkit0r> i saw mov ecx, eax or summin, and another line below similar
<Akkit0r> no
<Akkit0r> push eax
<Akkit0r> was the line it was on
<Akkit0r> now i and the doctors couldnt decide if..
<Akkit0r> the debugger was before or after him dieing
<Akkit0r> i did something
<Akkit0r> which i dun really know
<Akkit0r> i think i continued the debugger
<Akkit0r> and he was in hospital preparing to die
<Akkit0r> had just a really short time
<Akkit0r> but it didnt bother me
<Akkit0r> then i had another dream in a pub
<Akkit0r> but had been out and bout, maybe on a train
<Akkit0r> im not sure
<Akkit0r> got to this pub
<Akkit0r> this pub had a seperate shop
<Akkit0r> but i took the batterys to the bar
<Akkit0r> i was gonna pay but i saw a friend
<Akkit0r> so i didnt buy batterys
<Akkit0r> sat with her for a minute
<Akkit0r> then woke up
<Akkit0r> is that strange?
<Akkit0r> (and i remembered it all)
<henson> ye
<Akkit0r> im putting that on my site
<Akkit0r> cracking tutorial part 2
<Akkit0r> what u recon?
<Akkit0r> warning
<henson> oookkkkaaayy
```

bloody strange eh?
Ive found that you cant get it out of your head, i was thinking bout it at work inbetween thinking i work with cunts lol.

I was also thinking, when i first got into cracking i found many great tutorials on patching, softice breakpoints, nagg screen removing
all extremly important.
Problem is i have so much to write and find it difficult to keep it organised enough for text,
it is worth searching google for cracking tutorial.
http://neworder.box.sk <-- many documents and articles in many scenes, cracking hacking.
http://astalavista.box.sk <-- security search engine where i found most tutorials from.
the box.sk network has many sites, dvd phones and more, at the top of the page is links take a
look through the network.

```
   _|_  o |__, |__,  __      __|     __  __        |__,
    |_, | |  \ |  \ (__(  -  (__|  .  (___ (__)  .  (__(_ |  \

   - by Tikka, email:     -             - no elite ascii art-
    - akkit_is@hotmail.com -           - irc.cocytusuk.org -
     -   Software Cracking Issue 3 - Fishing for a serial  -
```

Introduction

After writing an intresting article on dreams i felt a need to bang out another issue.
Fishing for a serial number, nice and easy ;)

What do we need?

```
Target : BossKey.EXE
Website: http://www.sonetics.net
Tools  : OllyDbg1.09c
```

Application intoduction

BossKey is a stealthing application, it will hide your running programs from eyes.
Perfect for college students who want to chat in irc all day.

Lets go!

OK, Fire up bosskey and what do we see?
A nag, not only do we have a message box but a help file to close aswell, ARGH!
You will also notice in the message that you can only stealth for 30 seconds at a time.
Now we are inside the program you can see a register button, lets click it and see where
it takes us, ahh enter serial.
tikka
1234
click ok -> Your blah isnt on our database.
well isnt that just wonderful!

Ok fire up your debugger and stick in bosskey.exe

Chugg Chugg.. When its ready right click the disassembled code window,
Search for -> ALL referenced text strings.

