Title: **How to inject code into a exe file**
Autor: **Iman Karim**
Email: iman.karim@smail.inf.fh-bonn-rhein-sieg.de
Home: http://www2.inf.fh-bonn-rhein-sieg.de/~ikarim2s/

***** THIS TUTORIAL IS ONLY FOR EDUCATIONAL PURPOSES!*****

(english mistakes are default :P)

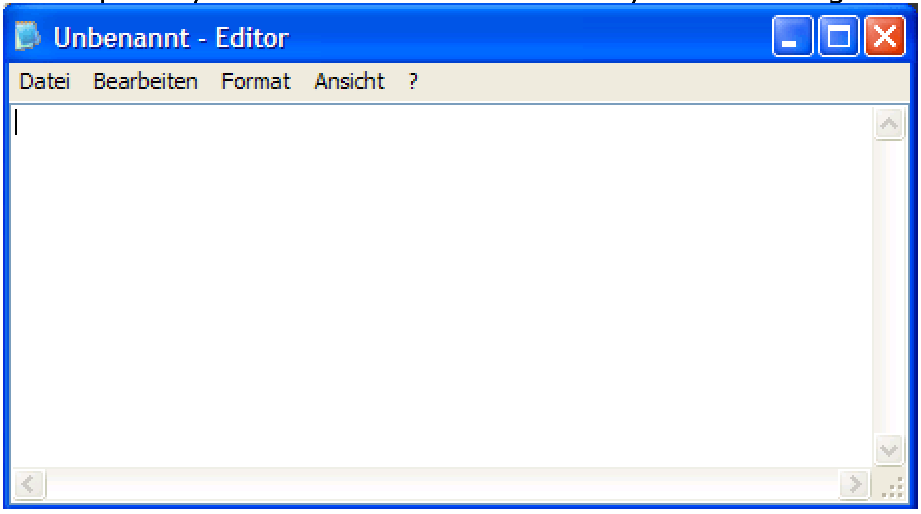
Content:

-Requirements-

At first we need a debugger. I prefer OLLYDBG(the best debugger on earth :P)
At twice we need a target application to inject our code.
I will take the windows NOTEPAD.EXE .
Make a copy if this EXE in a new folder named *CodeInjectTest*.
Thats all ;)

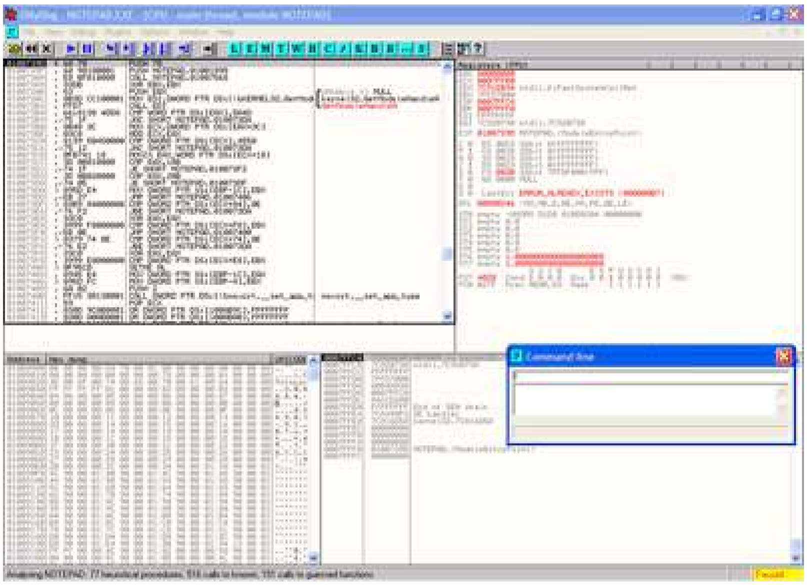
-Getting started-

Our goal is to inject some code into the *Notepad.EXE* .
In our case we'll inject a simple *MessageBox* at *Notepads* start.
Start up *Notepad* now and validize that *Notepad* is the original one.



