

# **C.8 DEZASAMBLARE SI PATCHING CU X96DBG**

**PAUL A. GAGNIUC**

Academia Tehnică Militară „Ferdinand I”

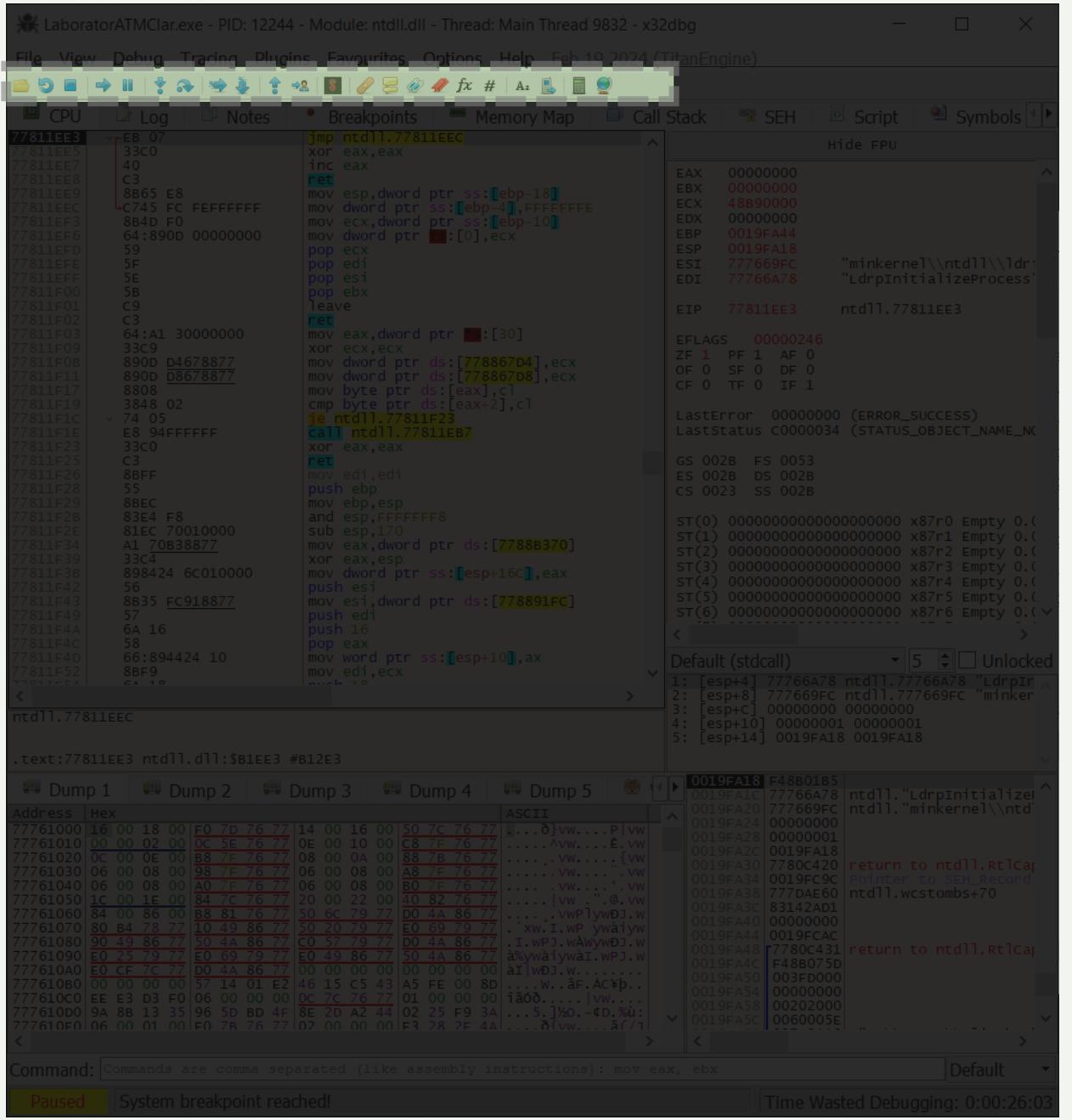


# PRINCIPALELE PĂRȚI ALE PREZENTĂRII

## C.8 Dezasamblare și Patching cu X32dbg / X64dbg:

- C.8.1 IDENTIFICAREA PAROLELOR ASCUNSE IN EXECUTABILE
- C.8.2 ADĂUGAREA DE COD MAȘINĂ
- C.8.3 MODIFICARE ENTRY POINT ȘI INSERTIE DE COD MAȘINĂ
- C.8.4 ADAUGAREA DE STRUCTURI DE DATE IN EXECUTABILE
- C.8.5 MODIFICĂRI DE ADRESE CĂTRE ALTE ARGUMENTE





# X64dbg User Interface

## Play Controls

Bara de meniu oferă acces la opțiuni avansate de configurare, controlul execuției, încărcarea sau salvarea de sesiuni, integrarea de pluginuri și personalizarea mediului de depanare. Rolul ei este de a centraliza și structura toate comenziile disponibile, servind drept punct de plecare pentru orice acțiune pe care utilizatorul dorește să o efectueze asupra executabilului analizat.

LaboratorATMClar.exe - PID: 12244 - Module: ntdll.dll - Thread: Main Thread 9832 - x32dbg

File View Debug Tracing Plugins Favourites Options Help Feb 19 2024 (TitanEngine)

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols

```

77811EE5 33C0 xor eax,eax
77811EE7 40 inc eax
77811EE8 C3 ret
77811EE9 8B65 E8 mov esp,dword ptr ss:[ebp-18]
77811EEC 745 FC FFFFFFFF mov dword ptr ss:[ebp-4],FFFFFFFE
77811EF3 884D F0 mov ecx,dword ptr ss:[ebp-10]
77811EF6 64:890D 00000000 mov dword ptr [0],ecx
77811EFD 59 pop ecx
77811EFF 5F pop edi
77811FF0 5E pop esi
77811FF1 5B pop ebx
77811F00 C9 leave
77811F02 C3 ret
77811F03 64:A1 30000000 mov eax,dword ptr [30]
77811F09 33C9 xor ecx,ecx
7809D D4678877 mov dword ptr ds:[77886704],ecx
77811F11 890D D8678877 mov dword ptr ds:[77886708],ecx
77811F17 8808 mov byte ptr ds:[eax],cl
77811F19 3848 02 cmp byte ptr ds:[eax-2],cl
77811F1C 74 05 je ntdll.77811F23
77811F1E 8844 FFFF mov eax,4FFFFFFF
77811F23 33C0 xor eax,eax
77811F25 C3 ret
77811F26 8BF mov edi,edi
77811F28 55 push ebp
77811F29 8BEC mov esp,esp
77811F2B 83E4 F8 and esp,FFFFFFF8
77811F2E 81EC 70010000 sub esp,170
77811F34 A1 70838877 mov eax,dword ptr ds:[7788B370]
77811F39 33C4 xor eax,esp
77811F3B 898424 6C010000 mov dword ptr ss:[esp+16C],eax
77811F42 56 push esi
77811F43 8B35 FC918877 mov esi,dword ptr ds:[778891FC]
77811F49 57 push edi
77811F4A 6A 16 push 16
77811F4C 58 pop eax
77811F4D 66:894424 10 mov word ptr ss:[esp+10],ax
77811F52 8BF9 mov edi,ecx
77811F5A 54:10
Default (stdcall) 5 Unlocked
1: [esp+4] 77766A78 ntdll.77766A78 "LdrpIn
2: [esp+8] 777669FC ntdll.777669FC "minker
3: [esp+c] 00000000 00000000
4: [esp+10] 00000001 00000001
5: [esp+14] 0019FA18 0019FA18

```

ntdll.77811EEC

.text:77811EE3 ntdll.dll:\$B1EE3 #B12E3

Dump 1 Dump 2 Dump 3 Dump 4 Dump 5

Address	Hex	ASCII
77761000	16 00 18 00	F0 7D 76 77 14 00 16 00 50 7C 76 77 ... .vw....P vw
77761010	00 00 02 00	0E 5E 76 77 0E 00 10 00 C8 7F 76 77 ... .vw....E.vw
77761020	0C 00 00 00	B8 7F 76 77 08 00 0A 00 88 7B 76 77 ... .vw....{vw
77761030	06 00 08 00	98 7E 76 77 06 00 08 00 A8 7E 76 77 ... .vw....vw
77761040	06 00 08 00	A0 7E 76 77 06 00 08 00 B0 7E 76 77 ... .vw....vw
77761050	1C 00 1E 00	B4 7C 76 77 20 00 22 00 40 82 76 77 ... . vw...@.vw
77761060	84 00 86 00	88 81 76 77 50 6C 79 77 D0 4A 86 77 ... .vw!lywDj.w
77761070	80 B4 78 77	10 49 86 77 50 20 79 77 E0 69 79 77 . xw.I.w!wyaiyw
77761080	90 49 86 77	50 4A 86 77 C0 57 79 77 D0 4A 86 77 I.wP1.wAwywDj.w
77761090	E0 25 79 77	E0 69 79 77 E0 49 86 77 50 4A 86 77 a!ywyaiyw!wpJ.w
777610A0	00 CF 7C 77	D0 4A 86 77 00 00 00 00 00 00 00 00 a!wDj.w....
777610B0	00 00 00 00	57 14 01 E2 46 15 C5 43 A5 FE 00 8D ... .w..!F.Ac!p..
777610C0	EE E3 D3 F0	06 00 00 00 0C 76 77 01 00 00 00 f!ao...!vw....
777610D0	9A 8B 13 35	96 50 BD 4F 8E 2D A2 44 02 25 F9 3A ... 5.%o.-CD.%u: ..
777610F0	06 00 01 00	F0 7B 76 77 02 00 00 00 F3 28 2F 4A ... !ivw...!a?/1 ..

Command: Commands are comma separated (like assembly instructions): mov eax, ebx Default

Paused System breakpoint reached!

Time Wasted Debugging: 0:00:26:03

# X64dbg User Interface

## View Tabs

Taburile ofera acces rapid la functiile esentiale ale depanatorului, precum vizualizarea CPU, log-urile de executie, notiile, gestionarea breakpoints, harta de memorie, stiva de apeluri (Call Stack), gestionarea exceptiilor, rularea scripturilor si afisarea simbolurilor.

Practic, aceste taburi controleaza ce tip de informații sunt afisate în fereastra de lucru, iar utilizatorul poate comuta între ele pentru a analiza diferite aspecte ale executabilului. Bara de instrumente are rolul de a centraliza principalele unelte de analiză, oferind o navigare rapidă și structurată prin componentele procesului aflat în execuție.

LaboratorATMClar.exe - PID: 12244 - Module: ntdll.dll - Thread: Main Thread 9832 - x32dbg

File View Debug Tracing Plugins Favourites Options Help Feb 19 2024 (TitanEngine)

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Hide FPU

```

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Hide FPU

77811EE3 EB 07 jmp ntdll.77811EEC
77811EE5 33C0 xor eax, eax
77811EE7 40 inc eax
77811EE8 C3 ret
77811EE9 8B65 E8 mov esp, dword ptr ss:[ebp-18]
77811EEC 40C7 44 FF mov dword ptr ss:[ebp-4], FFFFFFFE
77811EF3 884D F0 mov ecx, dword ptr ss:[ebp-10]
77811EF6 64:890D 00000000 mov dword ptr ss:[0], ecx
77811EF9 59 pop ecx
77811F00 5F pop edi
77811F01 5E pop esi
77811F02 5B pop ebx
77811F03 C9 leave
77811F04 C3 ret
77811F05 64:A1 30000000 mov eax, dword ptr ss:[30]
77811F09 33C9 xor ecx, ecx
77811F0B 890D D4678877 mov dword ptr ds:[778867D4], ecx
77811F11 890D D8678877 mov dword ptr ds:[778867D8], ecx
77811F17 8808 mov byte ptr ds:[eax], cl
77811F19 3848 02 cmp byte ptr ds:[eax+2], cl
77811F1C 74 05 je ntdll.77811F23
77811F1E E8 FFFFFFFF call ntdll.77811EB7
77811F23 33C0 xor eax, eax
77811F25 C3 ret
77811F26 8BF mov edi, edi
77811F28 55 push ebp
77811F29 8BEC mov ebp, esp
77811F2B 83E4 F8 and esp, FFFFFFF8
77811F2E 81EC 70010000 sub esp, 170
77811F34 A1 70B38877 mov eax, dword ptr ds:[7788B370]
77811F38 33C4 xor eax, esp
77811F3B 898424 6C010000 mov dword ptr ss:[esp+16C], eax
77811F42 56 push esi
77811F43 8835 FC918877 mov esi, dword ptr ds:[778891FC]
77811F49 57 push edi
77811F4A 6A 16 push 16
77811F4C 58 pop eax
77811F4D 66:894424 10 mov word ptr ss:[esp+10], ax
77811F52 8BF9 mov edi, ecx
    
```

LastError 00000000 (ERROR\_SUCCESS)  
LastStatus 00000034 (STATUS\_OBJECT\_NAME\_NK)

GS 002B FS 0053  
ES 002B DS 002B  
CS 0023 SS 002B

ST(0) 00000000000000000000000000000000 x87r0 Empty 0.(  
ST(1) 00000000000000000000000000000000 x87r1 Empty 0.(  
ST(2) 00000000000000000000000000000000 x87r2 Empty 0.(  
ST(3) 00000000000000000000000000000000 x87r3 Empty 0.(  
ST(4) 00000000000000000000000000000000 x87r4 Empty 0.(  
ST(5) 00000000000000000000000000000000 x87r5 Empty 0.(  
ST(6) 00000000000000000000000000000000 x87r6 Empty 0.(

Default (stdcall) 5 Unlocked

0019FA18 F48B01B5 0019FA1C 77766A78 ntdll.77766A78 "LdrpIn...  
0019FA20 777669FC ntdll.777669FC "minker...  
0019FA24 00000000  
0019FA28 00000001  
0019FA2C 0019FA18  
0019FA30 7780C420 return to ntdll.RtlCaj...  
0019FA34 0019FC9C Pointer to SEH\_Record  
0019FA38 777DAE60 ntdll.wcstombs+70  
0019FA3C 83142AD1  
0019FA40 00000000  
0019FA44 0019FCAC  
0019FA48 7780C431 return to ntdll.RtlCaj...  
0019FA4C F48B075D  
0019FA50 003FD000  
0019FA54 00000000  
0019FA58 00202000  
0019FA5C 0060005E

Dump 1 Dump 2 Dump 3 Dump 4 Dump 5

Address	Hex	ASCII
77761000	16 00 18 00	F0 7D 76 77 14 00 16 00 50 7C 76 77 ... .vv....P vv
77761010	00 00 02 00	0E 5E 76 77 0E 00 10 00 C8 7F 76 77 ... .vv....E vv
77761020	0C 00 00 00	B8 7F 76 77 08 00 0A 00 88 7F 76 77 ... .vv....{vv
77761030	06 00 08 00	98 7F 76 77 06 00 08 00 A8 7F 76 77 ... .vv.... vv
77761040	06 00 08 00	A0 7F 76 77 06 00 08 00 B0 7F 76 77 ... .vv.... .vv
77761050	1C 00 1E 00	B4 7C 76 77 20 00 22 00 40 82 76 77 ...  vv ..@.vv
77761060	84 00 86 00	88 81 76 77 50 6C 79 77 D0 4A 86 77 ... .vv lywDJ.w
77761070	80 B4 78 77	10 49 86 77 50 20 79 77 E0 69 79 77 ... .xw.I.w wyaiyw
77761080	90 49 86 77	50 4A 86 77 C0 57 79 77 D0 4A 86 77 ... I.wP1.wAwywDJ.w
77761090	E0 25 79 77	E0 69 79 77 E0 49 86 77 50 4A 86 77 ... ayywaiywJ.wPj.w
777610A0	00 CF 7C 77	D0 4A 86 77 00 00 00 00 00 00 00 00 ... a wDj.w.....
777610B0	00 00 00 00	57 14 01 E2 46 15 C5 43 A5 FE 00 8D ... .w..âF.AC@p..
777610C0	EE E3 D3 F0	06 00 00 00 0C 76 77 01 00 00 00 00 00 ... iâoâ... vv..
777610D0	9A 8B 13 35	96 50 BD 4F 8E 2D A2 44 02 25 F9 3A ... .5.%0.-CD.%%;
777610F0	06 00 01 00	F0 7B 76 77 02 00 00 00 F3 28 2F 4A ... .â vw...â/

Command: Commands are comma separated (like assembly instructions): mov eax, ebx Default

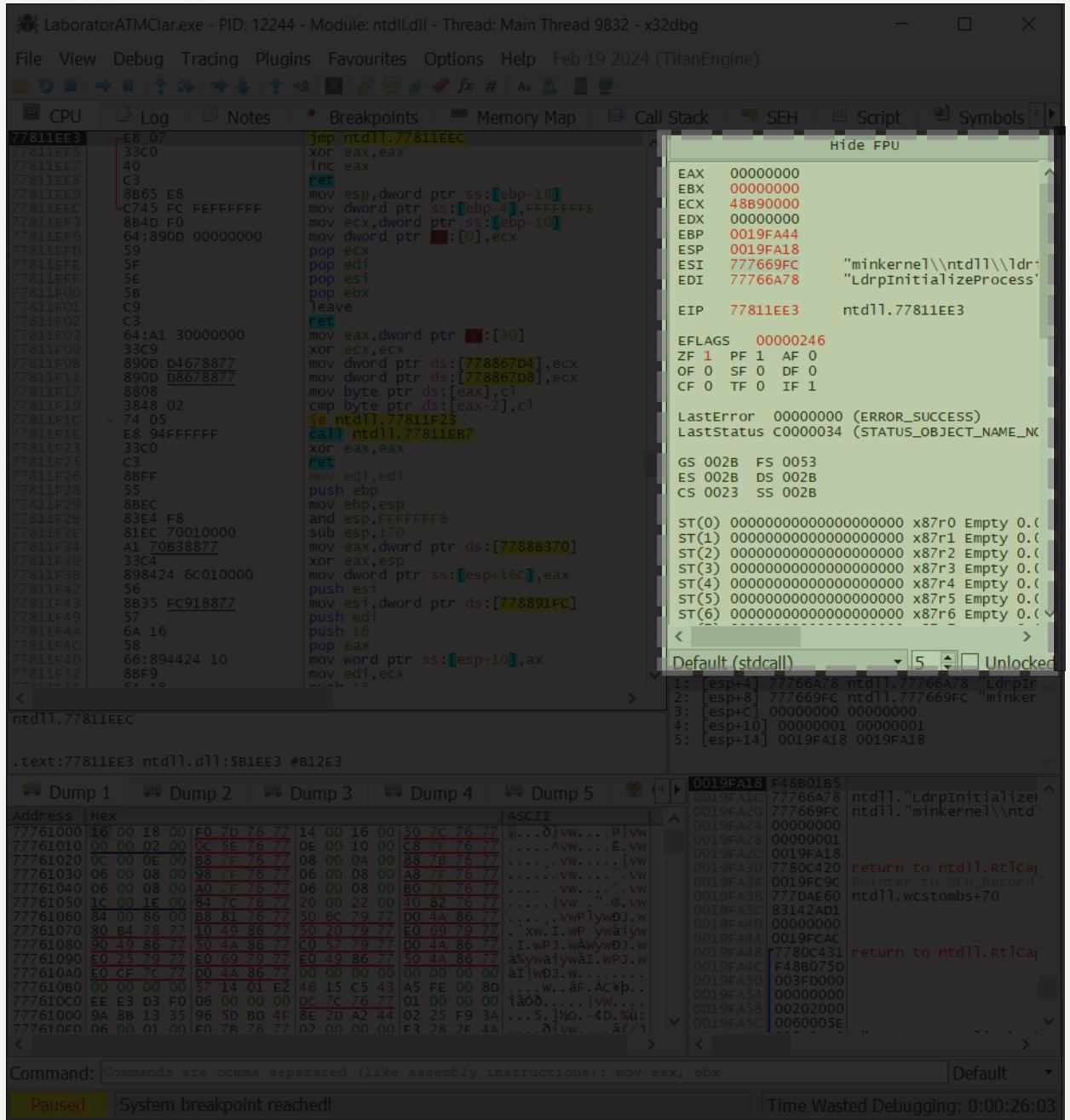
Paused System breakpoint reached!

Time Wasted Debugging: 0:00:26:03

# X64dbg User Interface

## CPU Assembly

Aici se află fereastra principală de disassembly. Rolul ei este să arate instrucțiunile pe care procesorul urmează să le execute, traduse din cod mașină în limbaj de asamblare. Utilizatorul poate urmări pas cu pas cum se execută programul, poate seta breakpoints, poate vedea fluxul de execuție și apelurile de funcții. Practic, această fereastră este „inima” analizării unui executabil, deoarece afișează direct logica internă a programului aşa cum este interpretată de procesor.



# X64dbg User Interface

# Registers

Rolul acestei zone este să ofere o imagine completă asupra execuției programului la nivel de procesor. Aici se văd valorile regiștrilor și ale flag-urilor procesorului, care se actualizează după fiecare instrucțiune executată. În partea de jos sunt afișate informații despre stivă, inclusiv adresele de return și parametrii funcțiilor. Practic, aceasta zonă este nucleul analizei: combină instrucțiunile, starea regiștrilor și conținutul stivei pentru a oferi utilizatorului control și vizibilitate totală asupra execuției unui fișier executabil.

## Legenda:

- **EAX:** Math and Return Values
  - **EBX:** Base index (for use with arrays)
  - **ECX:** Counter
  - **EDX:** Math
  - **EBP:** Base Pointer
  - **ESP:** Stack Pointer
  - **ESI/EDI:** Memory Transfer

- **CS (Code Segment)** indică segmentul unde se află codul executabil.
- **DS (Data Segment)** segmentul implicit pentru date.
- **SS (Stack Segment)** segmentul folosit pentru stivă.
- **ES (Extra Segment)** segment suplimentar pentru operații pe date.
- **FS / GS** registrii de segment speciali, folositi de sistemul de operare pentru structuri interne (în Windows, FS și GS sunt utilizate pentru Thread Environment Block (TEB) și alte date legate de thread/proces).