```
004048C1   PUSH BossKey.0040B5B0                     ASCII "Incorrect password"
004048C6   PUSH BossKey.0040B588                     ASCII "The password you entered is incorrect!"
00404A12   PUSH BossKey.00407F08                     ASCII "https://order.kagi.com/?APO"
00404A29   PUSH BossKey.00407F08                     ASCII "https://order.kagi.com/?APO"
00404A2E   PUSH BossKey.0040B5C4                     ASCII "open"
00404A8E   PUSH BossKey.0040B6F0                     ASCII "Full Stealth Enabled"
00404A93   PUSH BossKey.0040B688                     ASCII "Thanks for registering!  Feel free to visit our web site at
http://www.sonetics.net for future updates."
00404A9A   PUSH BossKey.0040B670                     ASCII "Registration Invalid"
00404A9F   PUSH BossKey.0040B5CC                     ASCII "The registration information you entered does not match our database.  Please
make sure you have entered your user name and serial number exactly as it was given."
```

"The password you entered is incorrect!"
This isnt what we want, this is when the program is in stealth and user must enter a password
to unhide the hidden applications.
Further down we see, ' ASCII "The registration information you entered does not match '.
Click this once and then press F2, F2 will set a Breakpoint(toggle) for when this message is
displayed.

It is now fine to close the strings window and start bosskey inside teh debugger, to do this click the RUN icon or from the menu bar,
Debug -> Run (F9)

Next you will see the nagging messagebox and help file.
Close them, Click Register and enter tikka, 1234.
On one machine, OllyDBG kicks in here and needs me to click ok but on my other machine its doesnt
irritate me.
When you click ok, the debugger will kick in and you wont see the invalid registration box yet because the debugger stopped the program
just before it.

```
00404A9F   . 68 CCB54000    PUSH BossKey.0040B5CC                    ;  ASCII "The registration information you entered does not match our
database.  Please make sure you have entered your user name and serial number exactly as it was given."
```

Well, it has already checked our fake serial number with the correct one inside the program.
It could be a good idea to use IDA and dissassemble bosskey and look its dead listing,
Makes it easier to understand what the program is doing.

Lets scroll up in the debugger and see what happens before this message.

```
00404A8C  JE SHORT BossKey.00404A9A
00404A8E  PUSH BossKey.0040B6F0                    ;  ASCII "Full Stealth Enabled"
00404A93  PUSH BossKey.0040B688                    ;  ASCII "Thanks for registering!
00404A98  JMP SHORT BossKey.00404AA4
00404A9A  PUSH BossKey.0040B670                    ;  ASCII "Registration Invalid"
00404A9F  PUSH BossKey.0040B5CC                    ;  ASCII "The registration informat
```

just a few lines above we see a conditional jump, JUMP IF EQUAL.
BossKey.00404A9A = Process "BossKey" @ location 00404A9A which in this case points to the

invalid serial message.

-- begin note --
You are probebly thinking lets patch this and make it accept any serial number.
Problem with this would be, when you restart the application it would have the wrong serial
number stored in the registry and will mean the program starts up as unregistered every time.
-- end note --

Ok lets see if we can find the real registration by setting a breakpoint on the line

```
00404A8C  JE SHORT BossKey.00404A9A
found above
00404A8E  PUSH BossKey.0040B6F0                    ;  ASCII "Full Stealth Enabled"
```

using f2 we have set a break point on the line and now need to let the program continue running.
press f9 to run.

Ok, the message will appear saying that we have an invalid serial click ok to close that and
click ok again to enter our fake serial.

The debugger will appear and our breakpoint will be there looking at us,
When a serial is being generated inside the application memory registered store information.
if you look in the registers box (right hand window) you will see our fake username and serial.
EBX = "tikka"
ECX = "1234"
Unfortunatly it seems the real serial is not there so lets trace back up the program some more.

```
00404A7F  CALL BossKey.00404B15
00404A84  POP ECX
00404A85  MOV EDI,EAX
00404A87  POP ECX
00404A88  TEST EDI,EDI
00404A8A  PUSH 0
00404A8C  JE SHORT BossKey.00404A9A
00404A8E  PUSH BossKey.0040B6F0                    ;  ASCII "Full Stealth Enabled"
```

The next most significant line is the call to another line of code inside of BossKey @ 00404B15
Inside this call there could be anything, perhaps the routine to check the serial ;).
set a breakpoint on this line and continue the program again.
we have another couple of breakpoints already just keep them and keep continuing the application until u can get to enter the serial again.
Click OK, Debugger kicks in.

there are a few ways to navigate through the debugger and the code,
you can step in to calls and jumps.
step over which will still run the same code but instead of following the code into calls that are not important to us like user32 api calls.

lets Step into the call "00404A7F   CALL BossKey.00404B15" using F7,

take a look at the first and last lines and notice the

```
/$ Beginning of call
|. CODE
\$ End of call
```

this is handy when browsing the code because you can see that easily.
also notice the loop in the middle of this call ?

our name is stored inside ECX,
EAX is the loop integer. this means that when it wants the next letter in the string
it will use this as the pointer.

example

```
ECX = "tikka"
EAX = 0
MOV DL,BYTE PTR DS:[ECX+EAX+1]
```

This will put inside DL the letter "i".
in BASIC it would look like
mid(name, EAX + 1, 1)

This is part of the Algorythm used to generate and compare the fake and real serials.

before the loop it gets our name and fake serial put into corresponding registers.
Altho at this stage only our name is important.