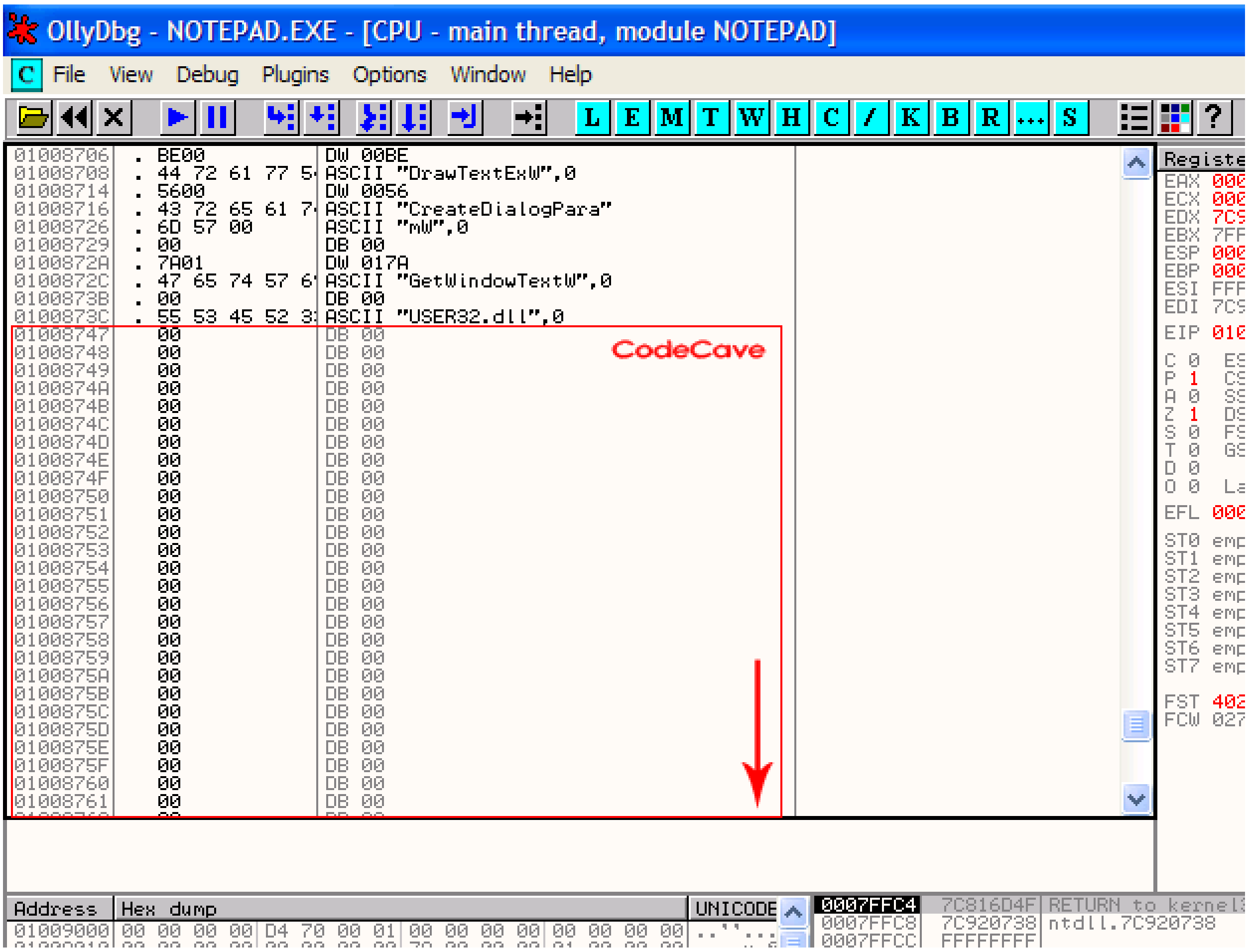
[My Notepad screenshot]

If you're sure now that's the original Notepad open the Notepad.exe with Olly.
Yuppi! If you're ready you get this window:



[click to enlarge]

Because we're going to inject some code we've to have some space to inject it.
In a EXE file are a lot of *CodeCaves* were nothing is done (DB 00).
So lets scroll the CPU window a little bit down until you find a *CodeCave*(look below).



OllyDbg - NOTEPAD.EXE - [CPU - main thread, module NOTEPAD]
C File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S
CodeCave
Registe
Address Hex dump UNICODE

```
01008706  .  BE00          DW 00BE
01008708  .  44 72 61 77 5 ASCII "DrawTextExW",0
01008714  .  5600          DW 0056
01008716  .  43 72 65 61 7 ASCII "CreateDialogPara"
01008726  .  6D 57 00      ASCII "mW",0
01008729  .  00            DB 00
0100872A  .  7A01          DW 017A
0100872C  .  47 65 74 57 6 ASCII "GetWindowTextW",0
0100873B  .  00            DB 00
0100873C  .  55 53 45 52 3 ASCII "USER32.dll",0
01008747     00            DB 00
01008748     00            DB 00
01008749     00            DB 00
0100874A     00            DB 00
0100874B     00            DB 00
0100874C     00            DB 00
0100874D     00            DB 00
0100874E     00            DB 00
0100874F     00            DB 00
01008750     00            DB 00
01008751     00            DB 00
01008752     00            DB 00
01008753     00            DB 00
01008754     00            DB 00
01008755     00            DB 00
01008756     00            DB 00
01008757     00            DB 00
01008758     00            DB 00
01008759     00            DB 00
0100875A     00            DB 00
0100875B     00            DB 00
0100875C     00            DB 00
0100875D     00            DB 00
0100875E     00            DB 00
0100875F     00            DB 00
01008760     00            DB 00
01008761     00            DB 00
```

```
EAX 000
ECX 000
EDX 7C9
EBX 7FF
ESP 000
EBP 000
ESI FFF
EDI 7C9

EIP 010

C 0  ES
P 1  CS
A 0  SS
Z 1  DS
S 0  FS
T 0  GS
D 0
O 0  La

EFL 000

ST0 emp
ST1 emp
ST2 emp
ST3 emp
ST4 emp
ST5 emp
ST6 emp
ST7 emp

FST 402
FCW 027
```

```
01009000 00 00 00 00 D4 70 00 01 00 00 00 00 00 00 00 00 ..''...

0007FFC4  7C816D4F  RETURN to kernel3
0007FFC8  7C920738  ntdll.7C920738
0007FFCC  FFFFFFFF
```

[CodeCaves]