**ZF (Zero Flag)** setat la 1 dacă rezultatul unei operații este zero.

**PF (Parity Flag)** indică dacă rezultatul are un număr par de biți setați la 1.

**AF (Adjust Flag)** folosit pentru operații aritmetice pe 4 biți (ex. BCD).

**OF (Overflow Flag)** semnalează depășirea limitei la operații pe numere semnate.

**SF (Sign Flag)** reflectă semnul rezultatului (1 = negativ).

**DF (Direction Flag)** determină direcția operațiilor pe șiruri (0 = înainte, 1 = înapoi).

**CF (Carry Flag)** semnalează un transport sau un împrumut la operații aritmetice.

**TF (Trap Flag)** dacă e setat, activează execuția pas-cu-pas (debugging).

**IF (Interrupt Flag)** controlează dacă intreruperile hardware sunt permise (1 = activat).

LaboratorATMClar.exe - PID: 12244 - Module: ntdll.dll - Thread: Main Thread 9832 - x32dbg

File View Debug Tracing Plugins Favourites Options Help Feb 19 2024 (TitanEngine)

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols

```

77811EE3 >EB 07 jmp ntdll.77811EEC
77811EE5 33C0 xor eax,eax
77811EE7 40 inc eax
77811EE8 C3 ret
77811EE9 8865 E8 mov esp,dword ptr ss:[ebp-18]
77811EEC 40 mov dword ptr ss:[ebp-4],FFFFFFFE
77811EF3 884D F0 mov ecx,dword ptr ss:[ebp-10]
77811EF6 64:890D 00000000 mov dword ptr [0],ecx
77811EFD 59 pop ecx
77811EFF 5F pop edi
77811F00 5E pop esi
77811F01 5B pop ebx
77811F02 C9 leave
77811F03 C3 ret
77811F04 64:A1 30000000 mov eax,dword ptr [30]
77811F09 33C9 xor ecx,ecx
77811F0B 890D D4678877 mov dword ptr ds:[77886704],ecx
77811F11 890D D8678877 mov dword ptr ds:[77886708],ecx
77811F17 8808 mov byte ptr ds:[eax],cl
77811F19 3848 02 cmp byte ptr ds:[eax-2],cl
77811F1C 74 05 je ntdll.77811F23
77811F1E E8 94FFFFFF call ntdll.77811E87
77811F23 33C0 xor eax,eax
77811F25 C3 ret
77811F26 88FF mov edi,edi
77811F28 55 push ebp
77811F29 8BEC mov ebp,esp
77811F2B 83E4 F8 and esp,FFFFFFF8
77811F2E 81EC 70010000 sub esp,170
77811F34 A1 70838877 mov eax,dword ptr ds:[7788B370]
77811F39 33C4 xor eax,esp
77811F3B 898424 6C010000 mov dword ptr ss:[esp+16C],eax
77811F42 56 push esi
77811F43 8B35 FC918877 mov esi,dword ptr ds:[778891FC]
77811F49 57 push edi
77811F4A 6A 16 push 16
77811F4C 58 pop eax
77811F4D 66:894424 10 mov word ptr ss:[esp+10],ax
77811F52 88F9 mov edi,ecx
77811F5A 54:10

.text:77811EE3 ntdll.dll:$B1EE3 #B12E3

Default (stdcall) 5 Unlocked
1: [esp+4] 77766A78 ntdll.77766A78 "LdrpIn
2: [esp+8] 777669FC ntdll.777669FC "minker
3: [esp+c] 00000000 00000000
4: [esp+10] 00000001 00000001
5: [esp+14] 0019FA18 0019FA18

```

Dump 1 Dump 2 Dump 3 Dump 4 Dump 5

Address	Hex	ASCII
77761000	16 00 18 00	F0 7D 76 77 14 00 16 00 50 7C 76 77 ...0.vw...P vw
77761010	00 00 02 00	0E 5E 76 77 0E 00 10 00 C8 7F 76 77 ...^vw...E.vw
77761020	0C 00 00 00	B8 7F 76 77 08 00 0A 00 88 7B 76 77 ...vw...{vw
77761030	06 00 08 00	98 70 76 77 06 00 08 00 A8 7E 76 77 ...vw...~vw
77761040	06 00 08 00	A0 7E 76 77 06 00 08 00 B0 7E 76 77 ...vw...~.vw
77761050	1C 00 1E 00	B4 7C 76 77 20 00 22 00 40 82 76 77 ... vw...@.vw
77761060	84 00 86 00	88 81 76 77 50 6C 79 77 D0 4A 86 77 ...vwlywBj.w
77761070	80 B4 78 77	10 49 86 77 50 20 79 77 E0 69 79 77 .xw.I.wyaiyw
77761080	90 49 86 77	50 4A 86 77 C0 57 79 77 D0 4A 86 77 I.wP1.wAwywBj.w
77761090	E0 25 79 77	E0 69 79 77 E0 49 86 77 a\wyaiyw\I.wPj.w
777610A0	00 CF 7C 77	D0 4A 86 77 00 00 00 00 00 00 00 00 a\wDj.w....
777610B0	00 00 00 00	57 14 01 E2 46 15 C5 A3 F0 FE 08 8D ...w..^F.Ac\p..
777610C0	EE E3 D3 F0	06 00 00 00 0C 76 77 01 00 00 00 f\ao\... vw...^/1
777610D0	9A 8B 13 35	96 50 BD 4F 8E 2D A2 44 02 25 F9 3A ...5.%0.-CD.%:
777610F0	06 00 01 00	F0 7B 76 77 02 00 00 00 F3 28 2F 4A ...^ivw...^/1

Command: Commands are comma separated (like assembly instructions): mov eax, ebx Default

Paused System breakpoint reached!

Time Wasted Debugging: 0:00:26:03

# X64dbg User Interface

## Stack Window

### Conținut brut al stivei (argumente, valori locale)

Această fereastră afișează conținutul stivei procesului, adică zona de memorie unde sunt plasate adrese de return, variabile locale și parametrii funcțiilor apelate. Fiecare linie corespunde unei adrese din stivă, iar valorile afișate permit urmărirea succesiunii apelurilor și a modului în care programul gestionează execuția funcțiilor. În analiza unui executabil, fereastra Stack este esențială pentru a înțelege cum se transmit parametrii, cum se realizează salturile în cod și pentru a identifica eventuale suprascrieri de memorie sau exploatari de tip buffer overflow.

LaboratorATMClar.exe - PID: 12244 - Module: ntdll.dll - Thread: Main Thread 9832 - x32dbg

File View Debug Tracing Plugins Favourites Options Help Feb 19 2024 (TitanEngine)

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols

```

77811EE3 >EB 07 jmp ntdll.77811EEC
77811EE5 33C0 xor eax,eax
77811EE7 40 inc eax
77811EE8 C3 ret
77811EE9 8B65 E8 mov esp,dword ptr ss:[ebp-18]
77811EEC 7C45 FC FFFFFFFF mov dword ptr ss:[ebp-4],FFFFFFFE
77811EF3 8B4D F0 mov ecx,dword ptr ss:[ebp-10]
77811EF6 64:890D 00000000 mov dword ptr [0],ecx
77811EFD 59 pop ecx
77811EFF 5F pop edi
77811FFF 5E pop esi
77811F00 5B pop ebx
77811F01 C9 leave
77811F02 C3 ret
77811F03 64:A1 30000000 mov eax,dword ptr [30]
77811F09 33C9 xor ecx,ecx
77811F0B 890D D4678877 mov dword ptr ds:[77886704],ecx
77811F11 890D D8678877 mov dword ptr ds:[77886708],ecx
77811F17 8808 mov byte ptr ds:[eax],cl
77811F19 3848 02 cmp byte ptr ds:[eax-2],cl
77811F1C 74 05 je ntdll.77811F23
77811F1E E8 94FFFFFF call ntdll.77811E87
77811F23 33C0 xor eax,eax
77811F25 C3 ret
77811F26 8BF7 mov edi,edi
77811F28 55 push ebp
77811F29 8BEC mov esp,esp
77811F2B 83E4 F8 and esp,FFFFFFF8
77811F2E 81EC 70010000 sub esp,170
77811F34 A1 70B38877 mov eax,dword ptr ds:[7788B370]
77811F39 33C4 xor eax,esp
77811F3B 898424 6C010000 mov dword ptr ss:[esp+16C],eax
77811F42 56 push esi
77811F43 8B35 FC918877 mov esi,dword ptr ds:[778891FC]
77811F49 57 push edi
77811F4A 6A 16 push 16
77811F4C 58 pop eax
77811F4D 66:894424 10 mov word ptr ss:[esp+10],ax
77811F52 8B9F pop edi,ecx
77811F5A 54:10
Default (stdcall) 5 Unlocked
1: [esp+4] 77766A78 ntdll.77766A78 "LdrpIn
2: [esp+8] 777669FC ntdll.777669FC "minker
3: [esp+c] 00000000 00000000
4: [esp+10] 00000001 00000001
5: [esp+14] 0019FA18 0019FA18

```

ntdll.77811EEC

.text:77811EE3 ntdll.dll:\$B1EE3 #B12E3

Dump 1 Dump 2 Dump 3 Dump 4 Dump 5

Address	Hex	ASCII
77761000	16 00 18 00	F0 7D 76 77 14 00 16 00 50 7C 76 77 ...0.vw...P vw
77761010	00 00 02 00	0C 5E 76 77 0E 00 10 00 C8 7F 76 77 ...^vw...E.vw
77761020	0C 00 00 00	B8 7F 76 77 08 00 0A 00 88 7B 76 77 ...vw...{vw
77761030	06 00 08 00	98 7F 76 77 06 00 08 00 A8 7E 76 77 ...vw...~.vw
77761040	06 00 08 00	A0 7E 76 77 06 00 08 00 B0 7F 76 77 ...vw...~.vw
77761050	1C 00 1E 00	B4 7C 76 77 20 00 22 00 40 82 76 77 ... vw...@.vw
77761060	84 00 86 00	88 81 76 77 50 6C 79 77 D0 4A 86 77 ...vwlywD.w
77761070	80 B4 78 77	10 49 86 77 50 20 79 77 E0 69 79 77 ...xw.I.wyaiyw
77761080	90 49 86 77	50 4A 86 77 C0 57 79 77 D0 4A 86 77 I.wP1.wAwywD.w
77761090	E0 25 79 77	E0 69 79 77 E0 49 86 77 aSwyaiywI.wPj.w
777610A0	E0 CF 7C 77	D0 4A 86 77 00 00 00 00 00 00 00 00 aI wDj.w...
777610B0	00 00 00 00	57 14 01 E2 46 15 C5 43 A5 FE 00 8D ...w..âF.AcP..
777610C0	EE E3 D3 F0	06 00 00 00 0C 76 77 01 00 00 00 iâoâ... vw...
777610D0	9A 8B 13 35	96 50 BD 4F 8E 2D A2 44 02 25 F9 3A ...5.%0.-CD.%;
777610F0	06 00 01 00	F0 7B 76 77 02 00 00 00 F3 28 2F 4A ...ñvw...â7/1

Command: Commands are comma separated (like assembly instructions): mov eax, ebx Default

Paused System breakpoint reached!

Time Wasted Debugging: 0:00:26:03

# X64dbg User Interface

## Call Stack

Lista apelurilor de funcții și adresele de return.

Această zonă arată succesiunea apelurilor de funcții efectuate până la punctul curent de execuție. Fiecare linie indică o adresă de return și, atunci când este posibil, numele funcției sau modulului asociat. Call Stack-ul este util pentru a înțelege contextul în care se află programul, ordinea apelurilor și pentru a reconstrui traseul execuției. În analiza executabilelor, această vizualizare este esențială pentru depistarea funcțiilor critice, urmărirea fluxului logic și identificarea eventualelor vulnerabilități ce țin de gestionarea apelurilor și a memoriei.

LaboratorATMClar.exe - PID: 12244 - Module: ntdll.dll - Thread: Main Thread 9832 - x32dbg

File View Debug Tracing Plugins Favourites Options Help Feb 19 2024 (TitanEngine)

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols

```

77811EE3 >EB 07 jmp ntdll.77811EEC
77811EE5 33C0 xor eax,eax
77811EE7 40 inc eax
77811EE8 C3 ret
77811EE9 8865 E8 mov esp,dword ptr ss:[ebp-18]
77811EEC >C745 FC FFFFFFFF mov dword ptr ss:[ebp-4],FFFFFFFE
77811EF3 884D F0 mov ecx,dword ptr ss:[ebp-10]
77811EF6 64:890D 00000000 mov dword ptr [0],ecx
77811EFD 59 pop ecx
77811EFF 5F pop edi
77811F00 5E pop esi
77811F01 5B pop ebx
77811F02 C9 leave
77811F03 C3 ret
77811F04 64:A1 30000000 mov eax,dword ptr [30]
77811F09 33C9 xor ecx,ecx
77811F0B 890D D4678877 mov dword ptr ds:[77886704],ecx
77811F11 890D D8678877 mov dword ptr ds:[77886708],ecx
77811F17 8808 mov byte ptr ds:[eax],cl
77811F19 3848 02 cmp byte ptr ds:[eax-2],cl
77811F1C > 74 05 je ntdll.77811F23
77811F1E E8 94FFFFFF call ntdll.77811E87
77811F23 33C0 xor eax,eax
77811F25 C3 ret
77811F26 88FF mov edi,edi
77811F28 55 push ebp
77811F29 8BEC mov esp,esp
77811F2B 83E4 F8 and esp,FFFFFFF8
77811F2E 81EC 70010000 sub esp,170
77811F34 A1 70B38877 mov eax,dword ptr ds:[7788B370]
77811F39 33C4 xor eax,esp
77811F3B 898424 6C010000 mov dword ptr ss:[esp+16C],eax
77811F42 56 push esi
77811F43 8B35 FC918877 mov esi,dword ptr ds:[778891FC]
77811F49 57 push edi
77811F4A 6A 16 push 16
77811F4C 58 pop eax
77811F4D 66:894424 10 mov word ptr ss:[esp+10],ax
77811F52 88F9 mov edi,ecx
77811F5A < 10
Default (stdcall) 5 Unlocked
1: [esp+4] 77766A78 ntdll.77766A78 "LdrpIn
2: [esp+8] 777669FC ntdll.777669FC "minker
3: [esp+c] 00000000 00000000
4: [esp+10] 00000001 00000001
5: [esp+14] 0019FA18 0019FA18

```

ntdll.77811EEC

.text:77811EE3 ntdll.dll:\$B1EE3 #B12E3

Dump 1 Dump 2 Dump 3 Dump 4 Dump 5

Address	Hex	ASCII
77761000	16 00 18 00	F0 7D 76 77 14 00 16 00 50 7C 76 77
77761010	00 00 02 00	0C 5E 76 77 0E 10 00 C8 7F 76 77
77761020	0C 00 0E 00	88 7F 76 77 08 00 0A 00 88 7B 76 77
77761030	06 00 08 00	98 7F 76 77 06 00 08 00 A8 7F 76 77
77761040	06 00 08 00	A0 7F 76 77 06 00 08 00 B0 7F 76 77
77761050	1C 00 1E 00	84 7C 76 77 20 00 22 00 40 82 76 77
77761060	84 00 86 00	88 81 76 77 50 6C 79 77 D0 4A 86 77
77761070	80 B4 78 77	50 20 79 77 E0 69 79 77
77761080	90 49 86 77	50 4A 86 77 C0 57 79 77 D0 4A 86 77
77761090	E0 25 79 77	E0 69 79 77 E0 49 86 77
777610A0	00 CF 7C 77	50 4A 86 77 00 00 00 00 00 00 00 00
777610B0	00 00 00 00	57 14 01 E2 46 15 C5 A3 F0 FE 08 8D
777610C0	EE E3 D3 F0	02 25 F9 3A
777610D0	06 00 00 00	00 00 00 00 F3 28 2F 4A
777610F0	9A 8B 13 35	...5.%0.-%D.%%;

Command: Commands are comma separated (like assembly instructions): mov eax, ebx Default

Paused System breakpoint reached!

Time Wasted Debugging: 0:00:26:03

# X64dbg User Interface

## Dumps (Heap)

**Zona de Dump, este cea mai importantă zonă pentru inginerie inversă!** Aici se afișează conținutul memoriei procesului sub două forme: în stânga valorile hexazecimale, iar în dreapta echivalentul ASCII acolo unde este posibil. Fereastra permite utilizatorului să vizualizeze și să inspecteze datele încărcate în anumite adrese, cum ar fi siruri de caractere, pointeri, structuri sau instrucțiuni codificate. În analiza unui executabil, Dump-ul este esențial pentru a observa direct cum sunt stocate și manipulate datele în memorie, fiind un instrument indispensabil în ingineria inversă și detectarea tehniciilor de obfuscare sau criptare.

**c.8.1**

# **IDENTIFICAREA PAROLELOR ASCUNSE IN EXECUTABILE**



# APLICAȚIE PROTEJATĂ PRIN PAROLĂ

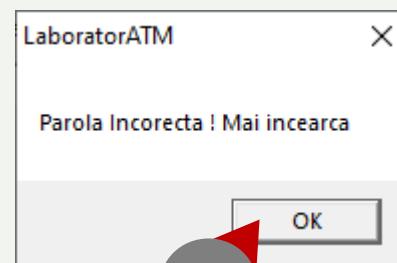
## CE VEDEM CÂND EXECUTĂM FIȘIERUL ȘI CUM FUNCȚIONEAZĂ?



Visual Basic 6.0 este RAD, adică Rapid Application Development)



Ce se află în spatele aplicației compilate?



1

Sursa executabilului:

```
Private secret As String

Private Sub Buton_Click()
    If (InputParola.Text = secret) Then
        MsgBox "Parola corecta !"
    Else
        MsgBox "Parola Incorecta ! Mai incarca"
    End If
End Sub

Private Sub Form_Load()
    secret = "atm2024"
End Sub
```

6

# X32dbg ne setează punctul de intrare initial (nu cel din .text):

LaboratorATMClar.exe - PID: 12244 - Module: ntdll.dll - Thread: Main Thread 9832 - x32dbg

File View Debug Tracing Plugins Favourites Options Help Feb 19 2024 (TitanEngine)

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols

**0**

```

77811EE3 EB 07 jmp ntdll.77811EEC
77811EE5 33C0 xor eax, eax
77811EE7 40 inc eax
77811EE8 C3 ret
77811EE9 8B65 E8 mov esp, dword ptr ss:[ebp-18]
77811EEA C745 FC FE mov dword ptr ss:[ebp-4], FFFFFFFF
77811EEB 884D F0 mov ecx, dword ptr ss:[ebp-10]
77811EED 64:890D 00000000 mov dword ptr [0], ecx
77811EEF 59 pop ecx
77811EFF 5E pop edi
77811FF0 5B pop esi
77811FF1 58 pop ebx
77811FF2 C9 leave
77811FF3 C3 ret
77811FF4 64:A1 30000000 mov eax, dword ptr [30]
77811FF5 33C9 xor ecx, ecx
77811FF6 890D D4678877 mov dword ptr ds:[778867D4], ecx
77811FF7 890D D8678877 mov dword ptr ds:[778867D8], ecx
77811FF8 8808 mov byte ptr ds:[eax], cl
77811FF9 3848 02 cmp byte ptr ds:[eax+2], cl
77811FFA 74 05 je ntdll.77811F23
77811FFB E8 94FFFFFF call ntdll.77811EB7
77811FFC 33C0 xor eax, eax
77811FFD C3 ret
77811FFE 8BFF mov edi, edi
77811FFF 55 push ebp
7781200 8BEC mov ebp, esp
7781201 83E4 F8 and esp, FFFFFFF8
7781202 81EC 70010000 sub esp, 170
7781203 A1 70B38877 mov eax, dword ptr ds:[7788B370]
7781204 33C4 xor eax, esp
7781205 898424 6C010000 mov dword ptr ss:[esp+16C], eax
7781206 56 push esi
7781207 8B35 FC918877 mov esi, dword ptr ds:[778891FC]
7781208 57 push edi
7781209 6A 16 push 16
778120A 58 pop eax
778120B 66:894424 10 mov word ptr ss:[esp+10], ax
778120C 8BF9 mov edi, ecx
778120D 6A 10

```

LastError 00000000 (ERROR\_SUCCESS)  
LastStatus C0000034 (STATUS\_OBJECT\_NAME\_NC)

GS 002B FS 0053  
ES 002B DS 002B  
CS 0023 SS 002B

ST(0) 00000000000000000000000000000000 x87r0 Empty 0.  
ST(1) 00000000000000000000000000000000 x87r1 Empty 0.  
ST(2) 00000000000000000000000000000000 x87r2 Empty 0.  
ST(3) 00000000000000000000000000000000 x87r3 Empty 0.  
ST(4) 00000000000000000000000000000000 x87r4 Empty 0.  
ST(5) 00000000000000000000000000000000 x87r5 Empty 0.  
ST(6) 00000000000000000000000000000000 x87r6 Empty 0.

Default (stdcall) 5 Unlocked

1: [esp+4] 77766A78 ntdll.77766A78 "LdrpIn...  
2: [esp+8] 777669FC ntdll.777669FC "minker...  
3: [esp+c] 00000000 00000000  
4: [esp+10] 00000001 00000001  
5: [esp+14] 0019FA18 0019FA18

ntdll.77811EEC

.text:77811EE3 ntdll.dll:\$B1EE3 #B12E3

Dump 1 Dump 2 Dump 3 Dump 4 Dump 5

Address	Hex	ASCII
77761000	16 00 18 00 F0 7D 76 77	....vw....P vw
77761010	00 00 02 00 0C 5E 76 77	....vw....E vw
77761020	0C 00 08 00 B8 7F 76 77	....vw....[vw
77761030	06 00 08 00 98 7F 76 77	....vw....vw
77761040	06 00 08 00 A0 7F 76 77	....vw....vw
77761050	1C 00 1E 00 84 7C 76 77	....vw....@.vw
77761060	84 00 86 00 B8 81 76 77	....vwPl wD .w
77761070	80 B4 78 77 10 49 86 77	....vwI.wP ywD .w
77761080	90 49 86 77 50 4A 86 77	I.wPJ.wAwvD .w
77761090	E0 25 79 77 E0 69 79 77	E0 49 86 77 %yw wvA w.P w
777610A0	E0 CF 7C 77 D0 4A 86 77	E0 49 86 77 %yw wvA w.P w
777610B0	00 00 00 00 57 14 01 E2	00 00 00 00 00 00 00 00
777610C0	46 15 C5 43 AF FE 00 8D	....W..AF.AC%p.
777610D0	EE E3 D3 F0 06 00 00 0C	iāōđ. .... vw
777610E0	9A 88 13 35 96 5D BD 4F	....5.%o.-CD.%ü:
777610F0	8E 2D A2 44 02 25 F9 3A	....%f. .... vw

Command: Commands are comma separated (like assembly instructions): mov eax, ebx Default

Paused System breakpoint reached! Time Wasted Debugging: 0:00:26:03

# Căutăm siruri în executabil:

LaboratorATMClar.exe - PID: 12244 - Module: ntdll.dll - Thread: Main Thread 9832 - x32dbg

File View Debug Tracing Plugins Favourites Options Help Feb 19 2024 (TitanEngine)

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols

**1**

```

77811EE3 EB 07 jmp ntdll.77811EEC
77811EE5 33C0 xor eax, eax
77811EE7 40 inc eax
77811EE8 C3 ret
77811EE9 8B65 E8 mov esp, dword ptr ss:[ebp-18]
77811EEA C745 FC FE mov dword ptr ss:[ebp-4], FFFFFFFF
77811EEB 884D F0 mov ecx, dword ptr ss:[ebp-10]
77811EED 64:890D 00000000 mov dword ptr [0], ecx
77811EEF 59 pop ecx
77811EFF 5E pop edi
77811FF0 5B pop esi
77811FF1 58 pop ebx
77811FF2 C9 leave
77811FF3 C3 ret
77811FF4 64:A1 30000000 mov eax, dword ptr [30]
77811FF5 33C9 xor ecx, ecx
77811FF6 890D D4678877 mov dword ptr ds:[778867D4], ecx
77811FF7 890D D8678877 mov dword ptr ds:[778867D8], ecx
77811FF8 8808 mov byte ptr ds:[eax], cl
77811FF9 3848 02 cmp byte ptr ds:[eax+2], cl
77811FFA 74 05 je ntdll.77811F23
77811FFB E8 94FFFFFF call ntdll.77811EB7
77811FFC 33C0 xor eax, eax
77811FFD C3 ret
77811FFE 8BFF mov edi, edi
77811FFF 55 push ebp
7781200 8BEC mov ebp, esp
7781201 83E4 F8 and esp, FFFFFFF8
7781202 81EC 70010000 sub esp, 170
7781203 A1 70B38877 mov eax, dword ptr ds:[7788B370]
7781204 33C4 xor eax, esp
7781205 898424 6C010000 mov dword ptr ss:[esp+16C], eax
7781206 56 push esi
7781207 8B35 FC918877 mov esi, dword ptr ds:[778891FC]
7781208 57 push edi
7781209 6A 16 push 16
778120A 58 pop eax
778120B 66:894424 10 mov word ptr ss:[esp+10], ax
778120C 8BF9 mov edi, ecx
778120D 6A 10

```

LastError 00000000 (ERROR\_SUCCESS)  
LastStatus C0000034 (STATUS\_OBJECT\_NAME\_NC)

GS 002B FS 0053  
ES 002B DS 002B  
CS 0023 SS 002B

ST(0) 00000000000000000000000000000000 x87r0 Empty 0.  
ST(1) 00000000000000000000000000000000 x87r1 Empty 0.  
ST(2) 00000000000000000000000000000000 x87r2 Empty 0.  
ST(3) 00000000000000000000000000000000 x87r3 Empty 0.  
ST(4) 00000000000000000000000000000000 x87r4 Empty 0.  
ST(5) 00000000000000000000000000000000 x87r5 Empty 0.  
ST(6) 00000000000000000000000000000000 x87r6 Empty 0.