```
/MOV DL,BYTE PTR DS:[ECX+EAX+1]   ; get the next letter in advance, t(i)kka
|TEST DL,DL                       ; Check to make sure serial has another letter
|JE SHORT BossKey.00404AF0        ; if the serial has an odd length if so jump
|MOVSX EDI,BYTE PTR DS:[ECX+EAX]  ; move (t), the first letter into EDI
|MOVSX EDX,DL                     ; contents of dl to go into EDX (dl is set4 lines above)
|IMUL EDI,EDX                     ; Multiply the ascii values of EDX and ESI storing in ESI
|ADD ESI,EDI                      ; Add EDI with ESI, ESI now holds the TOTAL
|INC ECX                          ; increment the loop integer by 1
|INC ECX                          ; incremented it again by 1
                                  ; reason is that, because it scans a letter in advance
                                  ; so to it keep lined up properly it must do this.
|CMP BYTE PTR DS:[ECX+EAX],0      ; check to see if we are at the end of the username
\JNZ SHORT BossKey.00404AD4       ; if so do whatever next
```

If the serial number is off odd length then it will multiply edx with 7B (123 in decimal).

after the algorythm it must check to see if our name and serial match the serial that has just been generated ;)

-- begin note --
If you would like to watch the algorythm in action you could set a break point on the first line of the loop and step through the code line by line.
-- end note --

handy being able to see calls and loops easily isnt it :)

```
00404ABA  /$ 8B4424 08      MOV EAX,DWORD PTR SS:[ESP+8]
00404ABE  |. 56             PUSH ESI
00404ABF  |. 33C9           XOR ECX,ECX
```

```
00404AC1  |. 33F6           XOR ESI,ESI
00404AC3  |. 8020 00        AND BYTE PTR DS:[EAX],0
00404AC6  |. 8B4424 08      MOV EAX,DWORD PTR SS:[ESP+8]
00404ACA  |. 85C0           TEST EAX,EAX
00404ACC  |. 74 45          JE SHORT BossKey.00404B13
00404ACE  |. 8038 00        CMP BYTE PTR DS:[EAX],0
00404AD1  |. 57             PUSH EDI
00404AD2  |. 74 1C          JE SHORT BossKey.00404AF0
00404AD4  |> 8A5401 01      /MOV DL,BYTE PTR DS:[ECX+EAX+1]
00404AD8  |. 84D2           |TEST DL,DL
00404ADA  |. 74 14          |JE SHORT BossKey.00404AF0
00404ADC  |. 0FBE3C01       |MOVSX EDI,BYTE PTR DS:[ECX+EAX]
00404AE0  |. 0FBED2         |MOVSX EDX,DL
00404AE3  |. 0FAFFA         |IMUL EDI,EDX
00404AE6  |. 03F7           |ADD ESI,EDI
00404AE8  |. 41             |INC ECX
00404AE9  |. 41             |INC ECX
00404AEA  |. 803C01 00      |CMP BYTE PTR DS:[ECX+EAX],0
00404AEE  |.^75 E4
     \JNZ SHORT BossKey.00404AD4
00404AF0  |> 8A0401         MOV AL,BYTE PTR DS:[ECX+EAX]
00404AF3  |. 5F             POP EDI
00404AF4  |. 84C0           TEST AL,AL
00404AF6  |. 74 08          JE SHORT BossKey.00404B00
00404AF8  |. 0FBEC0         MOVSX EAX,AL
00404AFB  |. 6BC0 7B        IMUL EAX,EAX,7B
00404AFE  |. 03F0           ADD ESI,EAX
00404B00  |> 56             PUSH ESI                                 ; /<%X>
00404B01  |. 68 08B74000    PUSH BossKey.0040B708                    ; |format = "%X"
00404B06  |. FF7424 14      PUSH DWORD PTR SS:[ESP+14]               ; |s
00404B0A  |. FF15 58734000  CALL DWORD PTR DS:[<&MSVCRT.sprintf>]    ; \sprintf
00404B10  |. 83C4 0C        ADD ESP,0C
00404B13  |> 5E             POP ESI
00404B14  \. C3             RETN
```

The next breakpoint we will set on
00404B01 PUSH BossKey.0040B708   ; |format = "%X"
reason is that %X means capitalised Hex value.
we know that the algorythm is manipulating our text in hex so lets breakpoint here and contine the program.

keep telling the debugger to run untill application lets you enter the serial again,
on the third or so go it will continue out of the break points we set earlier.

click ok in the registration dialog to enter our name and serial.
dont bother stepping into the call just press F9 (run)
and the debugger will stop at our last break point, format = "%X".
We see nothing in the registers so lets step over the code F8.
lets keep stepping over any calls we find, we will come to RETN which will return the code
back to the calling routine keep stepping and try to watch the box under the disassembled code window while watching this aswell to see whats happening.

7 presses of F8 after the breakpoint on
00404B01 PUSH BossKey.0040B708   ; |format = "%X"
we have a check,