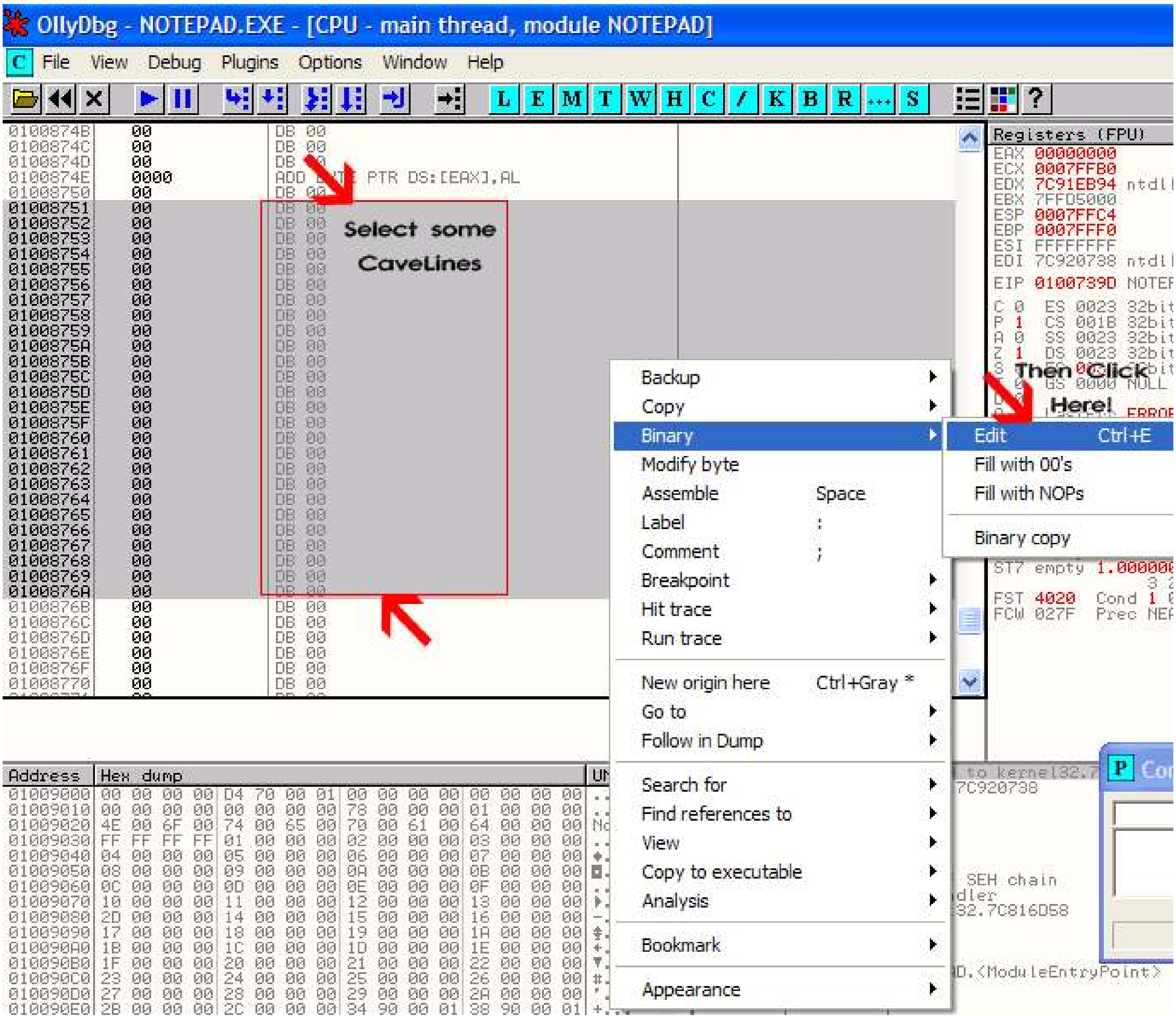
Do you see the red box I've drawn for you? ;) THIS is a CodeCave!
Here we can inject some custom code without interfering the programs flow.
If you know the API call for a MessageBox you don't need to
read this lines.

> ***** *START QUOTE FROM THE WIN32 PROGRAMMERS REFERENCE*
>
> The MessageBox function creates, displays, and operates a message box.
> The message box contains an application-defined message and title,
> plus any combination of predefined icons and push buttons.

```
int MessageBox(
    HWND hWnd,  // handle of owner window
    LPCTSTR lpText, // address of text in message box
    LPCTSTR lpCaption,  // address of title of message box
    UINT uType  // style of message box
   );
```

> ***** *END QUOTE FROM THE WIN32 PROGRAMMERS REFERENCE*

This is all you need to know about the MessageBox.
Now its time to do something with the CodeCaves.
At first to use the MessageBox we need to create some text for output it on the MessageBox. In the following picture I've selected some lines of the CodeCaves and highlighted the BinaryEdit menu for you.



Select some CaveLines

Then Click Here!

```
0100874B   00     DB 00
0100874C   00     DB 00
0100874D   00     DB 00
0100874E   0000   ADD BYTE PTR DS:[EAX],AL
01008750   00     DB 00
01008751   00     DB 00
01008752   00     DB 00
01008753   00     DB 00
01008754   00     DB 00
01008755   00     DB 00
01008756   00     DB 00
01008757   00     DB 00
01008758   00     DB 00
01008759   00     DB 00
0100875A   00     DB 00
0100875B   00     DB 00
0100875C   00     DB 00
0100875D   00     DB 00
0100875E   00     DB 00
0100875F   00     DB 00
01008760   00     DB 00
01008761   00     DB 00
01008762   00     DB 00
01008763   00     DB 00
01008764   00     DB 00
01008765   00     DB 00
01008766   00     DB 00
01008767   00     DB 00
01008768   00     DB 00
01008769   00     DB 00
0100876A   00     DB 00
0100876B   00     DB 00
0100876C   00     DB 00
0100876D   00     DB 00
0100876E   00     DB 00
0100876F   00     DB 00
01008770   00     DB 00
```

```
Registers (FPU)
EAX 00000000
ECX 0007FFB0
EDX 7C91EB94 ntdll
EBX 7FFD5000
ESP 0007FFC4
EBP 0007FFF0
ESI FFFFFFFF
EDI 7C920738 ntdll
EIP 0100739D NOTEF
C 0  ES 0023 32bit
P 1  CS 001B 32bit
A 0  SS 0023 32bit
Z 1  DS 0023 32bit
S 0  FS 0    32bit
     GS 0000 NULL
               ERROR
ST7 empty 1.000000
              3 2
FST 4020  Cond 1 0
FCW 027F  Prec NEF
```