Default (stdcall) 5 Unlocked

1: [esp+4] 77766A78 ntdll.77766A78 "LdrpIn...  
2: [esp+8] 777669FC ntdll.777669FC "minker...  
3: [esp+c] 00000000 00000000  
4: [esp+10] 00000001 00000001  
5: [esp+14] 0019FA18 0019FA18

ntdll.77811EEC

.text:77811EE3 ntdll.dll:\$B1EE3 #B12E3

Dump 1 Dump 2 Dump 3 Dump 4 Dump 5

**2**

Search for Find references to

**3**

All User Modules Command Constant All Modules String references Intermodul... Pattern GUID

**4**

return to ntdll.RtlCap Pointer to SEH\_Record ntDll.wcsomb...+70

Address	Hex	Current Region
77761000	16 00 18 00 F0 7D 76 77	....vw....P vw
77761010	00 00 02 00 0C 5E 76 77	....vw....E vw
77761020	0C 00 08 00 B8 7F 76 77	....vw....[vw
77761030	06 00 08 00 98 7F 76 77	....vw....vw
77761040	06 00 08 00 A0 7F 76 77	....vw....vw
77761050	1C 00 1E 00 84 7C 76 77	....vw....@.vw
77761060	84 00 86 00 B8 81 76 77	....vwPl wD .w
77761070	80 B4 78 77 10 49 86 77	....vwI.wP ywD .w
77761080	90 49 86 77 50 4A 86 77	I.wPJ.wAwvD .w
77761090	E0 25 79 77 E0 69 79 77	E0 49 86 77 %yw wvA w.P w
777610A0	E0 CF 7C 77 D0 4A 86 77	E0 49 86 77 %yw wvA w.P w
777610B0	00 00 00 00 57 14 01 E2	00 00 00 00 00 00 00 00
777610C0	46 15 C5 43 AF FE 00 8D	....W..AF.AC%p.
777610D0	EE E3 D3 F0 06 00 00 0C	iāōđ. .... vw
777610E0	9A 88 13 35 96 5D BD 4F	....5.%o.-CD.%ü:
777610F0	8E 2D A2 44 02 25 F9 3A	....%f. .... vw

Command: Commands are comma separated (like assembly instructions): mov eax, ebx Default

Paused System breakpoint reached! Time Wasted Debugging: 0:00:26:23

Prin forță brută putem testa toate șirurile, dar unele sunt evidente:

User Modules (Strings)			
Address	Disassembly	String A	String B
0041EEF0	cmp eax, 575D40	00575D40	&L"nsemanager-11-1-0"
0043F86F	push advapi32.7567FF95	7567FF95	"0\$brc"
00444D7E	and eax, 594D30	00594D30	"<t"
00466205	mov dword ptr ss:[ebp-64], laboratoratmclar.465C54	00465C54	L"Parola corecta !"
00466248	mov dword ptr ss:[ebp-64], laboratoratmclar.465C7C	00465C7C	L"Parola Incorreta ! Mai incearca"
00466330	push edx, laboratoratmclar.465CC0	00465CC0	L"atm2024"
00484755	push advapi32.7567FF95	7567FF95	"0\$brc"
00489C64	and eax, 594D30	00594D30	"<t"
004B6724	cmp eax, 575D40	00575D40	&L"nsemanager-11-1-0"

1

Facem dublu clic pe sirul de interese care ne-a permis accesul la program.

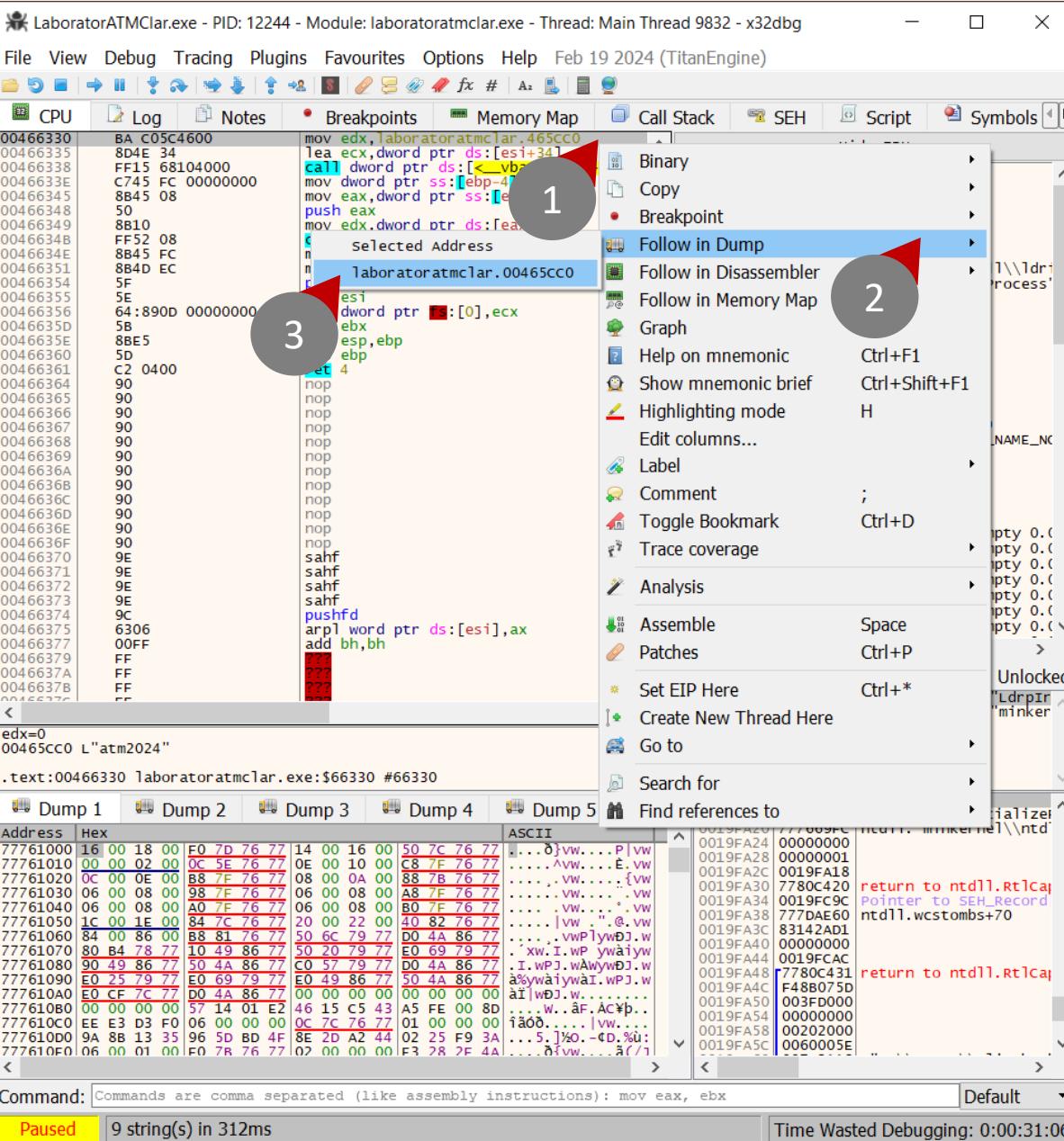


3

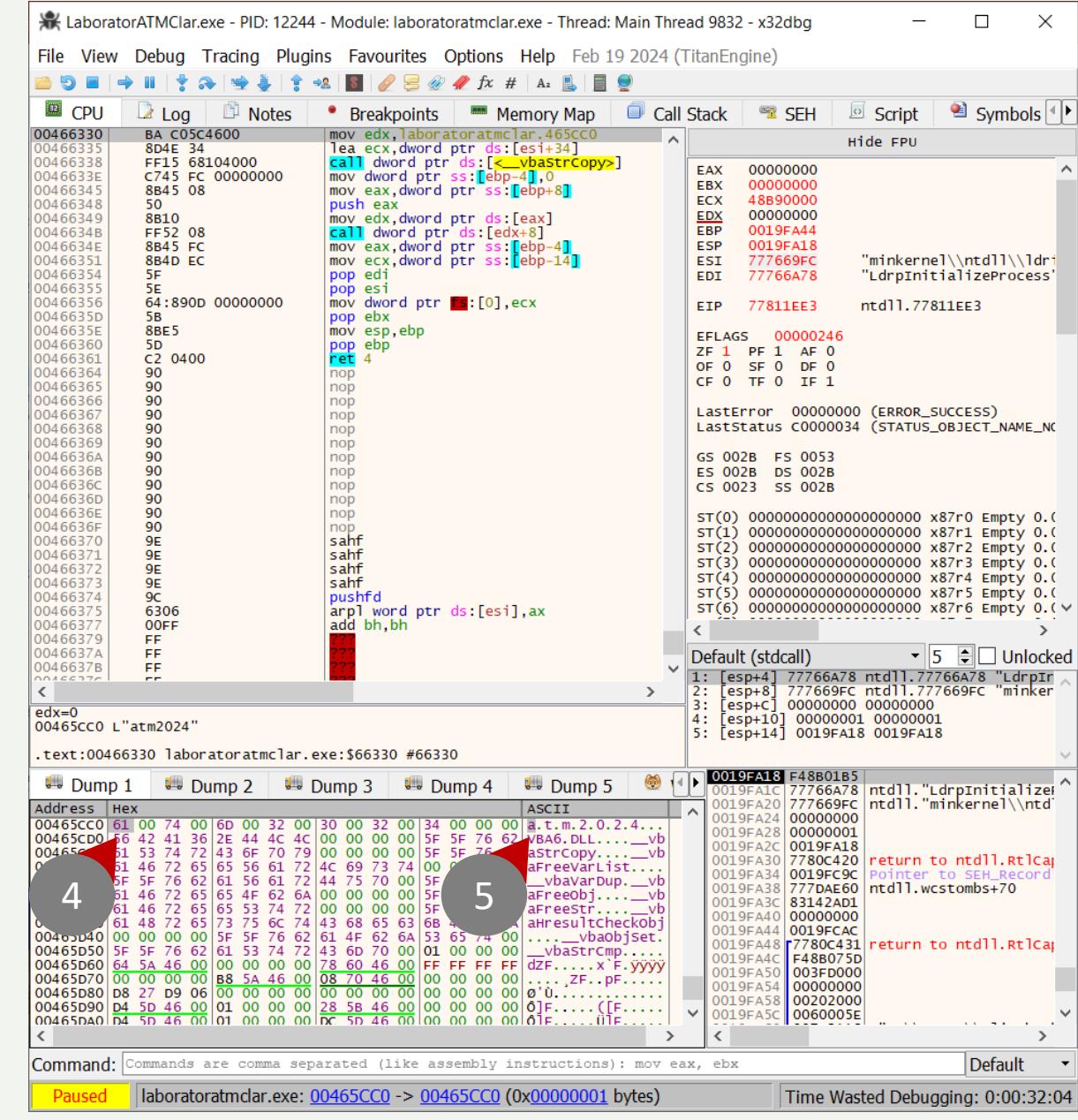
Putem verifica direct string-urile pe care le găsim. Parola este ușor de găsit atât timp cât este în text clar. Întrebarea este: Cum putem schimba parola din fisierul executabil?

The screenshot shows the x32dbg debugger interface. The assembly pane at the top displays assembly code for the function at address 00466330. A call instruction at offset 14 is highlighted with a red arrow and the number '2'. A blue callout box points to this instruction with the text: "Apoi X32dbg ne arată linia care indică o anumită adresă în dump (memorie)". The memory dump pane below shows memory starting at address 0019FA18, which corresponds to the address of the highlighted call instruction. The command bar at the bottom contains the text: "Commands are comma separated (like assembly instructions): mov eax, ebx".

Accesăm adresa *00465CC0* din memoria dump:



În memoria *dump* putem vedea clar parola:



Selectăm parola în *dump* pentru editare:

	Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	
Address	Hex	ASCII				
00465C90	6F 00 72 00	65 00 63 00	74 00 61 00	20 00 21 00	o.r.e.c.t.a.!	
00465CA0	20 00 4D 00	61 00 69 00	20 00 69 00	6E 00 63 00	.M.a.i. i.n.c.	
00465C90	65 00 61 00	72 00 63 00	61 00 00 00	OE 00 00 00	e.a.r.c.a....	
00465CC0	61 00 74 00	6D 00 32 00	30 00 32 00	34 00 00 00	t.m.2.0.2.4..	
00465CDD0	56 42 41 36	2E 44 4C 4C	00 00 00 00	5F 5F 76 00	BA6.DLL....._vb	
00465CE0	61 53 74 72	43 6F 70 79	00 00 00 00	5F 5F 76 00	StrCopy....._vb	
00465CF0	61 46 72 65	65 56 61 72	4C 69 73 74	00 00 00 00	FreeVarList.....	
00465D00	5F 5F 76 62	61 56 61 72	44 75 70 00	5F 5F 76 62	_vbavarDup._vb	
00465D10	61 46 72 65	65 4F 62 6A	00 00 00 00	5F 5F 76 62	aFreeobj....._vb	
00465D20	61 46 72 65	65 53 74 72	00 00 00 00	5F 5F 76 62	aFreestr....._vb	
00465D30	61 48 72 65	73 75 6C 74	43 68 65 63	6B 4F 62 6A	aHRESULTCheckobj	
00465D40	00 00 00 00	05 5F 76 62	61 4F 62 6A	53 65 74 00	....._vbaobjset.	
00465D50	5F 5F 76 62	61 53 74 72	43 6D 70 00	01 00 00 00	_vbastrCmp.....	
00465D60	64 5A 46 00	00 00 00 00	78 60 46 00	FF FF FF FF	dZF.....x`F.yyyy	
00465D70	00 00 00 00	R8 5A 46 00	08 70 46 00	00 00 00 00	7F...nF	

The screenshot shows a dialog box titled "Edit data at 00465CC0". The top menu bar includes "File", "Edit", "View", "Format", "Search", "Help", and "Close". Below the menu, there are tabs: "Hex", "String", and "Copy data". The "String" tab is selected, showing the text "atm2024". A red "X" icon next to "ASCII" indicates it is the current encoding. Below the text are sections for "UNICODE" (containing "atm2024") and "UTF-8" (containing "atm2024"). On the right side, there is a "Codepage..." button. The "Hex" tab displays the hex values: 61 00 74 00 6D 00 32 00 30 00 32 00 34 00. A large gray circle with a red arrow pointing to the number 5 is overlaid on the hex editor area. At the bottom left is a checkbox labeled "Keep Size". At the bottom right are "OK" and "Cancel" buttons.

Edităm secțiunea selectată în dump în modul binar:

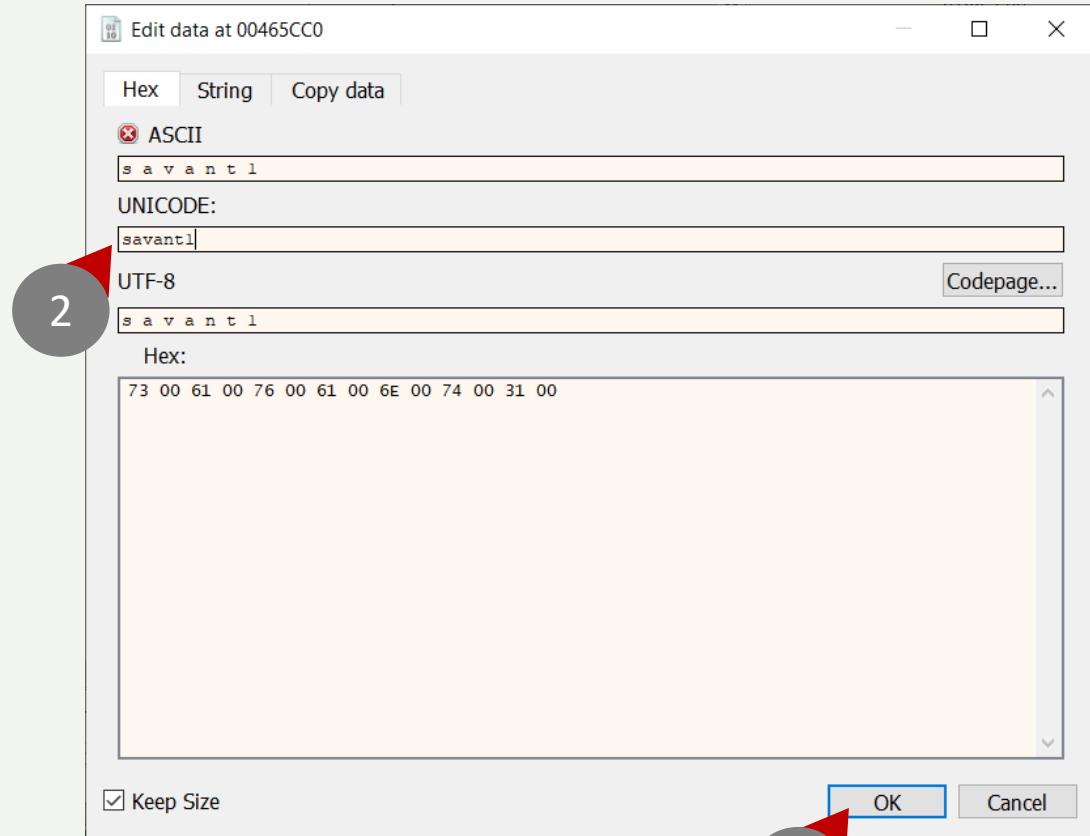
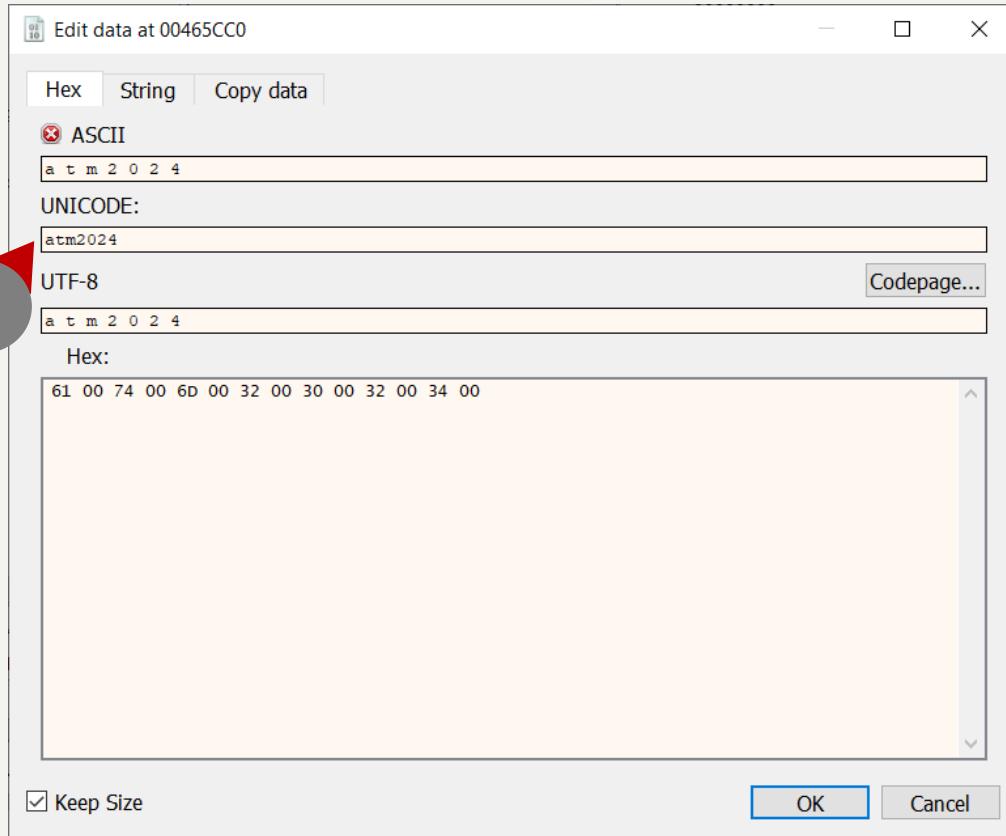
The screenshot shows the x32dbg debugger interface with several panes:

- Registers** pane: Shows CPU register values.
- Stack** pane: Shows the call stack.
- Memory Map** pane: Shows the memory map.
- Call Stack** pane: Shows the current call stack.
- Script** pane: Shows the current script.
- Symbols** pane: Shows symbols.
- Breakpoints** pane: Shows breakpoints.
- Log** pane: Shows log messages.
- Notes** pane: Shows notes.
- CPU** pane: Shows assembly code. A context menu is open over the assembly code area, with the number 4 indicating its position. The menu options include:
  - Edit
  - Ctrl+E
  - Fill F
  - Ctrl+C Shift+C
  - Save To A File
- Binary** pane: Shows binary data. A context menu is open over the binary data area, with the number 3 indicating its position. The menu options include:
  - Binary
  - Copy
  - Follow In Assembler
  - Follow In Memory Map
  - Label Current Address
  - Watch DWORD
  - Modify Value
  - Breakpoint
  - Find Pattern...
  - Find References
  - Sync with expression
  - Allocate Memory
  - Go to
  - Hex
  - A: Text
  - I: Integer
  - F: Float
  - Address
  - Disassembly
- Dump 1**, **Dump 2**, **Dump 3**, **Dump 4**, **Dump 5** panes: Show memory dumps.
- Command** pane: Shows the command input field.
- Paused** status bar: Shows the paused state.
- File Wasted Debugging** status bar: Shows the time wasted debugging.