```
00404B4A  |. 50             PUSH EAX                                 ; /s2 = "8AE8"
00404B4B  |. 56             PUSH ESI                                 ; |s1
00404B4C  |. FF15 60734000  CALL DWORD PTR DS:[<&MSVCRT._mbscmp>]    ; \_mbscmp
```

EAX hold the generated serial
ESI holds our fake one
_mbscmp compares the two arguments.

```
00404B4A  |. 50             PUSH EAX                                 ; /s2 = "8AE8"
```

the hex at the end is the serial number for our name "tikka"
try it and see :)

i hope you have had fun ;)

```
   _|_  o |__, |__,  __       __|      __  __         |__,
    |_, | |  \ |  \ (__(  -  (__|  .  (___ (__)  .  (__(_ |  \

   - by Tikka, email:     -              - no elite ascii art-
    - akkit_is@hotmail.com -            - irc.cocytusuk.org -
     -   Software Cracking Issue 4 - Looking at a keygen   -
```

Introduction

This is going to be a follow up from the serial fishing example in the last issue, using the same
target which is bosskey by sonetics.
I plan to make this tutorial small as its 5am and i should be in bed ;)
besides im sure i covered plenty in the last issue so use it for reference if you need.

Converting from asm to c

```
00404ABF  XOR ECX,ECX                    ; ecx = 0
00404AC1  XOR ESI,ESI                    ; esi = 0
00404AC3  AND BYTE PTR DS:[EAX],0        ;          nothing really happening
00404AC6  MOV EAX,DWORD PTR SS:[ESP+8]   ;  just making sure there is a name entered
00404ACA  TEST EAX,EAX                   ;  moving stuff around, doesnt affect us for
00404ACC  JE SHORT BossKey.00404B13      ;    the keygen at all so lets move on ;)
00404ACE  CMP BYTE PTR DS:[EAX],0        ;
00404AD1  PUSH EDI                       ;           if no name skip algo
00404AD2  JE SHORT BossKey.00404AF0      ;

ECX holds our name
:start
/MOV DL,BYTE PTR DS:[ECX+EAX+1]   ; get the next letter in advance, t(i)kka
|TEST DL,DL                       ; Check to make sure serial has another letter
|JE SHORT BossKey.00404AF0        ; if the serial has an odd length if so jump
|MOVSX EDI,BYTE PTR DS:[ECX+EAX]  ; move (t), the first letter into EDI
|MOVSX EDX,DL                     ; contents of dl to go into EDX (dl is set4 lines above)
|IMUL EDI,EDX                     ; Multiply the ascii values of EDX and ESI storing in ESI
|ADD ESI,EDI                      ; Add EDI with ESI, ESI now holds the TOTAL
|INC ECX                          ; increment the loop integer by 1
|INC ECX                          ; incremented it again by 1
                                  ; reason is that, because it scans a letter in advance
                                  ; so to it keep lined up properly it must do this.
|CMP BYTE PTR DS:[ECX+EAX],0      ; check to see if we are at the end of the username
\JNZ SHORT start of loop          ; If not at end of string jump to start of loop