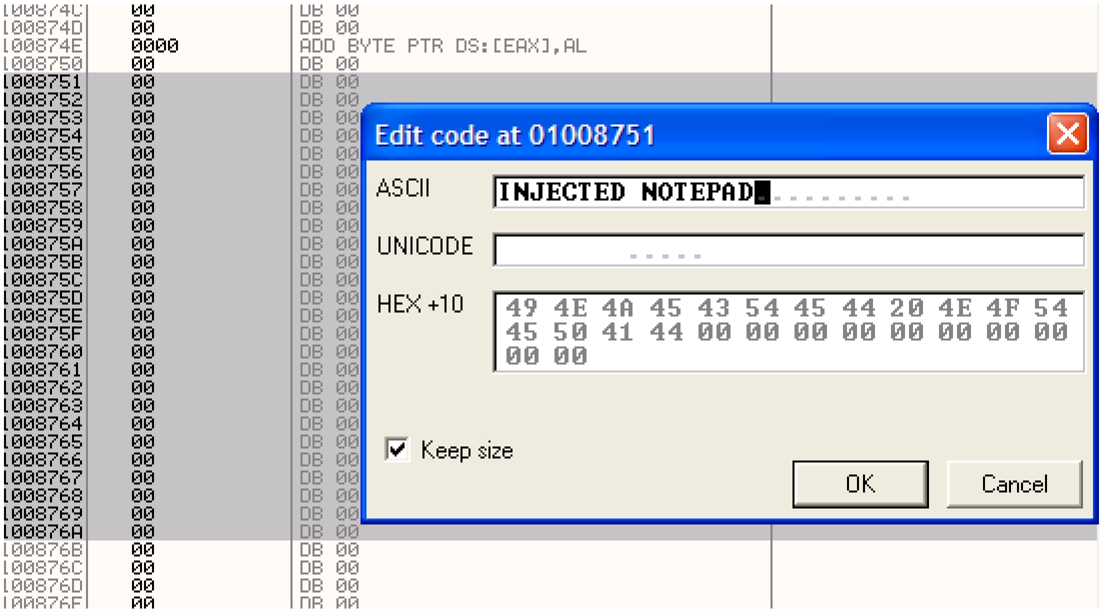
```
Address  Hex dump                                         UN
01009000 00 00 00 00 D4 70 00 01 00 00 00 00 00 00 00 00 ..
01009010 00 00 00 00 00 00 00 00 78 00 00 00 01 00 00 00 ..
01009020 4E 00 6F 00 74 00 65 00 70 00 61 00 64 00 00 00 No
01009030 FF FF FF FF 01 00 00 00 02 00 00 00 03 00 00 00 ..
01009040 04 00 00 00 05 00 00 00 06 00 00 00 07 00 00 00 ♦.
01009050 08 00 00 00 09 00 00 00 0A 00 00 00 0B 00 00 00 ◘.
01009060 0C 00 00 00 0D 00 00 00 0E 00 00 00 0F 00 00 00 ..
01009070 10 00 00 00 11 00 00 00 12 00 00 00 13 00 00 00 ►.
01009080 2D 00 00 00 14 00 00 00 15 00 00 00 16 00 00 00 -.
01009090 17 00 00 00 18 00 00 00 19 00 00 00 1A 00 00 00 ↨.
010090A0 1B 00 00 00 1C 00 00 00 1D 00 00 00 1E 00 00 00 ←.
010090B0 1F 00 00 00 20 00 00 00 21 00 00 00 22 00 00 00 ▼.
010090C0 23 00 00 00 24 00 00 00 25 00 00 00 26 00 00 00 #.
010090D0 27 00 00 00 28 00 00 00 29 00 00 00 2A 00 00 00 '.
010090E0 2B 00 00 00 2C 00 00 00 34 90 00 01 38 90 00 01 +.
```

```
to kernel32.7
7C920738

SEH chain
dler
32.7C816D58

D.<ModuleEntryPoint>
```

If you pressed on Binary=>Edit or (CTRL+E) you will see following window.
Just fill it out like me if you want.



```
l00874C    00      DB 00
l00874D    00      DB 00
l00874E    0000    ADD BYTE PTR DS:[EAX],AL
l008750    00      DB 00
l008751    00      DB 00
l008752    00      DB 00
l008753    00      DB 00
l008754    00      DB 00
l008755    00      DB 00
l008756    00      DB 00
l008757    00      DB 00
l008758    00      DB 00
l008759    00      DB 00
l00875A    00      DB 00
l00875B    00      DB 00
l00875C    00      DB 00
l00875D    00      DB 00
l00875E    00      DB 00
l00875F    00      DB 00
l008760    00      DB 00
l008761    00      DB 00
l008762    00      DB 00
l008763    00      DB 00
l008764    00      DB 00
l008765    00      DB 00
l008766    00      DB 00
l008767    00      DB 00
l008768    00      DB 00
l008769    00      DB 00
l00876A    00      DB 00
l00876B    00      DB 00
l00876C    00      DB 00
l00876D    00      DB 00
l00876E    00      DB 00
```