UTF-16 codifică fiecare caracter pe cel puțin 2 octeti (bytes), chiar și atunci când caracterul este ASCII pur ("a.t.m.2.0.2.4.").

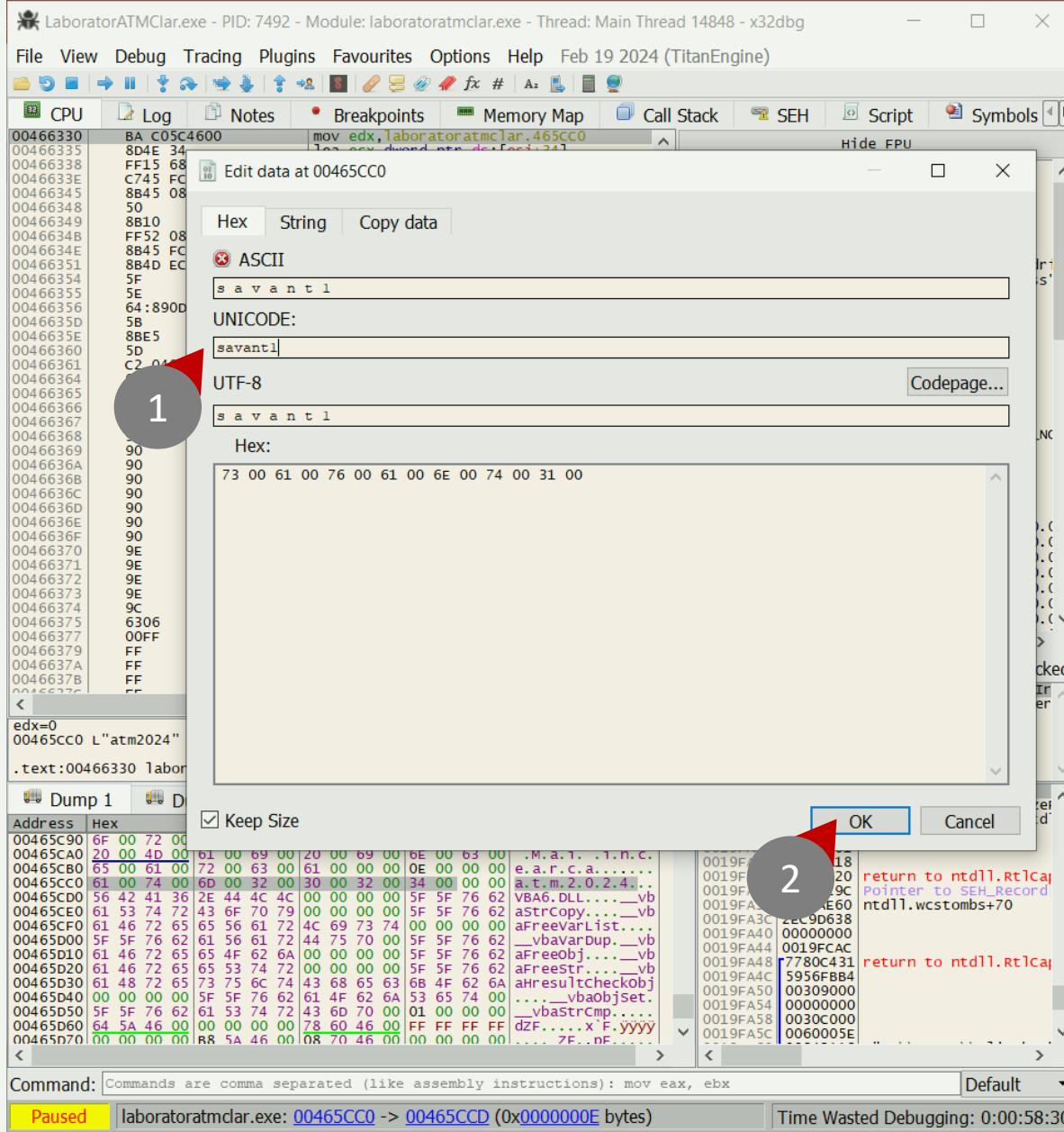
# EDITARE PAROLĂ

## MENȚINEREA DIMENSIUNII ESTE IMPORTANTĂ

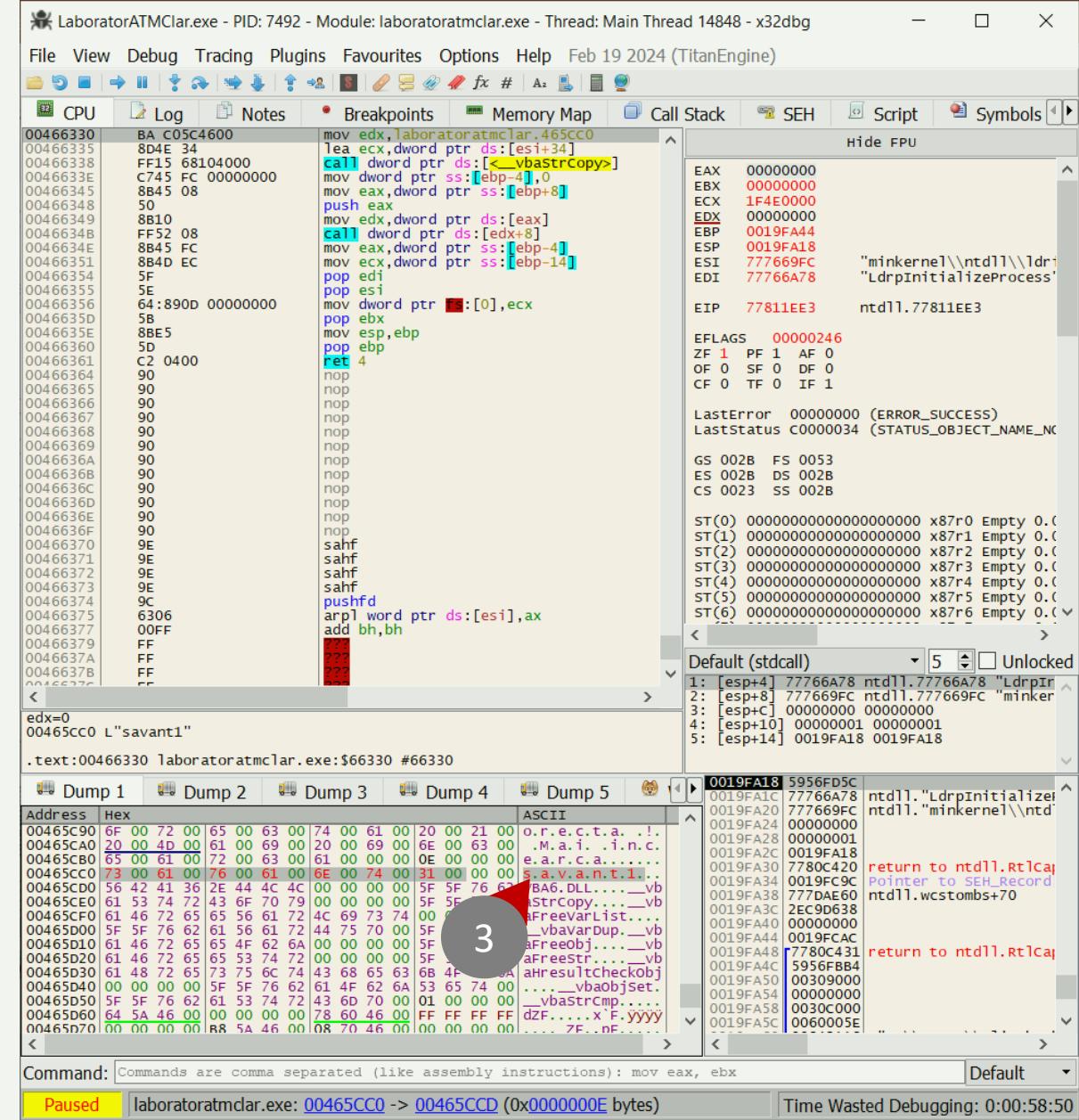


3

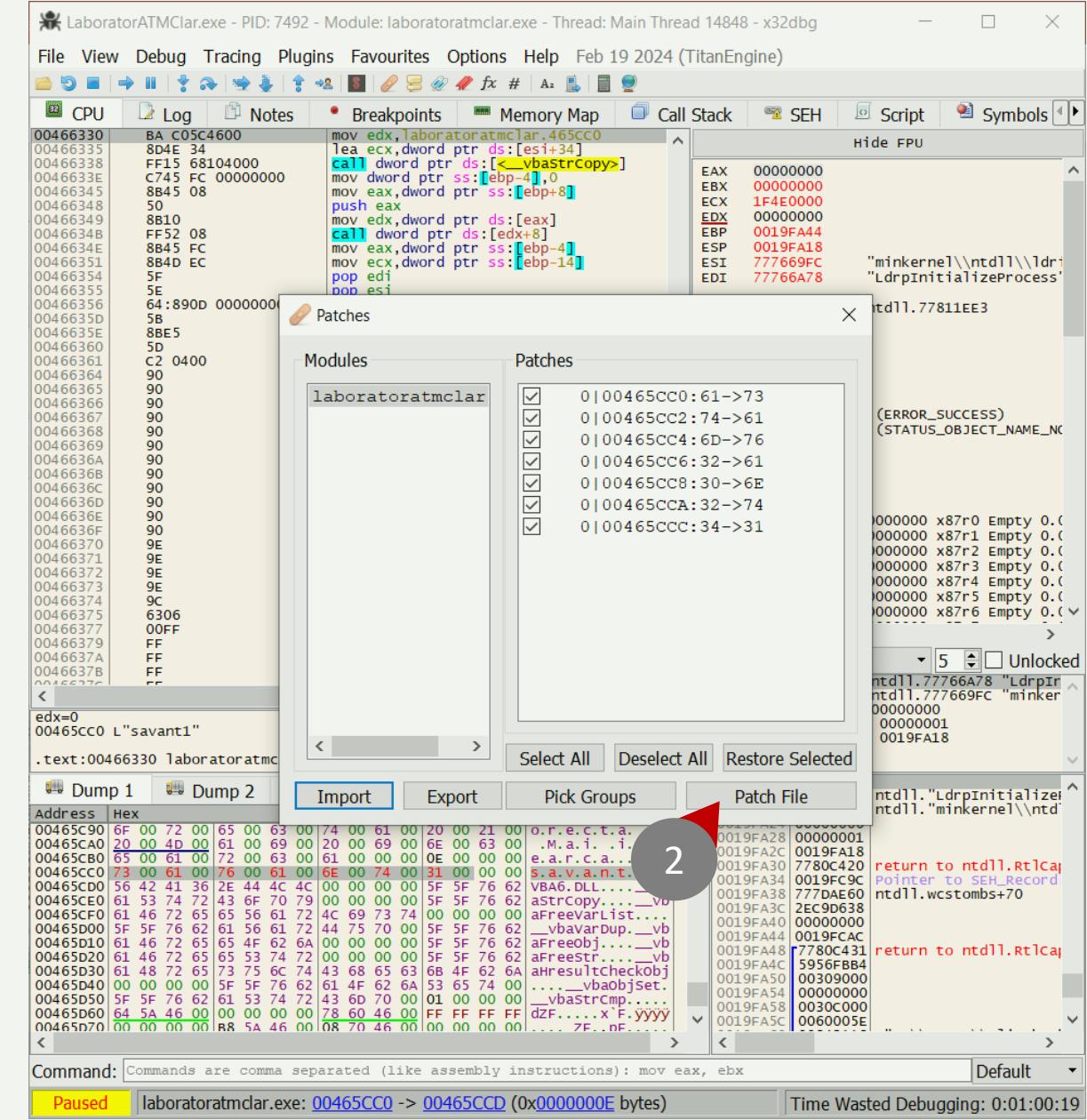
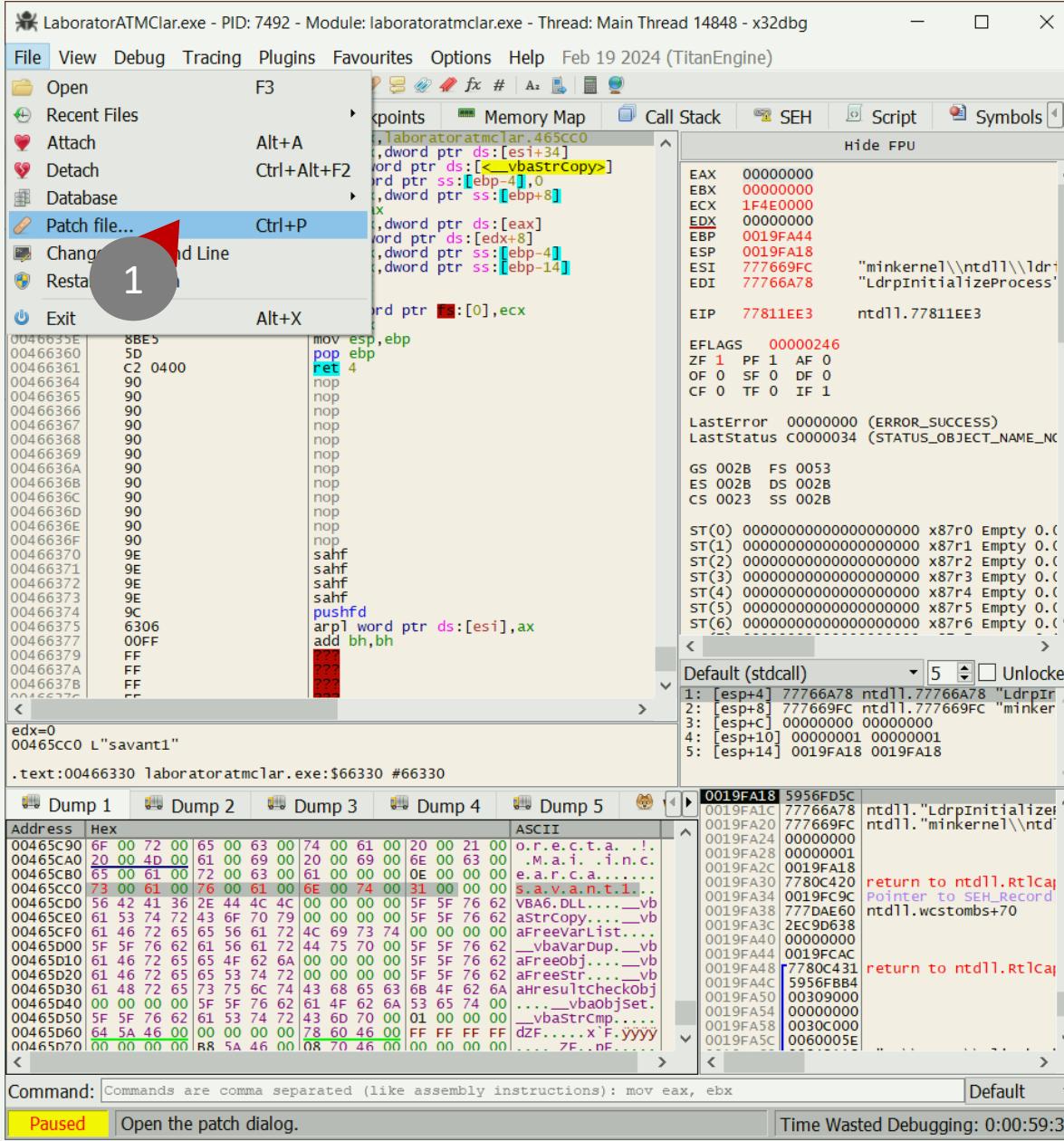
## Schimbă parola:



Observăm modificările salvate în *dump*:

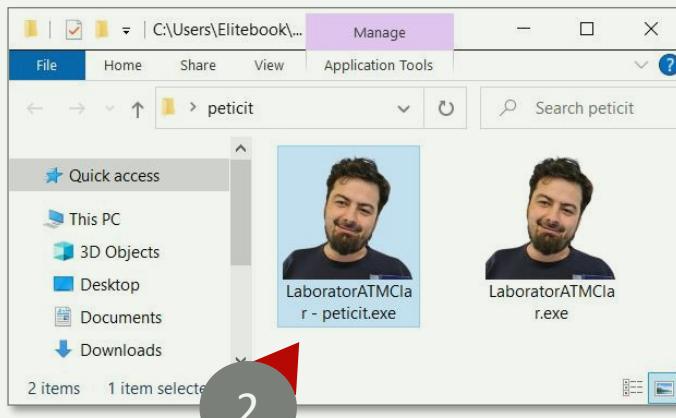
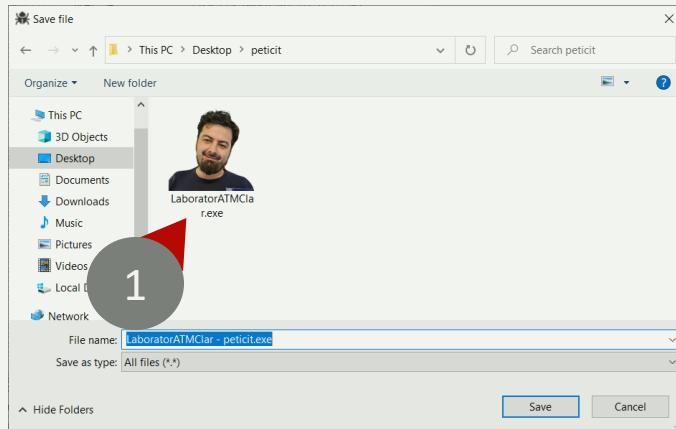


# Scriem noua versiune a executabilului:

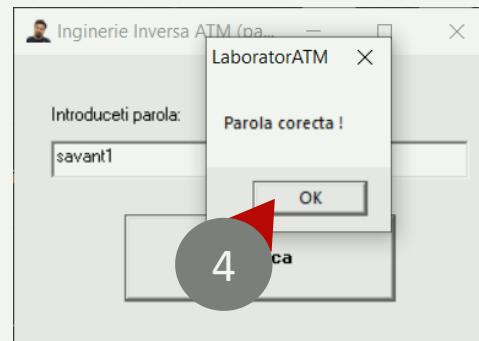


# PAROLĂ SCHIMBATĂ

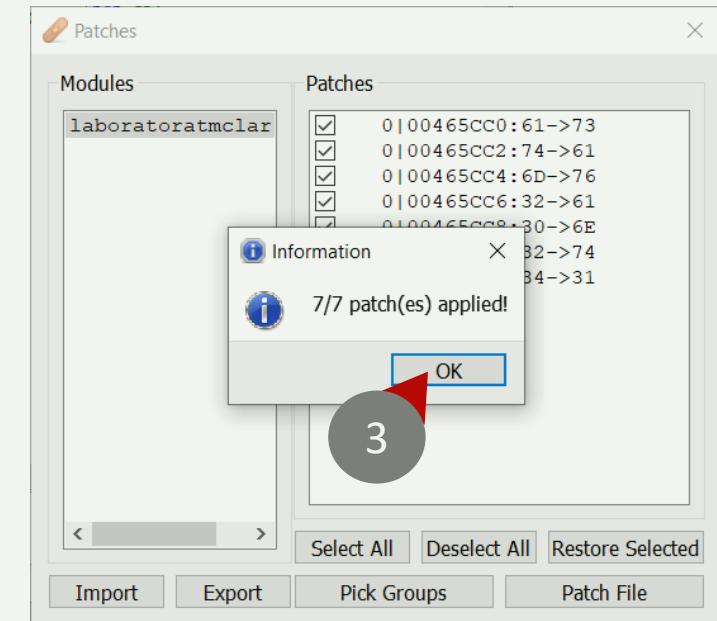
## TESTĂM NOUL EXECUTABIL



Noua parola functioneaza !



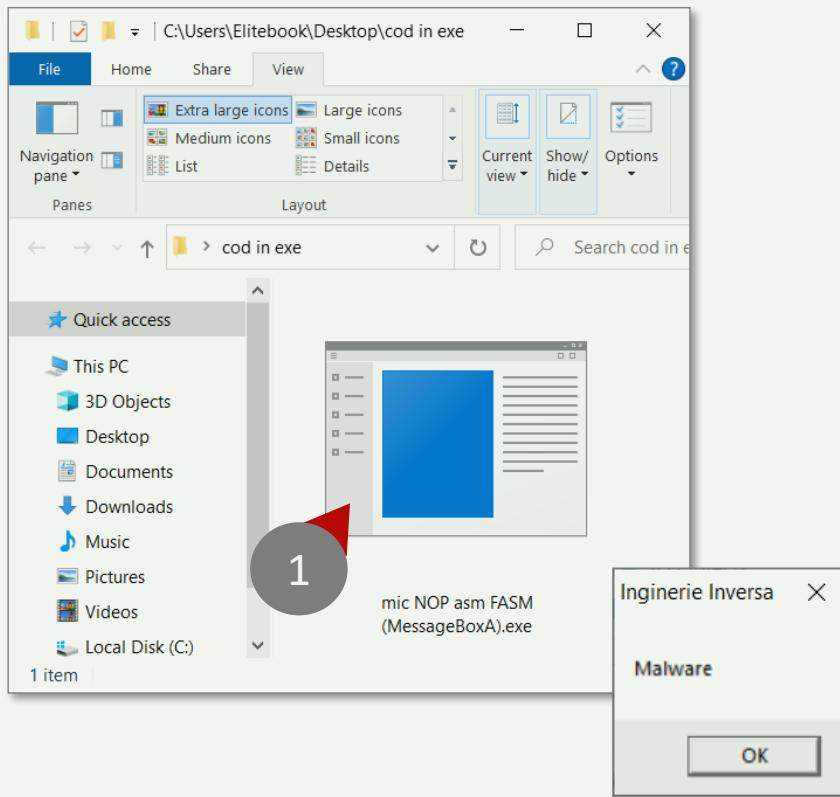
Cate modificari?



## C.8.2 ADĂUGAREA DE COD MAȘINĂ



# “PEŞTERA ÎN COD” (CODE CAVE) SAU CUM SĂ ADĂUGAȚI COD MAŞINĂ IN EXECUTABILE



Cum să adăugați cod mașină în fișiere executabile PE?