00404AF0  |> 8A0401         MOV AL,BYTE PTR DS:[ECX+EAX]            ; check if there is an even number of characters
00404AF3  |. 5F             POP EDI                                 ;
00404AF4  |. 84C0           TEST AL,AL
00404AF6  |. 74 08          JE SHORT BossKey.00404B00               ; if its even then jump over the next part of algo
00404AF8  |. 0FBEC0         MOVSX EAX,AL                            ; move AL (last characters hex value) into EAX
00404AFB  |. 6BC0 7B        IMUL EAX,EAX,7B                         ; multiply last letters value with 7B (123 decimal)
00404AFE  |. 03F0           ADD ESI,EAX                             ; add to the serial's total.
00404B00  |> 56             PUSH ESI                                ; /<%X>
00404B01  |. 68 08B74000    PUSH BossKey.0040B708                   ; |format = "%X" <---- this is the COMPLETED serial
00404B06  |. FF7424 14      PUSH DWORD PTR SS:[ESP+14]              ; |s                             in capitalised HEX
```

Explaination

```
It gets your name "tikka",
Then converts each character into its ascii value.
it takes the second, then first characters of each cycle in the loop   ; t and i
multiplys them together                                                 ; 74  * 69
then stores into a buffer which we could say is the total               ; total = 2F94
total = i*t
it makes eax = eax + 2
this is because we are moving along in steps of 2
goes to start of loop                                                   ; k and k
gets k*k then adds to total (which already has the value of i*t)        ; 6B * 6B = 2CB9
increments eax by 2 again                                               ; total = 2F94 + (6B*6B)
realises that we are at the end of our name                             ;                       (total = 5C4D)
and there is an ODD length of characters and jumpes to the next section; a and 7B
it then multiplys 'a' with 7B                                           ; 61 * 7B = 2E9B
adds to total                                                           ; add 2E9B with total and you get
makes it into uppcase hex.                                              ; '8AE8'
```

c++

```
#include <string.h>
#include <stdio.h>
// bosskey
int main(int argc, char *argv[])
{
          int total = 0;
          int i = 0; char  name[1024];
          printf("Enter name: ");       gets(name);
          while(i < strlen(name) + 1) {
          char ch1 = name[i];
          char ch2 = name[i+1];
          int mix; mix = ch1 * ch2;
          if (ch2==0) { mix = ch1*123; }
          total = total + mix; i++; i++;
          }
          printf("The serial: %X",total);
          char null[1024]; gets(null);
          return 0;
}
```

----

if at the moment you cannot understand c++ then here is basic quiv
Qbasic4.5

```
--------------------------------------------------------------------
lastone = 0
INPUT "enter your name: ", name$
IF LEN(name$) > 20 THEN PRINT "name to long"; : END
IF name$ = "" THEN name$ = "akkit"
FOR b = 1 TO LEN(name$) STEP 2
a = ASC(MID$(name$, b, 1))
IF b = LEN(name$) GOTO 20 ELSE c = ASC(MID$(name$, b + 1, 1))
GOSUB algo
NEXT b
20 blah = (a * 123)
current = current + blah
GOTO exit1
algo:
 current2 = a * c
 current = current2 + lastone
 lastone = current2
RETURN
exit1:
PRINT "Serial: ";
PRINT HEX$(current)
PRINT "more tutorials at http://www.tikka-d.co.uk"
--------------------------------------------------------------------
```

I have also included a mIRC script for the keygen,