Edit code at 01008751

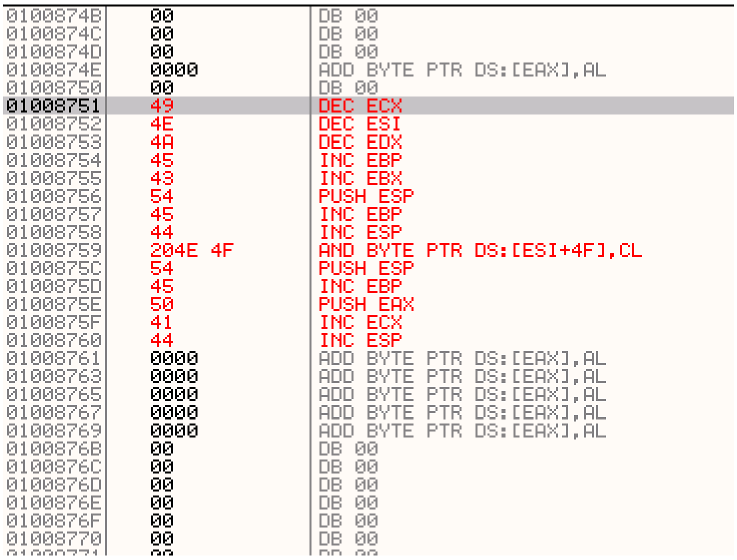
ASCII: INJECTED NOTEPAD

UNICODE:

HEX +10:
```
49 4E 4A 45 43 54 45 44 20 4E 4F 54
45 50 41 44 00 00 00 00 00 00 00 00
00 00
```

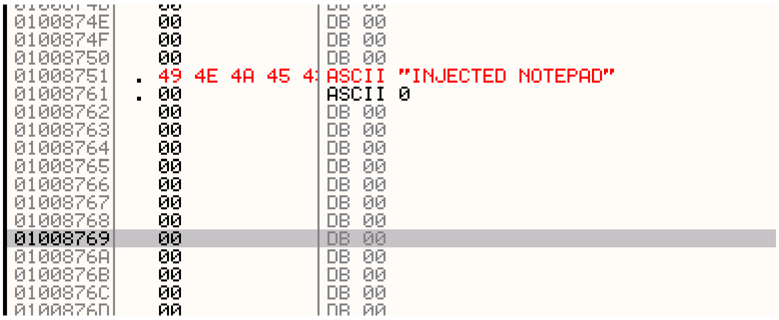
☑ Keep size

Press OK and you'll see the modified code in red:



```
0100874B    00        DB 00
0100874C    00        DB 00
0100874D    00        DB 00
0100874E    0000      ADD BYTE PTR DS:[EAX],AL
01008750    00        DB 00
01008751    49        DEC ECX
01008752    4E        DEC ESI
01008753    4A        DEC EDX
01008754    45        INC EBP
01008755    43        INC EBX
01008756    54        PUSH ESP
01008757    45        INC EBP
01008758    44        INC ESP
01008759    204E 4F   AND BYTE PTR DS:[ESI+4F],CL
0100875C    54        PUSH ESP
0100875D    45        INC EBP
0100875E    50        PUSH EAX
0100875F    41        INC ECX
01008760    44        INC ESP
01008761    0000      ADD BYTE PTR DS:[EAX],AL
01008763    0000      ADD BYTE PTR DS:[EAX],AL
01008765    0000      ADD BYTE PTR DS:[EAX],AL
01008767    0000      ADD BYTE PTR DS:[EAX],AL
01008769    0000      ADD BYTE PTR DS:[EAX],AL
0100876B    00        DB 00
0100876C    00        DB 00
0100876D    00        DB 00
0100876E    00        DB 00
0100876F    00        DB 00
01008770    00        DB 00
```

Press now CTRL+A to reanalyze the code.



```
0100874E    00                DB 00
0100874F    00                DB 00
01008750    00                DB 00
01008751  . 49 4E 4A 45 4:   ASCII "INJECTED NOTEPAD"
01008761  . 00                ASCII 0
01008762    00                DB 00
01008763    00                DB 00
01008764    00                DB 00
01008765    00                DB 00
01008766    00                DB 00
01008767    00                DB 00
01008768    00                DB 00
01008769    00                DB 00
0100876A    00                DB 00
0100876B    00                DB 00
0100876C    00                DB 00
0100876D    00                DB 00
```

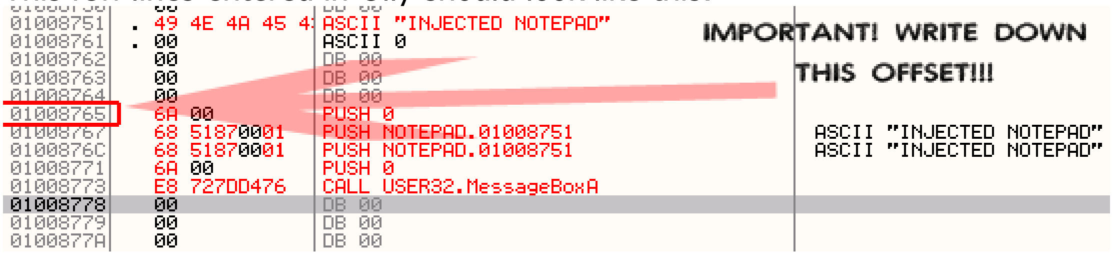
OK! If you want to have a different MessageBox Caption than the title you can repeat this step to make a second ASCII like the "INJECTED NOTEPAD".