Pentru a adăuga cod mașină:

- Trebuie să găsim o zonă din executabil care nu este folosită!
- Trebuie să eliminăm o parte din codul original fără a afecta „prea mult” funcționalitatea!

# Compilarea unui executabilul în FASM (din nou)

```
format PE GUI 4.0 ; specifică formatul executabilului ca fiind PE (Portable Executable) pentru interfața grafică Windows, versiunea 4.0
entry start ; definește punctul de intrare al programului, unde execuția va începe

include 'INCLUDE/win32a.inc' ; include fișierul de antet 'win32a.inc' care conține macro-uri și definiții pentru interfața cu API-ul Windows

section '.data' data readable writeable ; începe o secțiune de date care poate fi citită și scrisă
    title db 'Inginerie Inversă',0 ; definește un sir de caractere pentru titlu, terminat cu null (0) pentru a fi folosit în mesaj
    message db 'Malware',0 ; definește un sir de caractere pentru mesaj, terminat cu null (0)

section '.text' code readable executable ; începe secțiunea de cod, care poate fi citită și executată
    start: ; eticheta 'start', care este punctul de intrare al programului
        NOP ; introducem instrucțiuni NOP (No Operation), atât de căte dorim pentru exemplificare
        NOP
        NOP
        push 0 ; pune valoarea 0 pe stivă, reprezentând handle-ul pentru fereastra părinte (niciuna în acest caz)
        push title ; pune adresa sirului de titlu pe stivă
        push message ; pune adresa sirului de mesaj pe stivă
        push 0 ; pune valoarea 0 pe stivă, care reprezintă stilul cutiei de mesaj (MB_OK)
        call [MessageBoxA] ; apelează funcția MessageBoxA din user32.dll

        push 0 ; pune valoarea 0 pe stivă, argument pentru ExitProcess care indică statusul de ieșire
        call [ExitProcess] ; apelează funcția ExitProcess din kernel32.dll pentru a încheia programul

section '.idata' import data readable writeable ; începe secțiunea de date pentru importuri, care poate fi citită și scrisă
    library kernel32,'KERNEL32.DLL',\ ; specifică că vom folosi funcții din KERNEL32.DLL
        user32,'USER32.DLL' ; și din USER32.DLL

    import kernel32, \
        ExitProcess,'ExitProcess' ; începe definițiile de import pentru kernel32.dll
                                ; specifică că dorim să folosim funcția ExitProcess din acest DLL

    import user32, \
        MessageBoxA,'MessageBoxA' ; începe definițiile de import pentru user32.dll
                                ; specifică că dorim să folosim funcția MessageBoxA din acest DLL
```

# Dezasamblam executabilul:

x test.exe - PID: 15232 - Module: ntdll.dll - Thread: Main Thread 15824 - x32dbg

File View Debug Tracing Plugins Favourites Options Help Feb 19 2024 (TitanEngine)

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols

1

77811EE3 EB 07 jmp ntdll.77811EE3  
77811EE5 33C0 xor eax,eax  
77811EE7 40 inc eax  
77811EE8 C3 ret  
77811EE9 8B65 E8 mov esp,ebp-18h  
77811EEC C745 FC FFFFFFFF mov dword ptr ds:[ebp-4],FFFFFFFFFF  
77811EF3 884D F0 mov ecx,dword ptr ss:[ebp-10]  
77811EF6 64:890D 00000000 mov dword ptr ds:[0],ecx  
77811EF9 59 pop ecx  
77811EFF 5F pop edi  
77811F00 5E pop esi  
77811F01 5B pop ebx  
77811F02 C9 leave  
77811F03 C3 ret  
77811F09 64:A1 30000000 mov eax,dword ptr ds:[30]  
77811F0B 33C9 xor ecx,ecx  
77811F0D 890D D4678877 mov dword ptr ds:[778867D4],ecx  
77811F11 890D D8678877 mov dword ptr ds:[778867D8],ecx  
77811F17 8808 mov byte ptr ds:[eax],cl  
77811F19 3848 02 cmp byte ptr ds:[eax+2],cl  
77811F1C 74 05 je ntdll.77811F23  
77811F1E E8 94FFFFFF call ntdll.77811EB7  
77811F23 33C0 xor eax,eax  
77811F25 C3 ret  
77811F26 8BFF mov edi,edi  
77811F28 55 push ebp  
77811F29 8BEC mov ebp,esp  
77811F2B 83E4 F8 and esp,FFFFFFF8  
77811F2E 81EC 70010000 sub esp,170  
77811F34 A1 70B38877 mov eax,dword ptr ds:[7788B370]  
77811F39 33C4 xor eax,esp  
77811F3B 898424 6C010000 mov dword ptr ss:[esp+16C],eax  
77811F42 56 push esi  
77811F43 8B35 FC918877 mov esi,dword ptr ds:[778891FC]  
77811F49 57 push edi  
77811F4A 6A 16 push 16  
77811F4C 58 pop eax  
77811F4D 66:894424 10 mov word ptr ss:[esp+10],ax  
77811F52 8BF9 mov edi,ecx  
77811F54 6A 18 push 18  
77811F56 58 pop eax  
77811F57 66:894424 12 mov word ptr ss:[esp+12],ax  
77811F5C 8D4424 70 lea eax,dword ptr ss:[esp+70]  
77811F60 894424 6C mov dword ptr ss:[esp+6C],eax  
77811F64 33C0 xor eax,eax  
77811F66 C74424 14 54477677 mov dword ptr ss:[esp+14],ntdll.7776475477  
77811F6E C74424 68 00000001 mov dword ptr ss:[esp+68],1000000  
77811F76 66:894424 70 mov word ptr ss:[esp+70],ax  
77811F7B 85F6 test esi,esi  
77811F7D 74 29 je ntdll.77811FA8

ntdll.77811EEC

.text:77811EE3 ntdll.dll:\$B1EE3 #B12E3

Dump 1 Dump 2 Dump 3 Dump 4 Dump 5

Address	Hex	ASCII
77761000	16 00 18 00 F0 7D 76 77	...}vw...Pvw
77761010	00 00 02 00 0C 5E 76 77	....}vw...Évw
77761020	0C 00 0E 00 B8 7F 76 77	....}vw...;vw
77761030	06 00 08 00 98 7F 76 77	....}vw...vw

Command: Commands are comma separated (like assembly instructions): mov eax, ebx Default

Paused Show this address in memory map view. Equivalent command "memmapdump ad Time Wasted Debugging: 0:03:17:43



Selectați „Memory map” și căutați secțiunea .text a executabilului:

Adresa de început: 00402000  
Dimensiunea (Size): 00001000 (în hexazecimal)  
Secțiunea .text are 4096 bytes în memorie

Dublu Click

Deși codul efectiv are doar 96 bytes, sistemul alocă minim o pagină întreagă (4KB) în memorie pentru .text. În coloana „Size” apare: 00001000 → interpretat ca hex → 4096 în zecimal.

0x1000 = 4 KB  
0x200 = 512 bytes  
0x400 = 1024 bytes etc.

Command: Commands are comma separated (like assembly instructions): mov eax, ebx Default

Paused Show this address in memory map view. Equivalent command "memmapdump ad" Time Wasted Debugging: 0:03:18:24

Suntem îndrumați la adresa punctului de intrare (real):

2

EAX 00000000  
EBX 00000000  
ECX BAA20000  
EDX 00000000  
EBP 000DFA44  
ESP 000DFA18  
ESI 777669FC "minkernel\\ntdll\\LdrpInitializeProc  
EDI 77766A78  
EIP 77811EE3 ntdll.77811EE3  
EFLAGS 00000246  
ZF 1 PF 1 AF 0  
OF 0 SF 0 DF 0  
CF 0 TF 0 IF 1  
LastError 00000000 (ERROR\_SUCCESS)  
LastStatus C0000034 (STATUS\_OBJECT\_NAME\_NOT\_FOUND)  
GS 002B FS 0053

Spațiu liber umplut cu instrucțiuni ADD [EAX], AL (opcode 0x00 0x00) sau pur și simplu zero (00).

Acet spațiu apare imediat după finalul efectiv al codului (call dword ptr ds:[ExitProcess]).

.text:00402000 test.exe:\$2000 #400 <optionalHeader.AddressofEntryPoint>

Dump 1 Dump 2 Dump 3 Dump 4 Dump 5

Address	Hex	ASCII
77761000	16 00 18 00 F0 7D 76 77 14 00 16 00 50 7C 76 77 ... 8 }vw... P vw	000DFA18 E8E4DDC4
77761010	00 00 02 00 0C 5E 76 77 0E 00 10 00 C8 7F 76 77 ... .\vw... E.vw	000DFA1C 77766A78
77761020	00 00 0E 00 B8 7F 76 77 08 00 0A 00 88 7B 76 77 ... .\vw... {vw	000DFA20 777669FC
77761030	06 00 08 00 98 7F 76 77 06 00 08 00 AB 7F 76 77 ... .\vw... vw	000DFA24 00000000

Command: Commands are comma separated (like assembly instructions): mov eax, ebx Default

Paused Show this address in memory map view. Equivalent command "memmapdump ad" Time Wasted Debugging: 0:03:18:46

0x1000 (hexazecimal) = 4096 (zecimal)

```

Command Prompt
Microsoft Windows [Version 10.0.19045.5608]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Paul>cd desktop
C:\Users\Paul\Desktop>dumpbin /headers m.exe
Microsoft (R) COFF Binary File Dumper Version 6.00.8168
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.

Dump of file m.exe
PE signature found
File Type: EXECUTABLE IMAGE

FILE HEADER VALUES
    8664 machine (Unknown)
        5 number of sections
    65F34149 time date stamp Thu Mar 14 20:26:17 2024
        0 file pointer to symbol table
        0 number of symbols
        F0 size of optional header
    22E characteristics
        Executable
        Line numbers stripped
        Symbols stripped
        Application can handle large (>2GB) addresses
        Debug information stripped

```

Mai sus se prezintă utilizarea comenții `dumpbin /headers` aplicată asupra unui fișier executabil (m.exe), prin care sunt afișate informații detaliate despre structura internă a fișierului PE (Portable Executable). Rezultatul include antetul fișierului (File Header), antetul optional (Optional Header) și anteturile secțiunilor (.text, .rdata, .pdata, .xdata, .idata). Aceste informații arată tipul fișierului, arhitectura săptămâna, dimensiuni ale codului și datelor, punctul de intrare, precum și caracteristicile fiecărei secțiuni, fiind utile pentru analiza statică a executabilelor și pentru înțelegerea modului în care sunt organizate datele și instrucțiunile în memorie.

```

Microsoft Windows [Version 10.0.19045.5608]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Paul>cd desktop
C:\Users\Paul\Desktop>dumpbin /headers m.exe
Microsoft (R) COFF Binary File Dumper Version 6.00.8168
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.

Dump of file m.exe
PE signature found
File Type: EXECUTABLE IMAGE

FILE HEADER VALUES
    8664 machine (Unknown)
        5 number of sections
    65F34149 time date stamp Thu Mar 14 20:26:17 2024
        0 file pointer to symbol table
        0 number of symbols
        F0 size of optional header
    22E characteristics
        Executable
        Line numbers stripped
        Symbols stripped
        Application can handle large (>2GB) addresses
        Debug information stripped

OPTIONAL HEADER VALUES
    20B magic #
    2.36 linker version
    200 size of code
    800 size of initialized data
    0 size of uninitialized data
    1000 RVA of entry point
    1000 base of code

```

<p><b>SECTION HEADER #1</b></p> <pre> .text name 60 virtual size 1000 virtual address 200 size of raw data 400 file pointer to raw data 0 file pointer to relocation table 0 file pointer to line numbers 0 number of relocations 0 number of line numbers 60500020 flags Code RESERVED - UNKNOWN RESERVED - UNKNOWN Execute Read </pre> <p><b>SECTION HEADER #2</b></p> <pre> .rdata name 50 virtual size 2000 virtual address 200 size of raw data 600 file pointer to raw data 0 file pointer to relocation table 0 file pointer to line numbers 0 number of relocations 0 number of line numbers 40500040 flags Initialized Data RESERVED - UNKNOWN RESERVED - UNKNOWN Read Only </pre> <p><b>SECTION HEADER #3</b></p> <pre> .pdata name C virtual size 3000 virtual address 200 size of raw data 800 file pointer to raw data 0 file pointer to relocation table 0 file pointer to line numbers 0 number of relocations 0 number of line numbers 40300040 flags Initialized Data RESERVED - UNKNOWN RESERVED - UNKNOWN Read Only </pre>	<p><b>SECTION HEADER #4</b></p> <pre> .xdata name 8 virtual size 4000 virtual address 200 size of raw data A00 file pointer to raw data 0 file pointer to relocation table 0 file pointer to line numbers 0 number of relocations 0 number of line numbers 40300040 flags Initialized Data RESERVED - UNKNOWN RESERVED - UNKNOWN Read Only </pre> <p><b>SECTION HEADER #5</b></p> <pre> .idata name 68 virtual size 5000 virtual address 200 size of raw data C00 file pointer to raw data 0 file pointer to relocation table 0 file pointer to line numbers 0 number of relocations 0 number of line numbers C0300040 flags Initialized Data RESERVED - UNKNOWN RESERVED - UNKNOWN Read Write </pre> <p><b>Summary</b></p> <pre> 68 .idata C .pdata 50 .rdata 60 .text 8 .xdata </pre>
---	---

# De ce există spațiu liber după cod în .text?

- Formatul PE cere ca fiecare secțiune (.text, .data, .rdata etc.) să fie aliniată la o dimensiune multiplă de *SectionAlignment* (RAM - implicit 4096 bytes) și *FileAlignment* (DISC - implicit 512 bytes) din headerul PE.
- De exemplu, chiar dacă .text conține doar 0x1A bytes de cod, secțiunea poate fi alocată pe 0x200 sau 0x1000 bytes în fișierul PE sau în memorie.



# Dacă .text conține 4097 bytes de cod, ce dimensiune va avea secțiunea .text în fișierul PE?

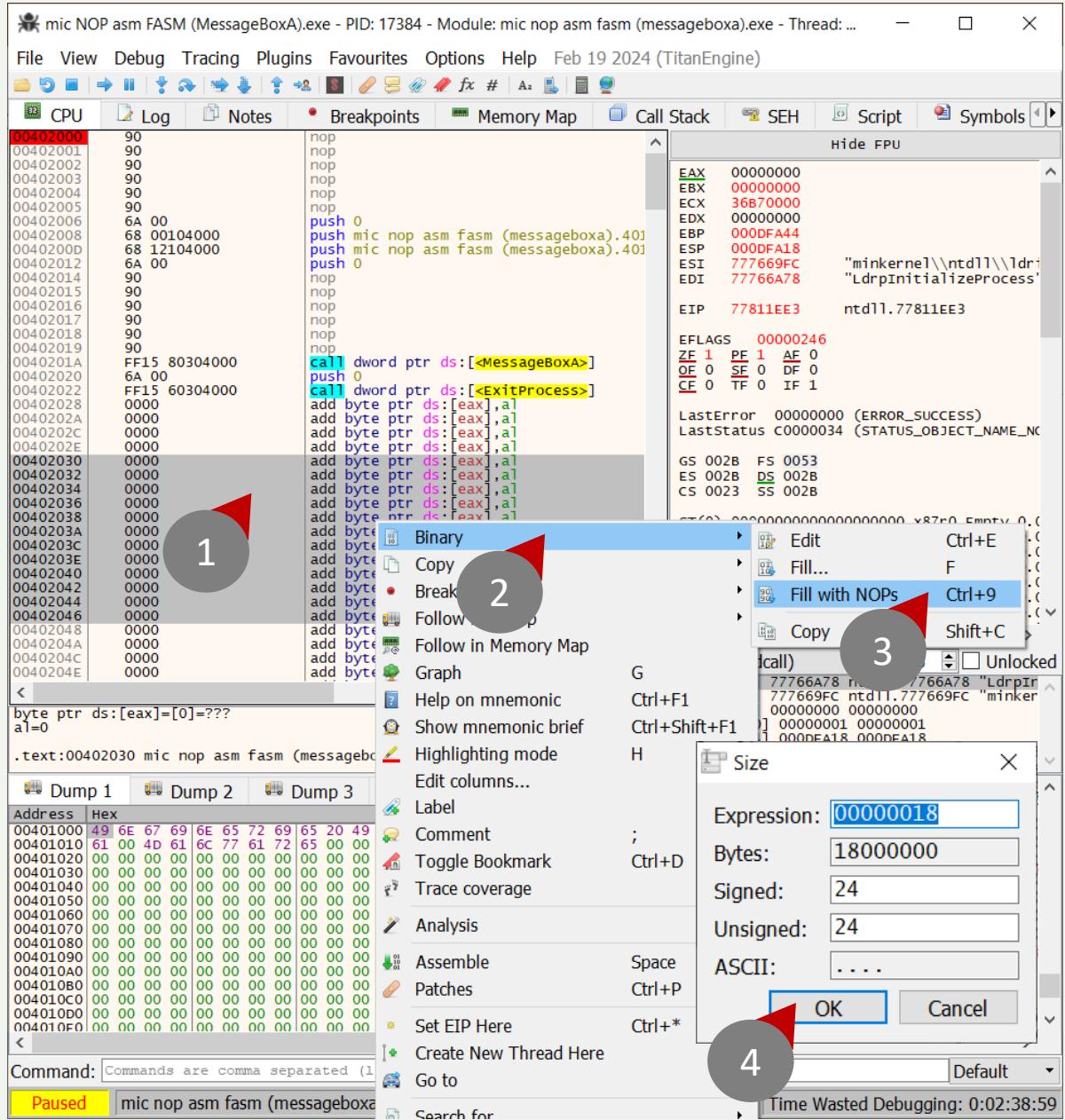
- În fișierul PE (on disk), dimensiunea secțiunii este rotunjită la multiplu de **FileAlignment** (implicit 512 bytes). Deci 4097 este rotunjit la 4608 bytes ( $9 \times 512$ ).
- În RAM (on load), dimensiunea secțiunii este rotunjită la multiplu de **SectionAlignment** (implicit 4096 bytes = 1 pagină de memorie). Deci 4097 este rotunjit la 8192 bytes ( $2 \times 4096$ ).

Prin urmare:

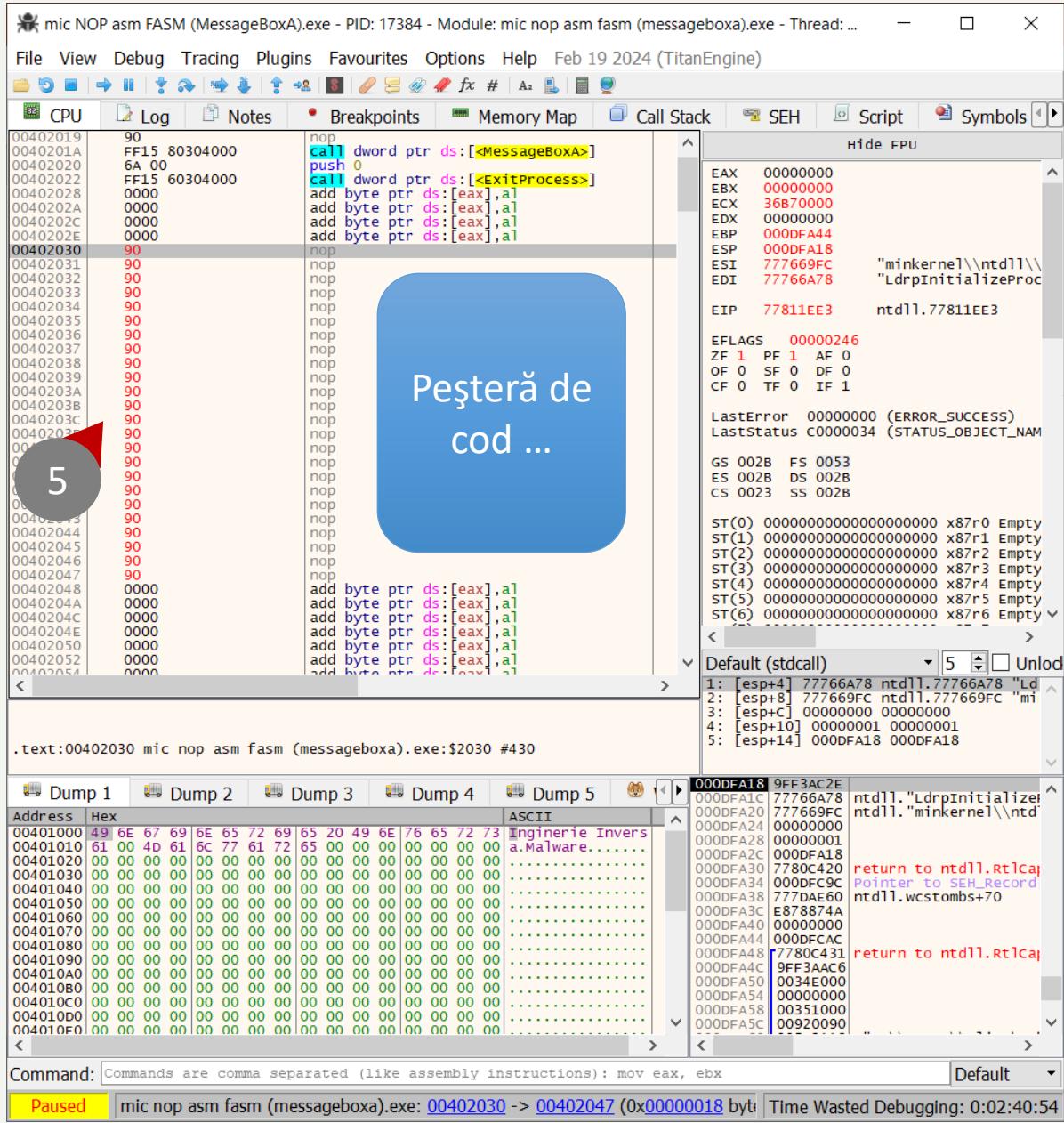
- În fișier: **4608 bytes**
- În RAM: **8192 bytes**