```
--------------------------------------------------------------------
alias /keygen {
  var %i = 1
  var %ch1 = 0
  var %ch2 = 0
  var %mix = 0
  var %last = 0
  :start
  if %i > $len($1-) { goto end }
  if $mid( $1- , $calc( %i + 1 ) , 1 ) = $null { goto section2 }
  set %ch1 $asc($mid($1-,$calc(%i),1))
  set %ch2 $asc($mid($1-,$calc(%i + 1),1))
  set %mix $calc(%ch1 * %ch2)
  set %last $calc(%last + %mix)
  inc %i
  inc %i
  goto start
  :section2
  set %ch1 $asc($mid($1-,$calc(%i),1))
  set %mix $calc(%ch1 * 123)
  set %last $calc(%last + %mix)
  :end
  //say #1,0Key#15gen# #14# $+ $1- $+ # # $+ $base(%last,10,16) $+ #
}
```

I really recommend learning c, visual basic is not going to do you
any favours. QBasic can make exes but best to do it in c.
mIRC script example was coz i was bored.

Special thanks to henson for help with converting between languages.

```
   _|_  o |__, |__,  __     __|     __  __      |__,
    |_, | |¯¯\ |¯¯\ (__(  -  (__|  .  (___ (__)  .  (__(_ |¯¯\

   - by Tikka, email:     -              - no elite ascii art-
    - akkit_is@hotmail.com -            - irc.cocytusuk.org -
     -   Software Cracking Issue 5 - Resource Modification -
```

Introduction

Today we are going to talk about resource modification which is often refered to as
resource hacking which sounds pretty pants but the tool that i like is "resource hacker"
which isnt so pants. I just downloaded it because i havent needed it for such a long
time but when i did i found a really nice function that has been added since last time
i downloaded. Resource hacker now has a function for adding resources to the dialog, i
mean from a dialog you can select what you would like to add whereas before you needed to
know the scripting side of things. When i first became interested in changing the
appearance of an application was when i hex edited mirc, which took me a really long time.
I swapped the about and register menus around by changing their id and redesigned the about
box to appear registered, removed the version reply and also modified the title bar from
mIRC32 to my scripts name. Using a program called microangelo which is an icon editor i was
able to change the icon aswell. Now using resource hacker you can do everything you need,
In this paper i will demonstrate an extremly easy but handy modification to the shoutcast
DSP plugin for winamp. I hate how you cannot minimize the dsp, or more to the point the size.
All i want to see is the VU meter so to do this i made the dialog resizable.

Tools

Resource Hacker, (free) http://www.users.on.net/johnson/resourcehacker/
Microangelo, (not free) http://www.microangelo.us/
Hex editor, this can be used to look at what changed but for this example i wont bother.

Target
------

http://www.shoutcast.com/downloads/shoutcast-dsp-1-8-2b-windows.exe

SHOUTcast Source v1.8.2b

I didnt check for the latest version before writing this, but this technique would work for
other versions if they need it.
Download resource hacker.
Got it? Good!
open your winamp and look at the DSP plugin, no dialog control is frustrating so lets sort
it out. In the preferences dialog unselect the DSP plugin this will unload the dll from memory,
this will be handy for looking at changes. When the plugin has been modified, close and open
winamps preferences as this will then list both the new and old plugin versions so you can
easily compare the 2. Fire up resource hacker and open c:\program files\winamp\plugins\dsp_sc.dll,

```
[+] Icon
[+] Dialog
[+] Icon Group
[+] 240
```

we at present are only interested in Dialog, expand Dialog and expand the first item.

```
[+] Icon
[-] Dialog
   [-] 101
       X 1033
```

You will see "SHOUTcast Source" Dialog and a code window with the dialog shown as text.

```
-------------------------------------------------------------------------------
101 DIALOGEX 0, 0, 188, 282
STYLE DS_CENTER | WS_MINIMIZEBOX | WS_POPUP | WS_CAPTION | WS_SYSMENU
EXSTYLE WS_EX_CONTROLPARENT
CAPTION "SHOUTcast Source"
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
FONT 8, "MS Sans Serif"
{
   CONTROL "Tab1", 1000, "SysTabControl32", TCS_TABS | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 2, 2, 184, 279
   CONTROL "", 1001, BUTTON, BS_GROUPBOX | WS_CHILD, 8, 17, 171, 257
}
-------------------------------------------------------------------------------
```

You can change the dialogs caption, size and what you see from the code above.