-Some ASM-

Now its time for some ASM ;)
We need to invoke a MessageBox from ASM. This is quite simple!

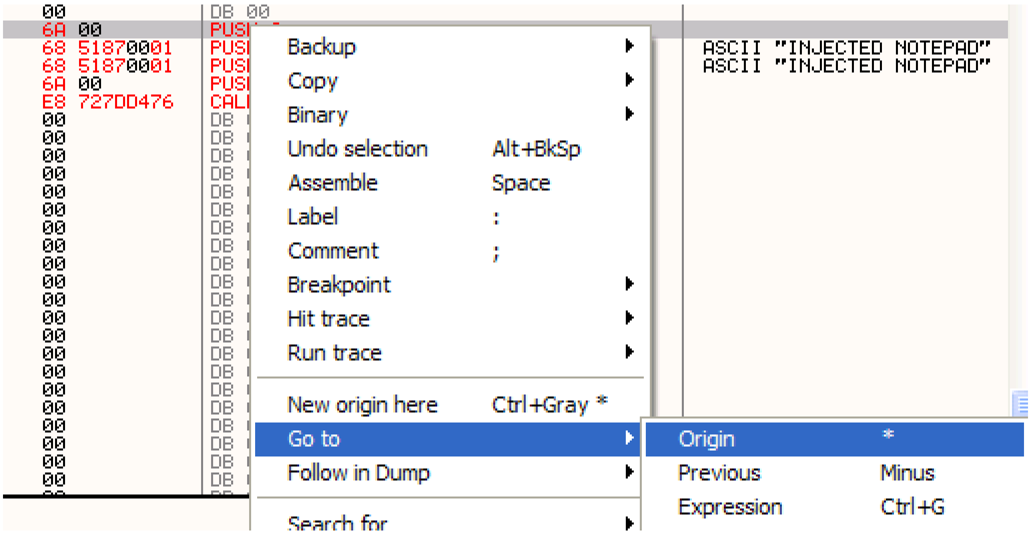
```
PUSH 0             ; BUTTONS = <OK ONLY>
PUSH 1008751       ; CAPTION  = Our adress of the "INJECTED NOTEPAD"
PUSH 1008751       ; MESSAGE  = Same like above.
PUSH 0             ; ICON      = <NO ICON>
CALL MessageBoxA; Run MessageBoxA with the Params above.
```

This few lines entered in Olly should look like this:



```
01008751  . 49 4E 4A 45 4  ASCII "INJECTED NOTEPAD"
01008761  . 00             ASCII 0
01008762    00             DB 00
01008763    00             DB 00
01008764    00             DB 00
01008765    6A 00          PUSH 0
01008767    68 51870001    PUSH NOTEPAD.01008751    ASCII "INJECTED NOTEPAD"
0100876C    68 51870001    PUSH NOTEPAD.01008751    ASCII "INJECTED NOTEPAD"
01008771    6A 00          PUSH 0
01008773    E8 727DD476    CALL USER32.MessageBoxA
01008778    00             DB 00
01008779    00             DB 00
0100877A    00             DB 00
```

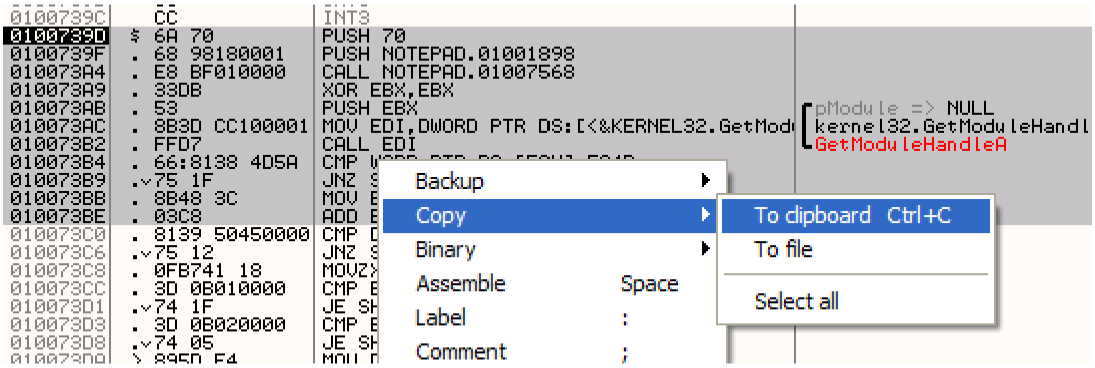
Do you see the arrow?! THIS IS NOW A VERY IMPORTANT STEP!
If we save it now and run it you will NOT see any effect. Why?
Because our litte routine is not called yet!
You need to write down the Offset of your first *"PUSH 0"* because we need to make a jump from the programs origin to here and back again ;)
If you noted the offset of the first PUSH down, goto the origin of the program like below.



```
00             DB 00
6A 00          PUSI
68 51870001    PUSI    ASCII "INJECTED NOTEPAD"
68 51870001    PUSI    ASCII "INJECTED NOTEPAD"
6A 00          PUSI
E8 727DD476    CALI
00             DB
00             DB
00             DB
00             DB
00             DB
00             DB
00             DB
00             DB
00             DB
00             DB
00             DB
00             DB
00             DB
00             DB
00             DB
00             DB
00             DB
00             DB
00             DB
00             DB
00             DB
```