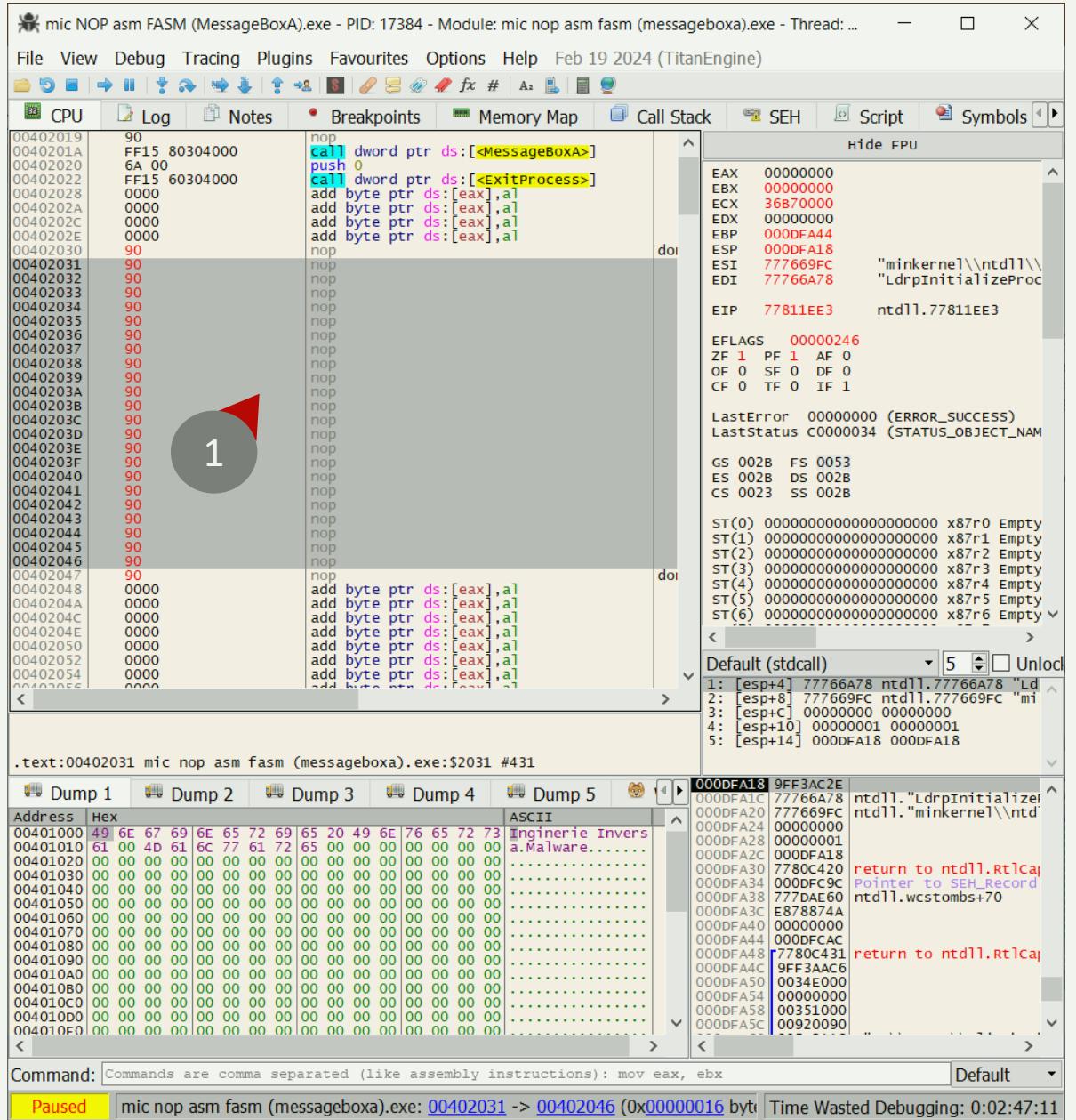
Facem un spațiu umplut cu instrucțiuni **nop**



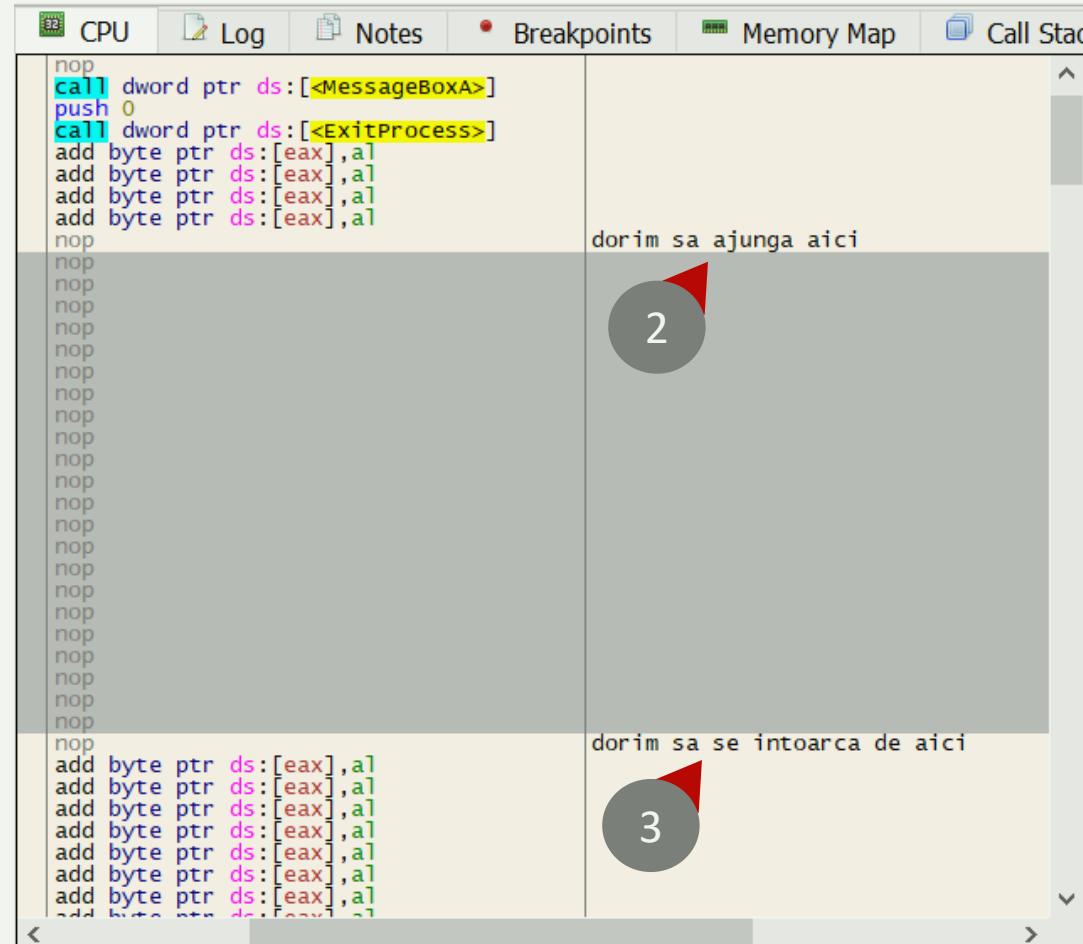
Acum am definit o regiune de interes în zona liberă a executabilului:



Selectați regiunea pentru o vizualizare mai bună:

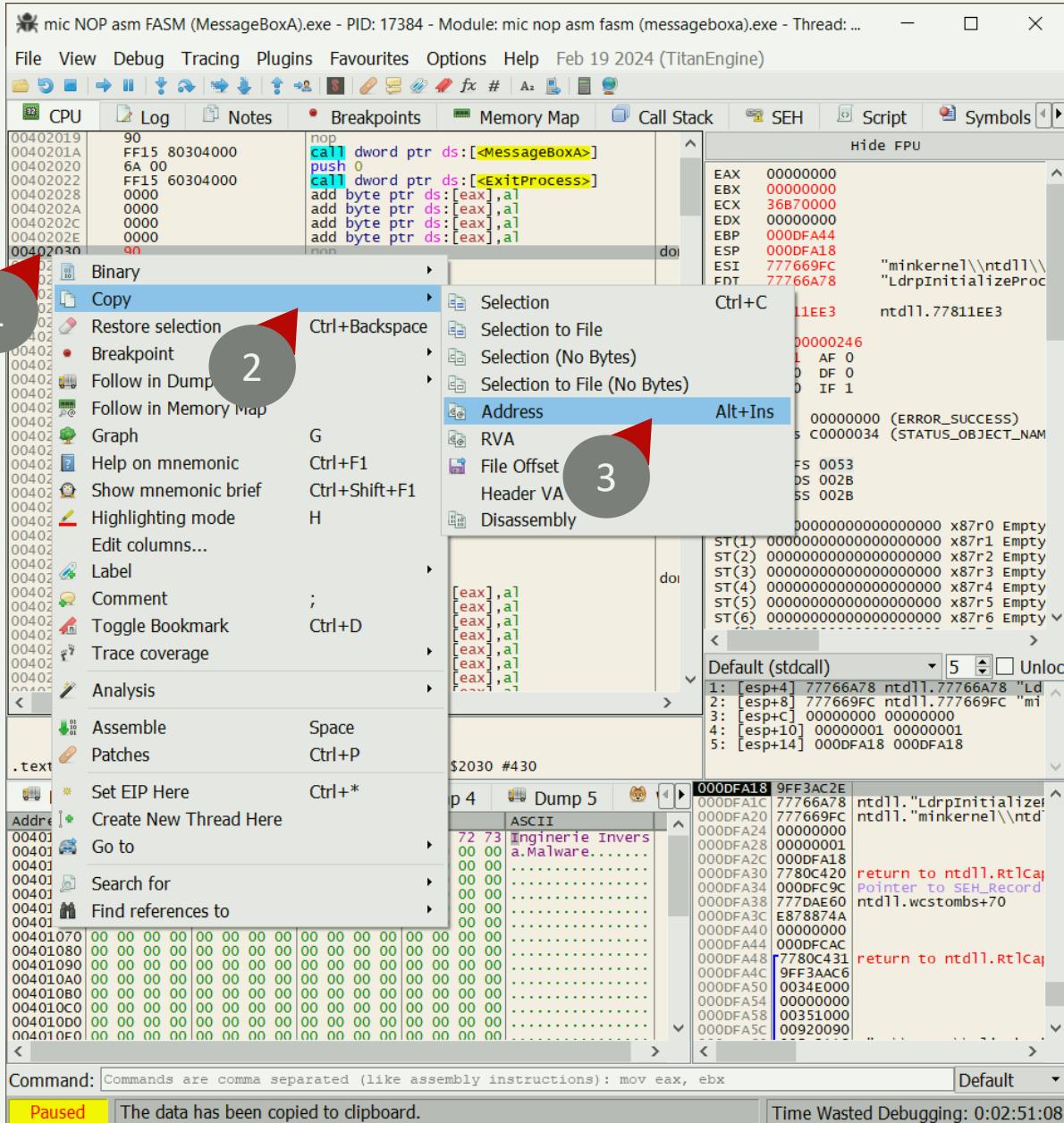


Comentăm punctele de intrare și ieșire din zona delimitată:

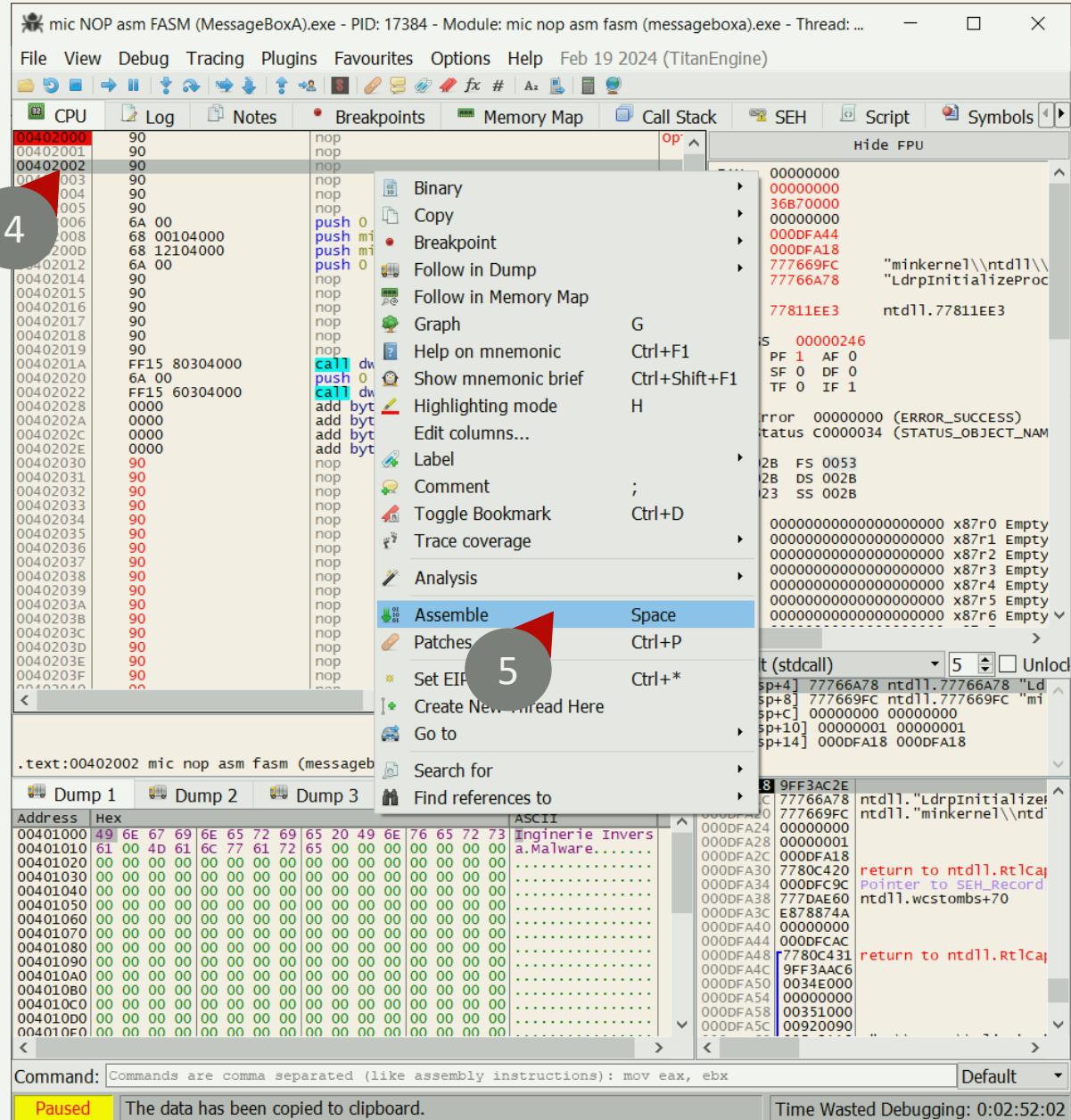


Dorim sa ajungem la adresa 00402030 și să revenim la adresa 00402046.

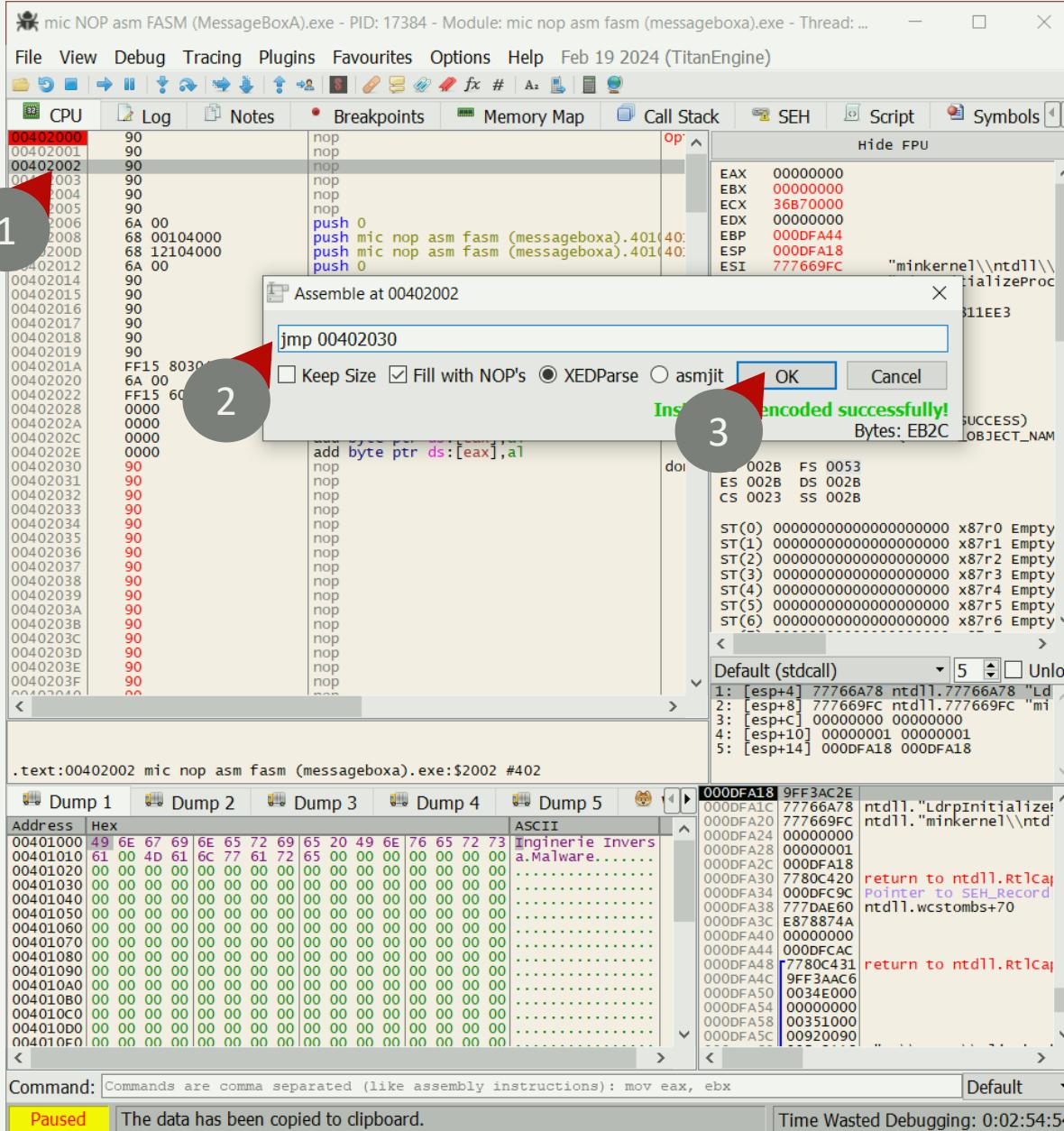
Este copiată adresa punctului de intrare în regiunea definită:



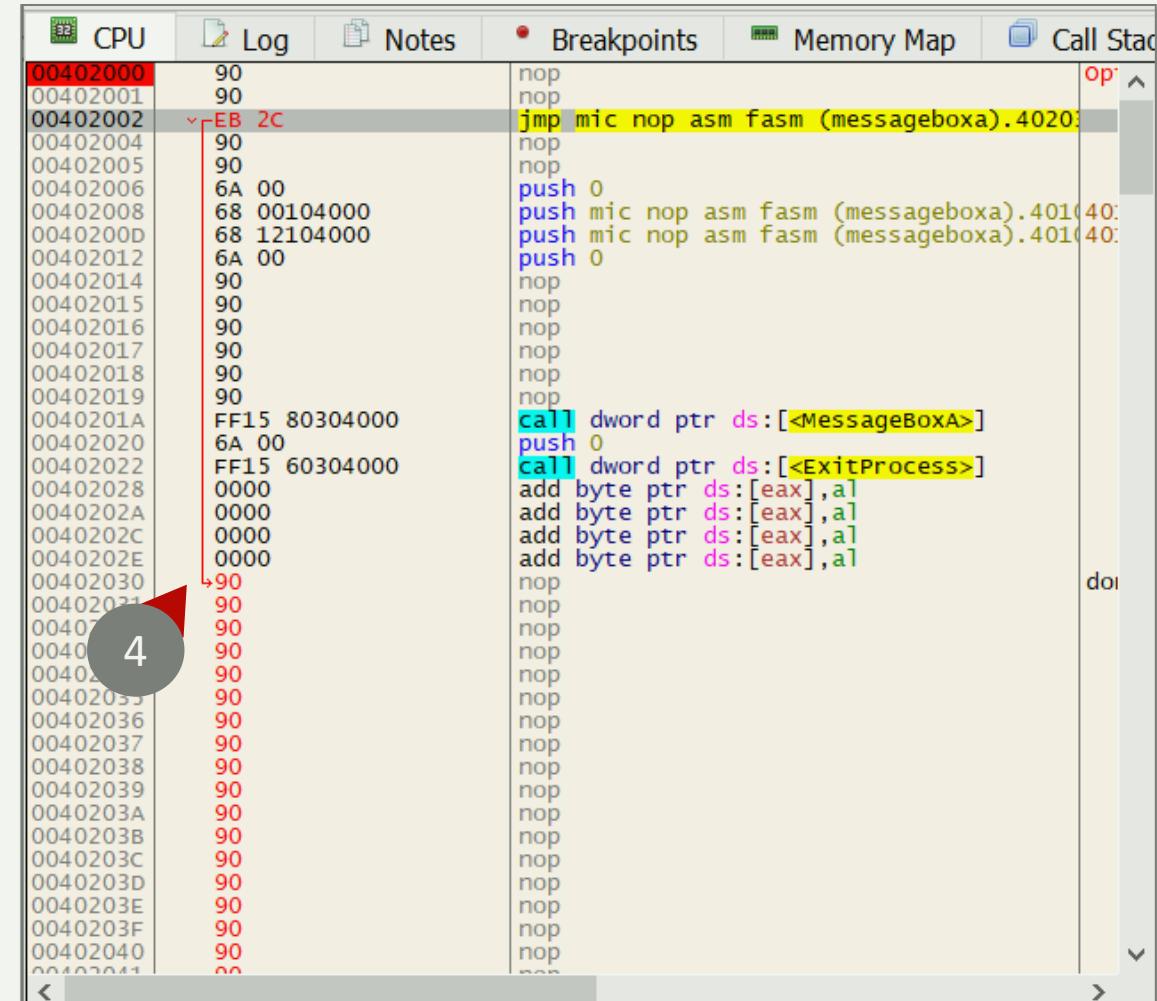
O instrucțiune JMP este inserată spre punctul de intrare în regiune:



Introduceți instrucțiunea JMP + adresa de unde începe regiunea:

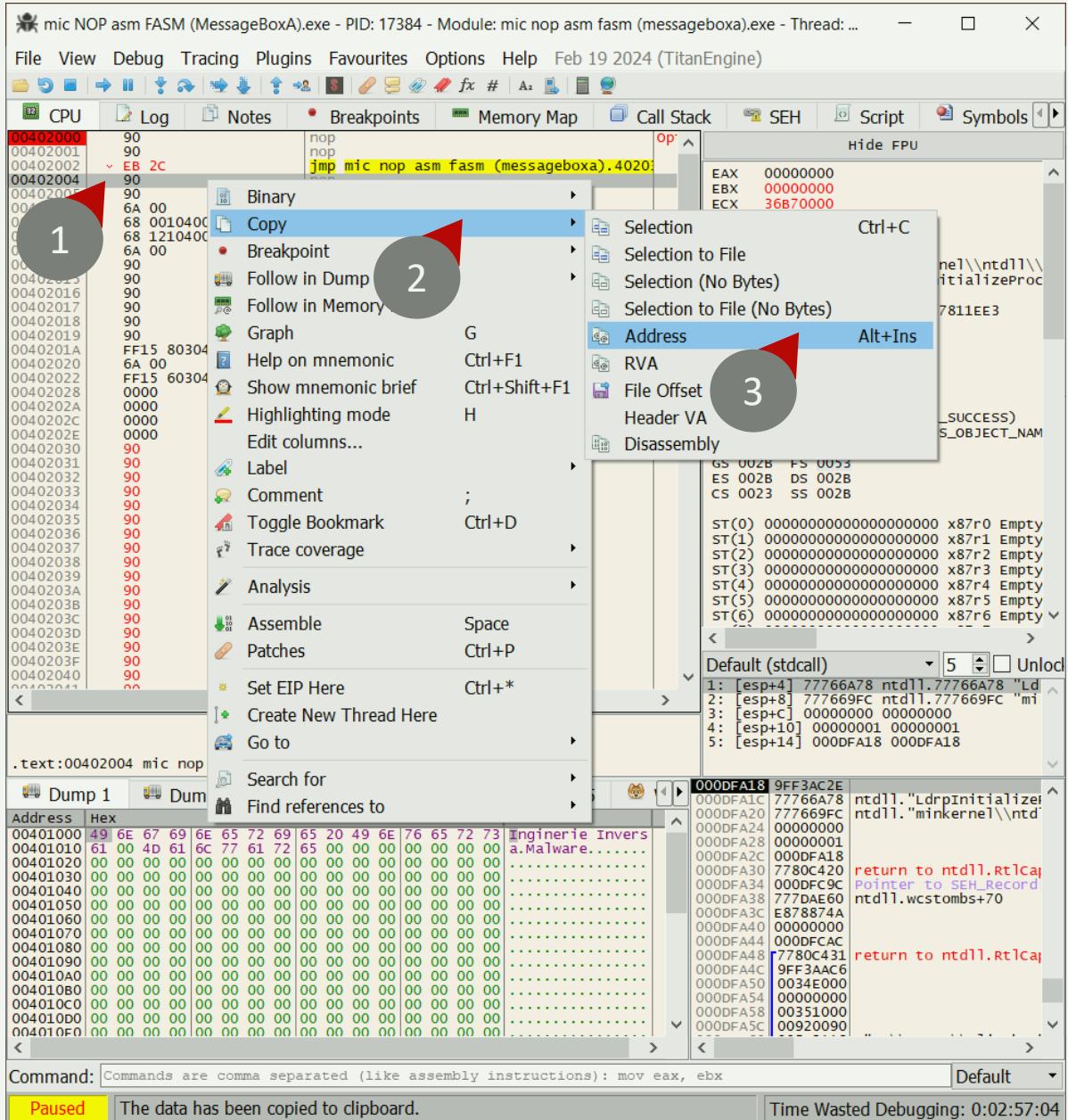


X32dbg permite vizualizarea salturilor cu ajutorul săgeților:

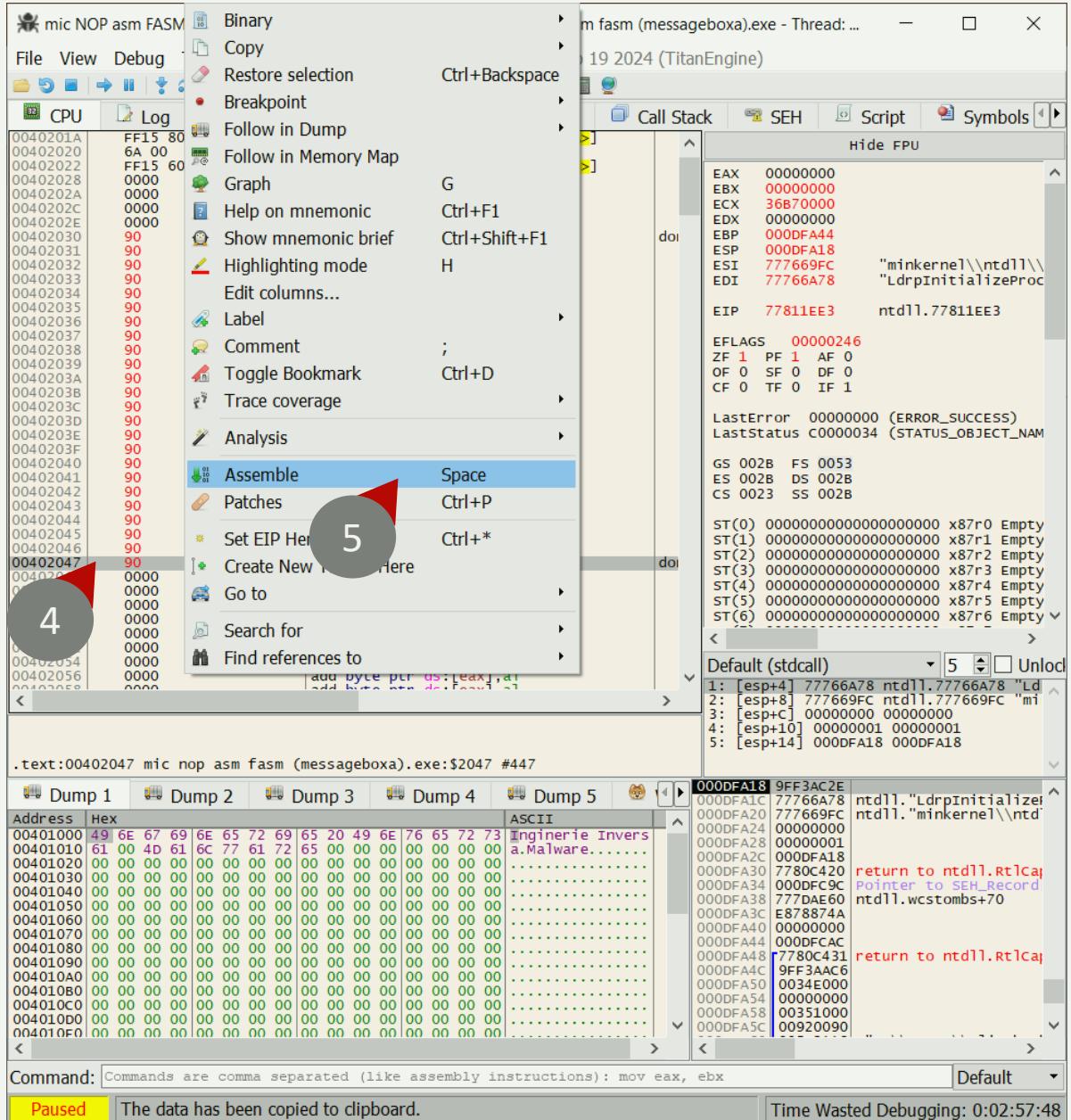


Saritura de la adresa 00402002, la zona imediat după adresa codului ! (00402030)

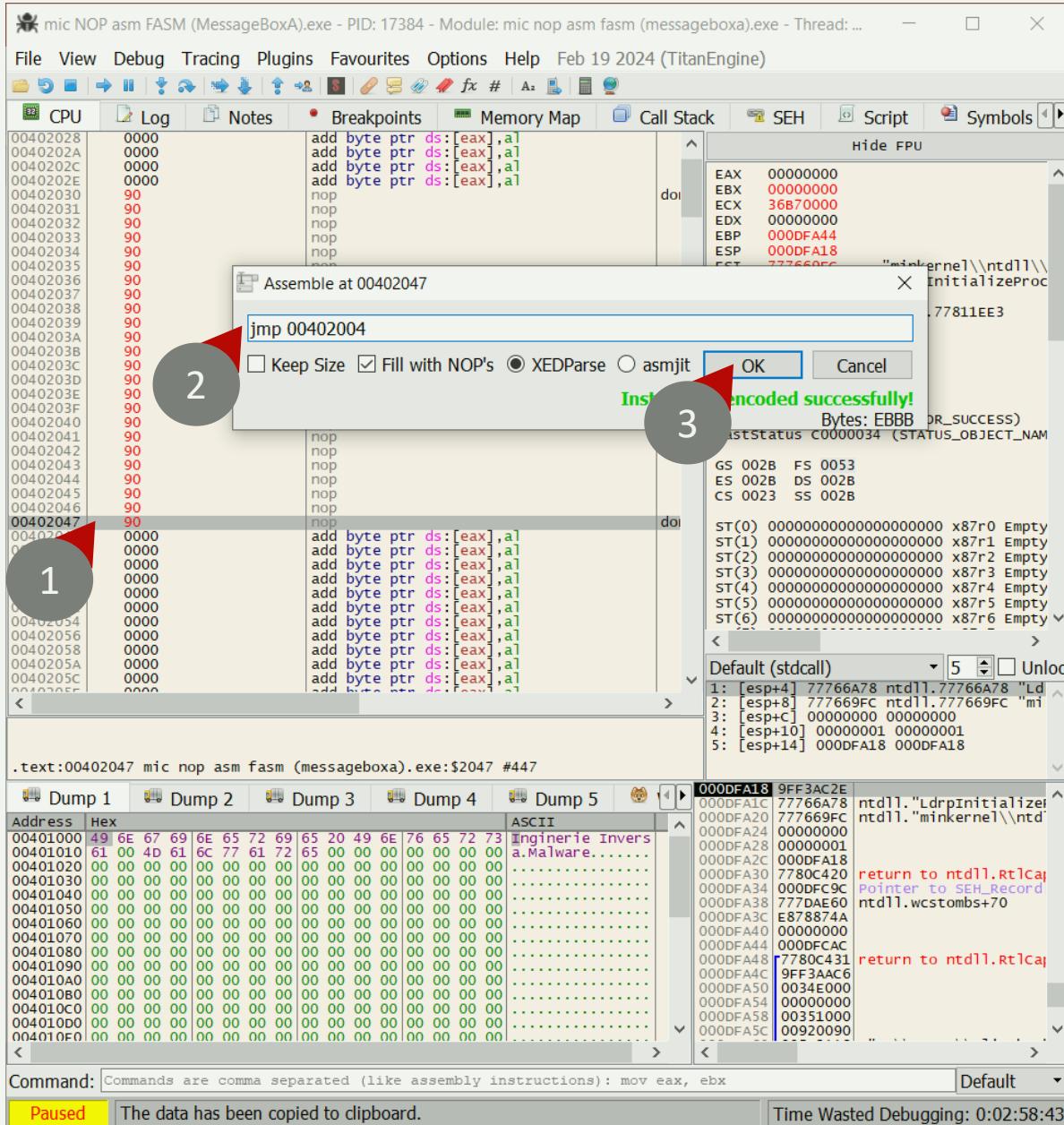
Adresa de după saltul inițial este copiată:



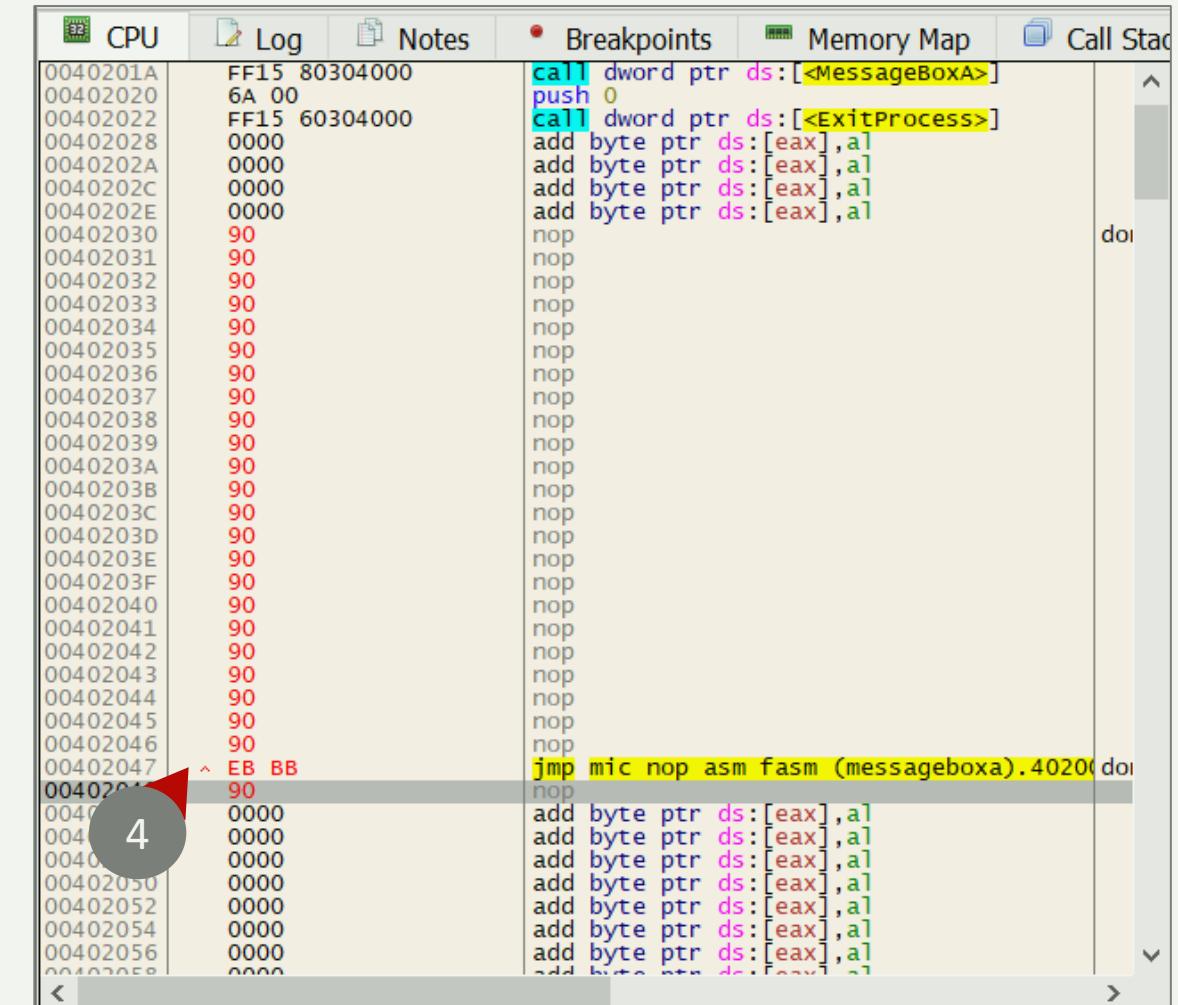
La punctul de ieșire al regiunii, se cere o inserție de instrucțiuni:



În punctul de ieșire din regiune, introduceți JMP + adresa copiată:



Instrucțiunea JMP + adresa copiată a fost scrisă!



Săritura de la adresa 00402047, la zona imediat după adresa punctului de intrare! (00402004)

# “PEŞTERA ÎN COD” (CODE CAVE)

ACUM, ÎN LOC DE INSTRUCȚIUNI NOP, PUTEM SCRIE COD

Avem un „buzunar” gol care poate fi folosit pentru a executa cod arbitrar (care se potrivește cu dimensiunea alocată de noi).

The screenshot shows the OllyDbg debugger's CPU window. The assembly code is as follows:

```
00402000 90          nop
00402001 90          nop
00402002 EB 2C        jmp mic nop asm fasm (messageboxa).4020: [1]
00402004 90          nop
00402005 90          nop
00402006 6A 00        push 0
00402008 68 00 01    push mic nop asm fasm (messageboxa).40140: [4]
0040200D 68 12 10    push mic nop asm fasm (messageboxa).40140: [4]
00402012 6A 00        push 0
00402014 90          nop
00402015 90          nop
00402016 90          nop
00402017 90          nop
00402018 90          nop
00402019 90          nop
0040201A FF15 80304000 call dword ptr ds:[<MessageBoxA>] [2]
00402020 6A 00        push 0
00402022 FF15 60304000 call dword ptr ds:[<ExitProcess>] [3]
00402028 0000
0040202A 0000
0040202C 0000
0040202E 0000
00402030 90          nop
00402031 90          nop
00402032 90          nop
00402033 90          nop
00402034 90          nop
00402035 90          nop
00402036 90          nop
00402037 90          nop
00402038 90          nop
00402039 90          nop
0040203A 90          nop
0040203B 90          nop
0040203C 90          nop
0040203D 90          nop
0040203E 90          nop
0040203F 90          nop
00402040 90          nop
00402041 90          nop
00402042 90          nop
00402043 90          nop
00402044 90          nop
00402045 90          nop
00402046 90          nop
00402047 EB BB        jmp mic nop asm fasm (messageboxa).4020: [4]
```

Annotations:

- Annotation 1: A red arrow points to the instruction at address 00402002, which is a jump to a location starting with EB.
- Annotation 2: A red arrow points to the instruction at address 0040201A, which is a call to the `MessageBoxA` function.
- Annotation 3: A red arrow points to the instruction at address 00402022, which is a call to the `ExitProcess` function.
- Annotation 4: A red arrow points to the instruction at address 00402047, which is a jump to a location starting with EB.

The screenshot shows the OllyDbg debugger's CPU window. The assembly code is as follows:

```
0040201A FF15 80304000 call dword ptr ds:[<MessageBoxA>]
00402020 6A 00        push 0
00402022 FF15 60304000 call dword ptr ds:[<ExitProcess>]
00402028 0000
0040202A 0000
0040202C 0000
0040202E 0000
00402030 90          nop
00402031 90          nop
00402032 90          nop
00402033 90          nop
00402034 90          nop
00402035 90          nop
00402036 90          nop
00402037 90          nop
00402038 90          nop
00402039 90          nop
0040203A 90          nop
0040203B 90          nop
0040203C 90          nop
0040203D 90          nop
0040203E 90          nop
0040203F 90          nop
00402040 90          nop
00402041 90          nop
00402042 90          nop
00402043 90          nop
00402044 90          nop
00402045 90          nop
00402046 90          nop
00402047 EB BB        jmp mic nop asm fasm (messageboxa).4020: [4]
00402048 90          nop
```

Annotations:

- Annotation 1: A red arrow points to the instruction at address 00402002, which is a jump to a location starting with EB.
- Annotation 2: A red arrow points to the instruction at address 0040201A, which is a call to the `MessageBoxA` function.
- Annotation 3: A red arrow points to the instruction at address 00402022, which is a call to the `ExitProcess` function.
- Annotation 4: A red arrow points to the instruction at address 00402047, which is a jump to a location starting with EB.

C.8.3

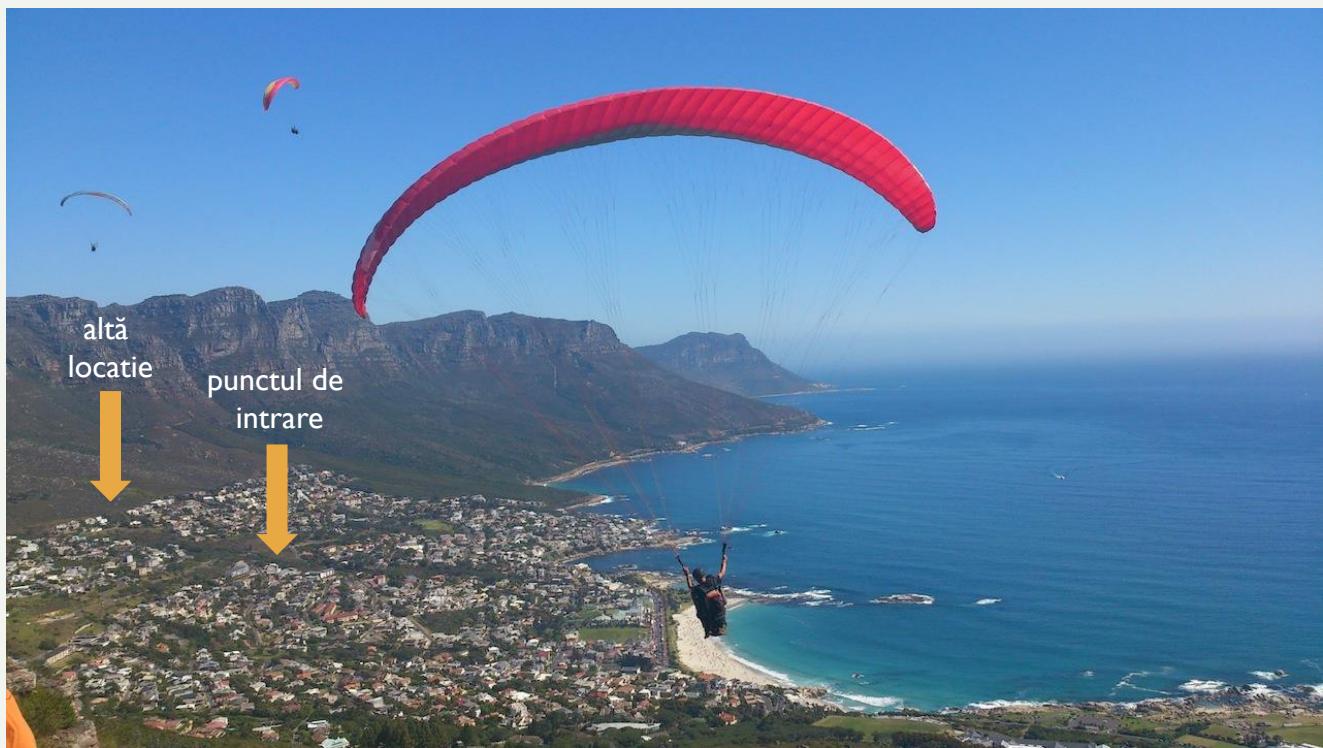
# MODIFICARE ENTRY POINT ȘI INSERTIE DE COD MAȘINĂ



# MODIFICARE ENTRY POINT

## PUNCTUL DE INTRARE CĂTRE ALT SEGMENT

1. Schimbați punctul de intrare la un alt segment
2. Rulați segmentul în regiunea selectată
3. Salt la adresa de sub punctul de intrare



X32dbg afișează punctul de inițializare (nu punctul de intrare):

File View Debug Tracing Plugins Favourites Options Help Feb 19 2024 (TitanEngine)