```
STYLE DS_CENTER | WS_MINIMIZEBOX | WS_POPUP | WS_CAPTION | WS_SYSMENU
```

That is what is set for the dialog at design time, other properties are set during runtime.
For example, you cannot resize the dialog but in the code window it says
WS_MINIMIZEBOX which means the dialog can be minimized but when we see it running it has been
dissabled. What can we do instead? Well instead of getting into the code and stopping it
being dissabled lets just make it resizable. change the style line from:

```
STYLE DS_CENTER | WS_MINIMIZEBOX | WS_POPUP | WS_CAPTION | WS_SYSMENU
```

to:

```
STYLE DS_CENTER | WS_MINIMIZEBOX | WS_POPUP | WS_CAPTION | WS_SYSMENU | WS_THICKFRAME
```

Nice and simple, click compile script and then save.
In Winamp Preferences, Click on the DSP plugin and now try to resize..

```
REM
REM
REM
REM    _|_  o |__, |__,  __     __|     __  __            |__,
REM     |_, | |  \ |  \ (__(  -  (__|  .  (___ (__)  .  (__(_ |  \
REM
REM    - by Tikka, email:     -              - no elite ascii art-
REM     - akkit_is@hotmail.com -            - irc.cocytusuk.org -
REM      -   Software Cracking Issue 6 - SysDate Keygen, QB    -
REM
REM
REM
REM Introduction
REM ------------
REM
REM This Keygen is not commented due to the fact that it is not much different from
REM the bosskey keygen, Altho the registers have been used for the variable names
REM for simplicity. Forgive me for the crap loops at the very end, i couldnt be
REM arsed to find a nice little command to do it instead.. altho i do realise
REM that one loop, removing spaces would of done it.. ah well, it werks fine :)
REM
REM This was coded using QBasic4.5 reason is that it can compile .exe's
REM besides i like it ;)
REM use google to find "qbasic45" thats where i found it.
REM
REM If you require a more indepth tutorial or perhaps ported to another language
REM then feel free to email me.
REM
REM
REM
REM

CLS
INPUT "Enter Username: ", str1$
IF str1$ = "" THEN PRINT "Invalid Length": END
edx = 0

eax = 0
ecx = 107
FOR eax = 1 TO LEN(str1$)
edx = ASC(MID$(str1$, eax, 1))
edx = edx + edx * 4
ecx = ecx + edx * 4
NEXT eax
ebx = ecx

eax = 0
ecx = 107
FOR eax = 1 TO LEN(str1$)
edx = ASC(MID$(str1$, eax, 1))
edx = edx + edx * 4
edx = edx + edx * 4
ecx = ecx + edx * 8
NEXT eax
section1 = ebx
section2 = ecx
edx = ASC(MID$(str1$, LEN(str1$), 1))
edx = edx + 2
section3 = edx
eax = ASC(MID$(str1$, LEN(str1$), 1))
eax = eax + eax * 4
edx = eax + eax * 4
eax = edx * 4 + 1
section4 = eax


PRINT "Serial  Number: ";
FOR a = 2 TO LEN(STR$(section1))
PRINT MID$(STR$(section1), a, 1);
NEXT a
PRINT "-";
FOR a = 2 TO LEN(STR$(section2))
PRINT MID$(STR$(section2), a, 1);
NEXT a
PRINT "-";
FOR a = 2 TO LEN(STR$(section3))
PRINT MID$(STR$(section3), a, 1);
NEXT a
PRINT "-";
FOR a = 2 TO LEN(STR$(section4))
PRINT MID$(STR$(section4), a, 1);
NEXT a

REM I love silly loops, dont you?
```

Any questions aim them at akkit_is@hotmail.com

Bored and want to chat somewhere?
Http://www.turntablism.info/chat

Regards, Tikka.