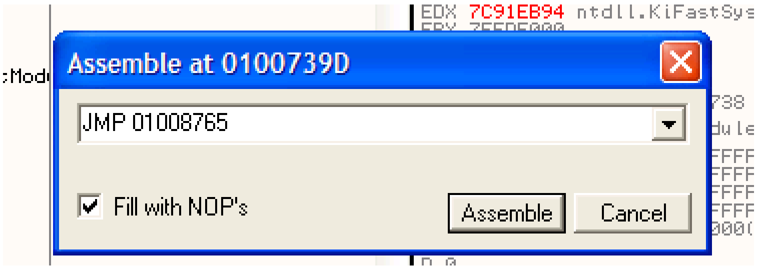
Now you're at the FIRST line of code which will be executed.
Do you remember that the first thing we wanted to do is to run our code? :)
Yehaa! We're on the right way!

Select now from the origin some lines and copy it into the Clipboard.



```
0100739C     CC             INT3
0100739D   $ 6A 70          PUSH 70
0100739F   . 68 98180001    PUSH NOTEPAD.01001898
010073A4   . E8 BF010000    CALL NOTEPAD.01007568
010073A9   . 33DB           XOR EBX,EBX
010073AB   . 53             PUSH EBX                                 [pModule => NULL
010073AC   . 8B3D CC100001  MOV EDI,DWORD PTR DS:[<&KERNEL32.GetMod  |kernel32.GetModuleHandl
010073B2   . FFD7           CALL EDI                                 GetModuleHandleA
010073B4   . 66:8138 4D5A   CMP W
010073B9   .v75 1F          JNZ S
010073BB   . 8B48 3C        MOV E
010073BE   . 03C8           ADD E
010073C0   . 8139 50450000  CMP D
010073C6   .v75 12          JNZ S
010073C8   . 0FB741 18      MOVZX
010073CC   . 3D 0B010000    CMP E
010073D1   .v74 1F          JE SH
010073D3   . 3D 0B020000    CMP E
010073D8   .v74 05          JE SH
```

Paste the clipboard into a texteditor and leave them there. We need this lines later.
If you have pasted it go to the first line of the program.
(In the picture upper it's the *PUSH 70*).
Double click on it and enter in the box "*JMP <adress of your first push>*"

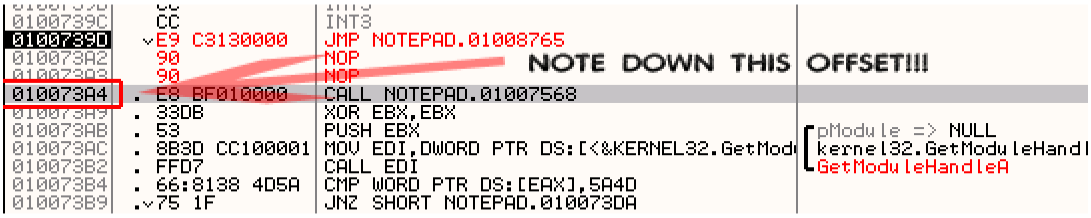


Assemble at 0100739D

JMP 01008765

☑ Fill with NOP's

Press on Assemble and you will the the again the red marked(patched) code.



```
0100739C     CC             INT3
0100739D    vE9 C3130000    JMP NOTEPAD.01008765
010073A2     90             NOP
010073A3     90             NOP
010073A4   . E8 BF010000    CALL NOTEPAD.01007568
010073A9   . 33DB           XOR EBX,EBX
010073AB   . 53             PUSH EBX                                 [pModule => NULL
010073AC   . 8B3D CC100001  MOV EDI,DWORD PTR DS:[<&KERNEL32.GetMod  |kernel32.GetModuleHandl
010073B2   . FFD7           CALL EDI                                 GetModuleHandleA
010073B4   . 66:8138 4D5A   CMP WORD PTR DS:[EAX],5A4D
010073B9   .v75 1F          JNZ SHORT NOTEPAD.010073DA
```

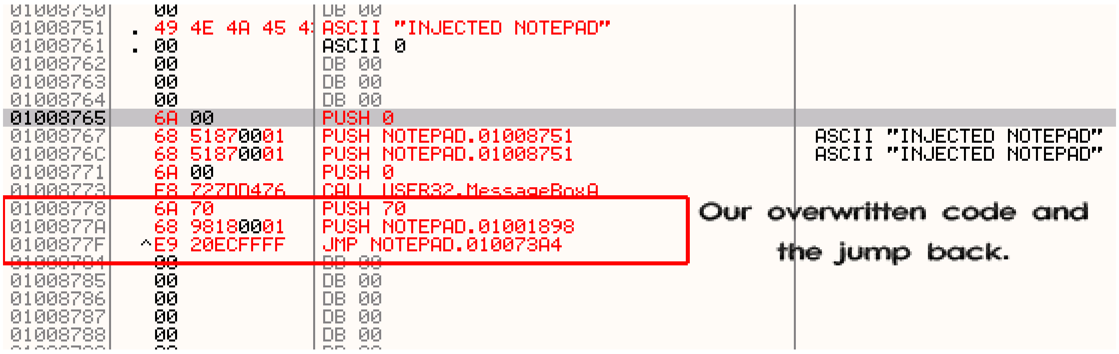
NOTE DOWN THIS OFFSET!!!

Look to the redbox! This is the address we need to jump to after our injected code.
If we jump here after the injection the program will execute like without our injection :)
But there is one thing we need to do at least!
Compare the "new" origin with the old one you've pasted into a clipboard.
You will see that there are a few lines overwritten! But this lines are needed to run the programm without errors. Identify the lines which get overwritten.
In my case the overwritten lines are:

PUSH 70
PUSH NOTEPAD.01001898

Click on the first line (our JMP) and press ENTER.
You'll dropped to your MessageBox invokation!
After our CALL MessageBoxA we need to insert now the overwritten lines AND the jump

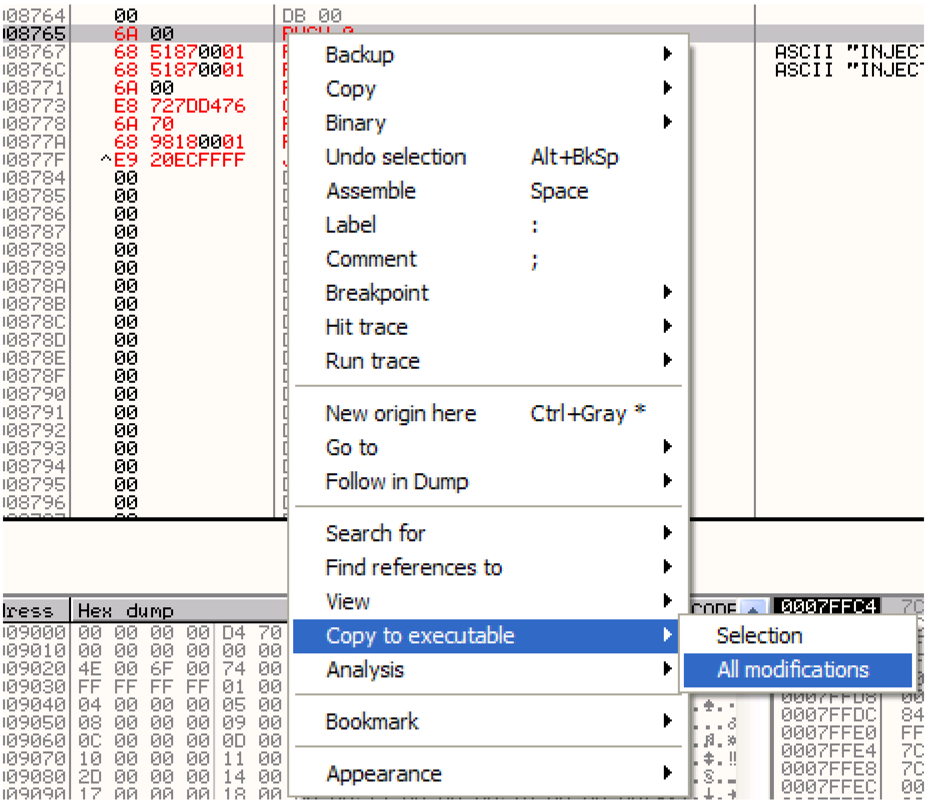
back!



```
01008751  . 49 4E 4A 45 4: ASCII "INJECTED NOTEPAD"
01008761  . 00             ASCII 0
01008762    00             DB 00
01008763    00             DB 00
01008764    00             DB 00
01008765    6A 00          PUSH 0
01008767    68 51870001    PUSH NOTEPAD.01008751            ASCII "INJECTED NOTEPAD"
0100876C    68 51870001    PUSH NOTEPAD.01008751            ASCII "INJECTED NOTEPAD"
01008771    6A 00          PUSH 0
01008773    E8 727DD476    CALL USER32.MessageBoxA
01008778    6A 70          PUSH 70
0100877A    68 98180001    PUSH NOTEPAD.01001898
0100877F   ^E9 20ECFFFF    JMP NOTEPAD.010073A4
01008784    00             DB 00
01008785    00             DB 00
01008786    00             DB 00
01008787    00             DB 00
01008788    00             DB 00
```

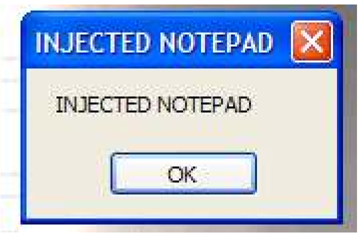
Our overwritten code and the jump back.

NOW you're done!
*(If you want to test the "JumpBack"-Jump just select it and press ENTER.
If you get to the right line you can be sure that's ok! If not check Offset!)*
To save the "new" Notepad take a look the the following picture:



```
08764   00
08765   6A 00
08767   68 51870001      ASCII "INJEC
0876C   68 51870001      ASCII "INJEC
08771   6A 00
08773   E8 727DD476
08778   6A 70
0877A   68 98180001
0877F  ^E9 20ECFFFF
08784   00
```

If you press on "All modifications" a new little window will be shown.
Press on "Copy All" on this window.
A new window with the new ASM code will be shown.
Close the new window (THE CHILD WINDOW! NOT OLLY DBG WINDOW!).
Then a save dialog let you choose a new filename.
Save the file and run it. If you're successful you will get this result:


INJECTED NOTEPAD

INJECTED NOTEPAD

Press on OK and Notepad will start normally ;)

-Last Words-

DO NOT abuse debuggers to attach shellcodes or things like that into EXEs!
Just lern and understand!
Greets to: All students from the FH-BRS!

Iman Karim.