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols

```

77811EE3 EB 07 jmp ntdll._77811EEC
77811EE5 33C0 xor eax, eax
77811EE7 40 inc eax
77811EE8 C3 ret
77811EE9 8B65 E8 mov esp, dword ptr [ebp-18]
77811EEA C745 FC FFFFFFFF mov dword ptr [esp-4], FFFFFFFF
77811EEB 8B4D F0 mov ecx, dword ptr [ebp-10]
77811EFC 64:890D 00000000 mov dword ptr [esp-10], ecx
77811EFD 59 pop ecx
77811EFF 5F pop edi
77811F00 5E pop esi
77811F01 5B pop ebx
77811F02 C9 leave
77811F03 ret
77811F04 64:A1 30000000 mov eax, dword ptr [esp+30]
77811F05 33C9 xor ecx, ecx
77811F06 890D D4678877 mov dword ptr ds:[778867D4], ecx
77811F07 890D D8678877 mov dword ptr ds:[778867D8], ecx
77811F08 8808 mov byte ptr ds:[eax], cl
77811F09 3848 02 cmp byte ptr ds:[eax+2], cl
77811F10 74 05 je ntdll._77811F23
77811F11 E8 94FFFFFF call ntdll._77811EB7
77811F12 33C0 xor eax, eax
77811F13 C3 ret
77811F14 8BFF mov edi, edi
77811F15 55 push ebp
77811F16 8BEC mov ebp, esp
77811F17 83E4 F8 and esp, FFFFFFFF
77811F18 81EC 70010000 sub esp, 170
77811F19 A1 70B38877 mov eax, dword ptr ds:[7788B370]
77811F1A 33C4 xor eax, esp
77811F1B 898424 6C010000 mov dword ptr ss:[esp+16C], eax
77811F1C 56 push esi
77811F1D 8B35 FC918877 mov esi, dword ptr ds:[778891FC]
77811F1E 57 push edi
77811F1F 6A 16 push 16
77811F20 58 pop eax
77811F21 66:894424 10 mov word ptr ss:[esp+10], ax
77811F22 8BF9 mov edi, ecx
77811F23 6A 18 push 18
77811F24 58 pop eax
77811F25 66:894424 12 mov word ptr ss:[esp+12], ax
77811F26 8D4424 70 lea eax, dword ptr ss:[esp+70]
77811F27 894424 6C mov dword ptr ss:[esp+6C], ax

```

ntdll.\_77811EEC

.text:77811EE3 ntdll.dll:\$B1EE3 #B1EE3

Dump 1 Dump 2 Dump 3 Dump 4 Dump 5

Address	Hex	ASCII
77761000	16 00 18 00	E0 7D 76 77 14 00 16 00
77761010	00 00 02 00	0C 5E 76 77 0E 00 10 00
77761020	00 00 0E 00	B8 7F 76 77 08 00 0A 00
77761030	06 00 08 00	88 78 76 77 06 00 08 00
77761040	06 00 08 00	A8 7F 76 77 06 00 08 00
77761050	1C 00 18 00	84 7C 76 77 20 00 22 00
77761060	84 00 86 00	B8 81 76 77 50 6C 79 77
77761070	80 84 78 77	10 49 86 77 50 4A 86 77
77761080	90 49 86 77	50 4A 86 77 C0 57 79 77
77761090	ED 25 79 77	ED 69 79 77 ED 49 86 77

Command: Commands are comma separated (like assembly instructions): mov eax, ebx Default

Paused System breakpoint reached! Time Wasted Debugging: 0:03:29:52

Selectați fila „Memory map” și accesați secțiunea .text:

File View Debug Tracing Plugins Favourites Options Help Feb 19 2024 (TitanEngine)

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols

Address	size	Party	Info
00010000	00001000	User	Reserved
00020000	00001000	User	Reserved
00030000	00001000	User	Reserved
00040000	0001D000	User	Reserved
00060000	00035000	User	Reserved
00095000	00008000	User	Reserved
000AA000	0003A000	User	Reserved
000DA000	00006000	User	stack (21220)
000E0000	00040000	User	Reserved
000F0000	00020000	User	Reserved
00100000	00001000	User	Reserved
00110000	00001000	User	Reserved
00120000	00001000	User	Reserved
00130000	00010000	User	Reserved
00140000	00035000	User	Reserved
00175000	00008000	User	Reserved
00180000	00035000	User	Reserved
001B5000	00008000	User	Reserved
001D0000	00007000	User	Reserved (001D0000)
001D7000	00009000	User	Reserved
00200000	0007F000	User	PEB, TEB (11880), Wow64 TEB (11880), TEB (21220), Wow64 TEB (21220), test.exe
0027F000	0000E000	User	Reserved (00200000)
00280000	00173000	User	Reserved
00400000	00010000	User	test.exe
00401000	00001000	User	..data
00402000	00001000	User	..text
00403000	00001000	User	..idata
00410000	000C9000	User	\Device\HarddiskVolume3\Windows\system32\locale.nls
004E0000	00035000	User	Reserved
00515000	00008000	User	Heap
005A0000	00006000	User	Reserved
005B0000	0000F000	User	Reserved
006A0000	00003000	User	Stack (18164)
006B0000	0000F000	User	Reserved
007A0000	00003000	User	Stack (11880)
007B0000	0000F000	User	Reserved
008A0000	00003000	User	win32u.dll
75811000	00001000	System	..text
75811000	00006000	System	..rdata
75817000	0000E000	System	..data
75825000	00001000	System	..rsrc
75826000	00001000	System	..reloc
75D40000	00010000	System	kernelbase.dll
75D41000	001DE000	System	..text
75F1E000	00004000	System	..data
75F23000	00006000	System	..idata
75F29000	00001000	System	..didat
75F2A000	00001000	System	..rsrc
75F2B000	00031000	System	..reloc
766E0000	00001000	System	msvcp_win.dll
766E1000	0006E000	System	..text
7674F000	00003000	System	..data
76752000	00002000	System	..idata
76754000	00001000	System	..didat
76755000	00001000	System	..rsrc
76756000	00005000	System	..reloc
76760000	00001000	System	user32.dll
76761000	000A5000	System	..text
76806000	00004000	System	..data
7680A000	00009000	System	..idata
76813000	00001000	System	..didat

Command: Commands are comma separated (like assembly instructions): mov eax, ebx Default

Paused System breakpoint reached! Time Wasted Debugging: 0:03:33:11

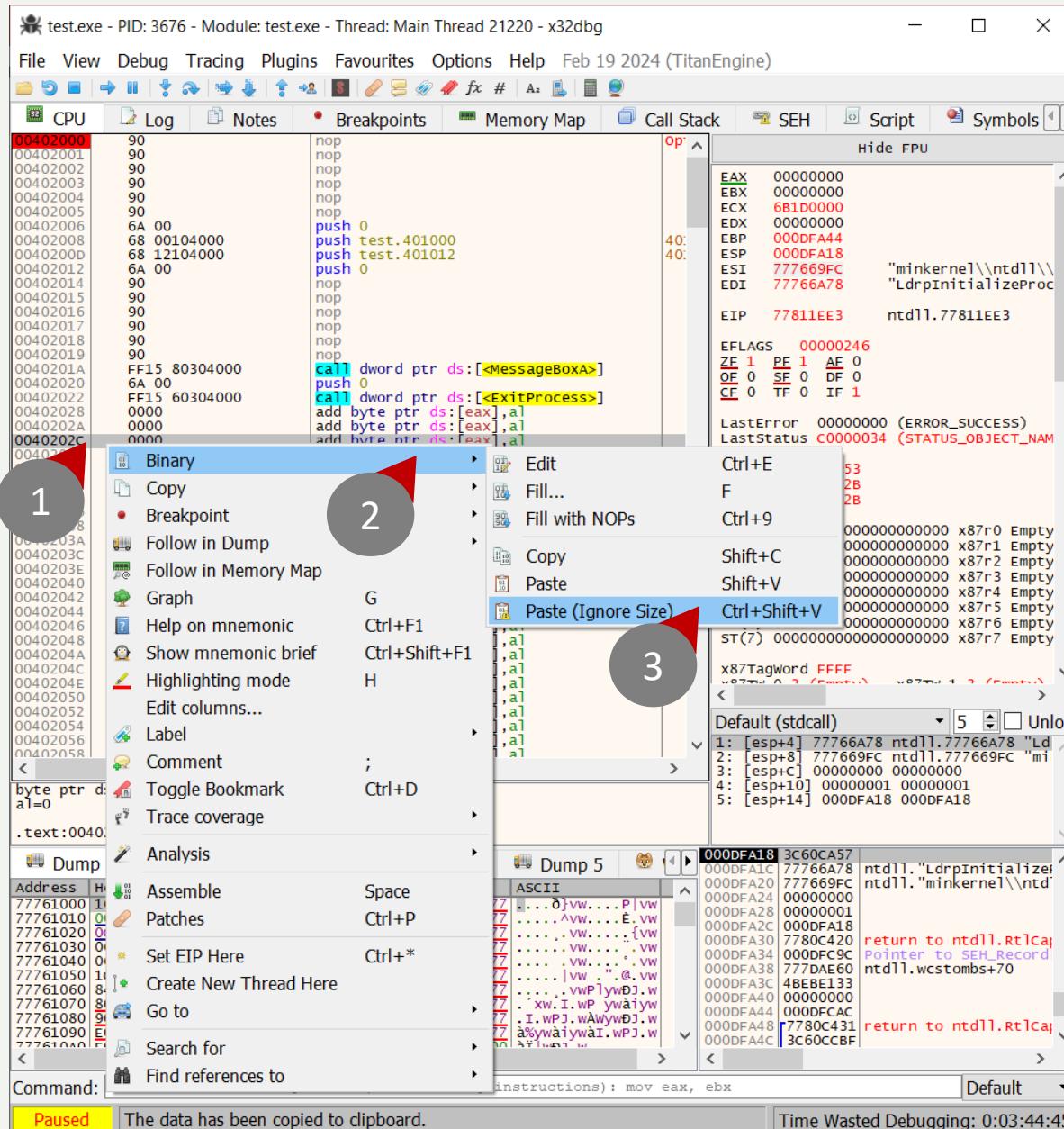
După accesarea secțiunii .text, x32dbg afișează punctul de intrare:

The screenshot shows the x32dbg debugger interface. The assembly window displays the entry point at address 00402000, which is a sequence of NOPs (opcode 90). The CPU window shows the registers EAX through EIP. The CPU register pane highlights the EIP register with the value 77811EE3. The memory dump window at the bottom shows the first 16 bytes of memory starting from address 77761000, which contain the string "minkernel\\ntdll\\LdrpInitializeProc". A red circle labeled '1' points to the assembly window.

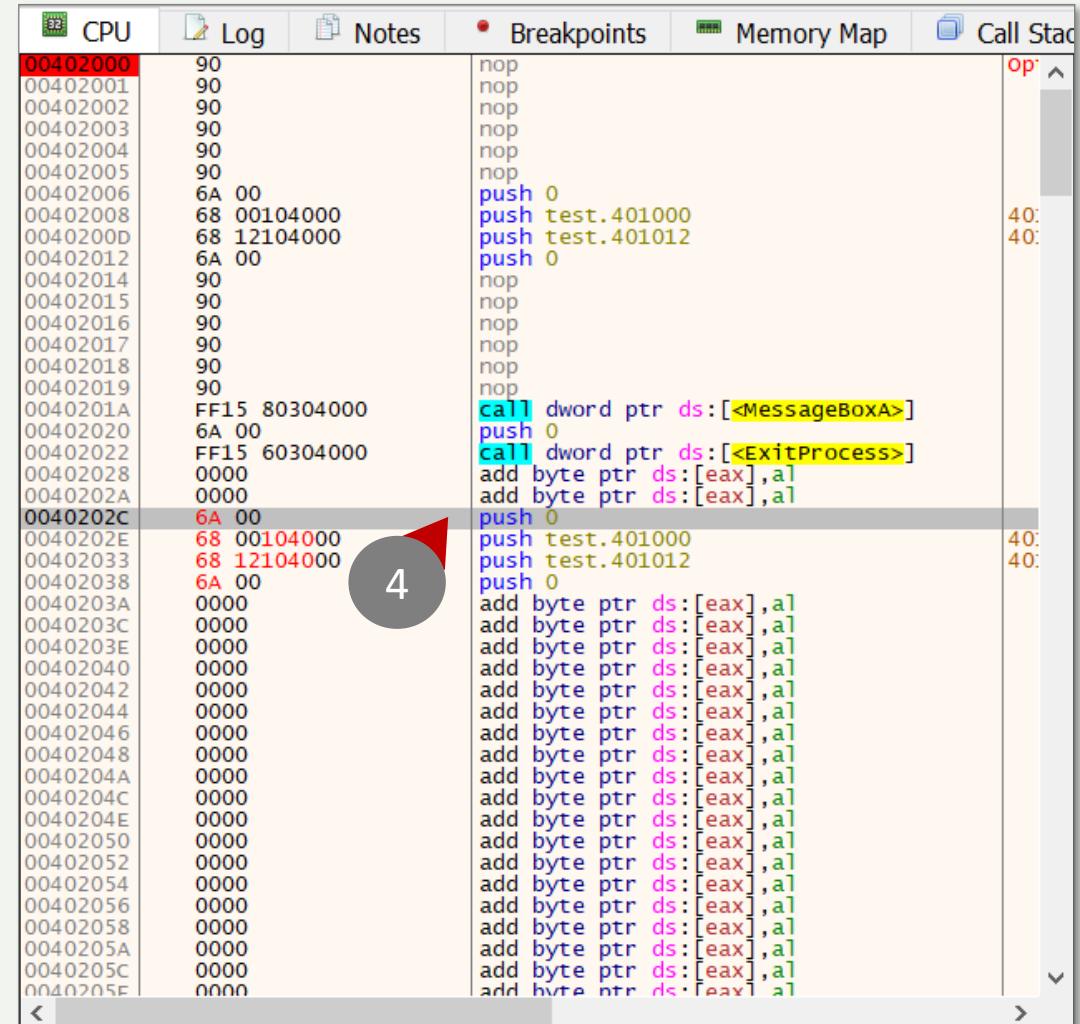
Selectați partea de cod, pe care apoi o copiați:

The screenshot shows the x32dbg debugger interface with the assembly window. A right-click context menu is open over the assembly code, specifically over the instruction at address 00402006. The menu is titled 'Binary' and includes options like 'Copy', 'Edit', 'Fill...', 'Fill with NOPs', and 'Copy'. A red circle labeled '2' points to the 'Copy' option in the menu. A red circle labeled '3' points to the 'Copy' option in the main menu bar. A red circle labeled '4' points to the 'Copy' option in the assembly context menu. The assembly window shows the instruction at address 00402006 as 'push 0'.

Clonăm cu „paste” codul într-o zonă liberă nedefinită:

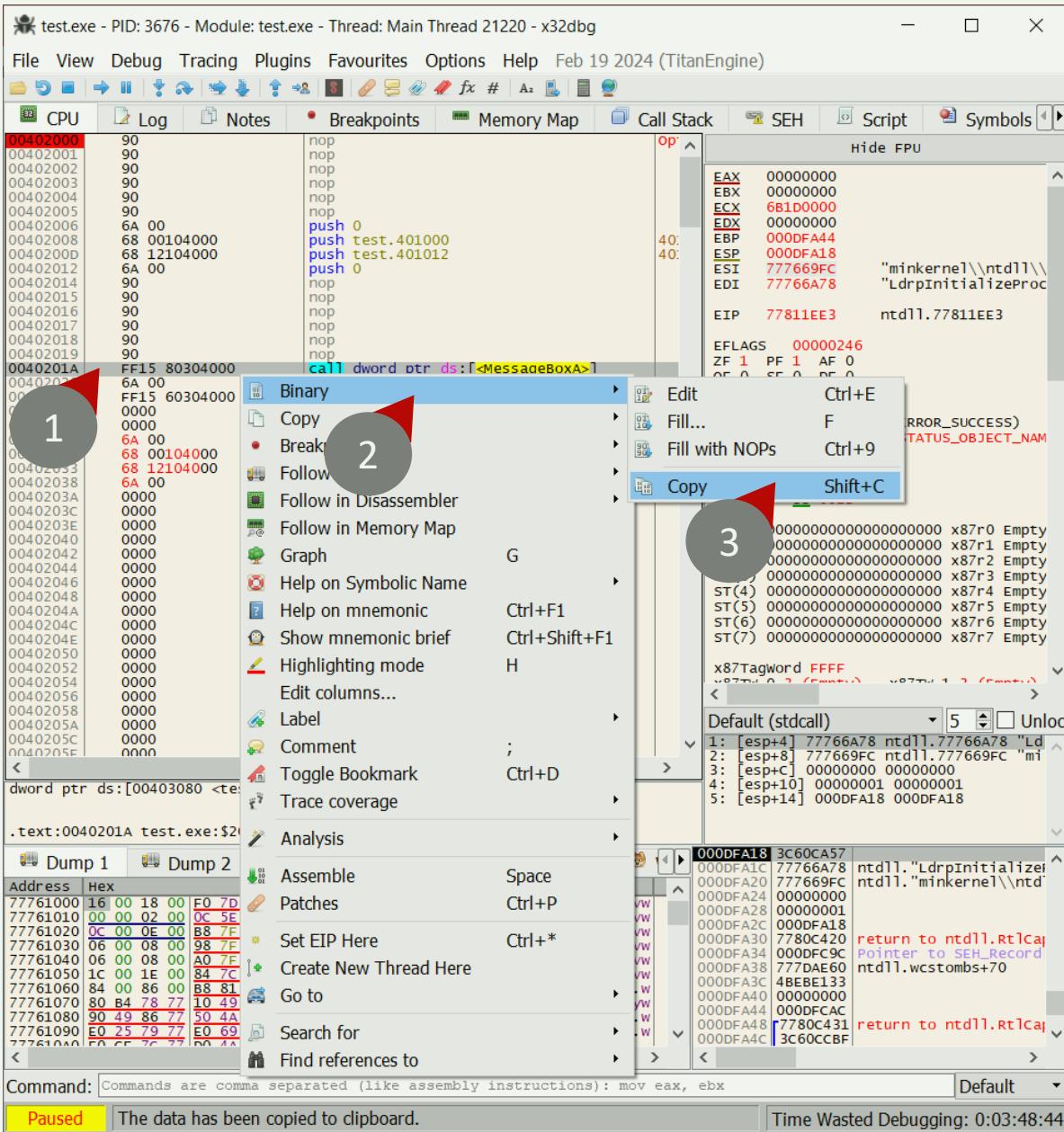


Codul clonat este afișat la adresa selectată:

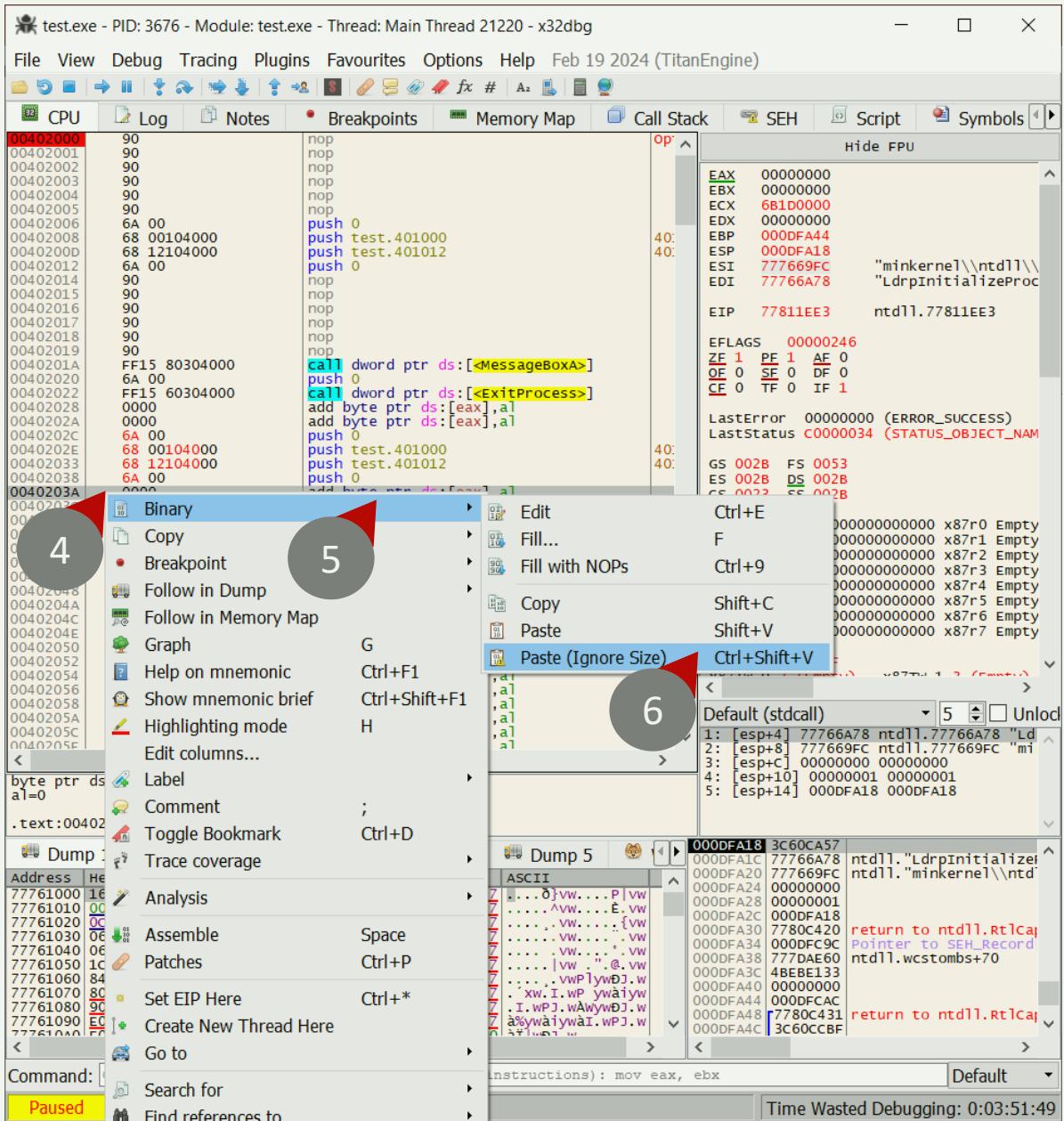


Copiați codul de la adresa 00402006 - 00402012 și inserați codul la adresa 0040202C - 00402038

Copiem linia care apelează funcția MessageBox:



Clonăm această linie imediat după codul clonat anterior:



Avem cod clonat care nu poate fi accesat (ExitProcess):

```
00402000 90          nop
00402001 90          nop
00402002 90          nop
00402003 90          nop
00402004 90          nop
00402005 90          nop
00402006 6A 00        push 0
00402008 68 00104000  push test.401000
0040200D 68 12104000  push test.401012
00402012 6A 00        push 0
00402014 90          nop
00402015 90          nop
00402016 90          nop
00402017 90          nop
00402018 90          nop
00402019 90          nop
0040201A FF15 80304000 call dword ptr ds:[<MessageBoxA>]
00402020 6A 00        push 0
00402022 FF15 60304000 call dword ptr ds:[<ExitProcess>]
00402028 0000          add byte ptr ds:[eax],al
0040202A 0000          add byte ptr ds:[eax],al
0040202C 6A 00        push 0
0040202E 68 00104000  push test.401000
00402033 68 12104000  push test.401012
00402038 6A 00        push 0
0040203A FF15 80304000 call dword ptr ds:[<MessageBoxA>]
00402040 0000          add byte ptr ds:[eax],al
00402042 0000          add byte ptr ds:[eax],al
00402044 0000          add byte ptr ds:[eax],al
00402046 0000          add byte ptr ds:[eax],al
00402048 0000          add byte ptr ds:[eax],al
0040204A 0000          add byte ptr ds:[eax],al
0040204C 0000          add byte ptr ds:[eax],al
0040204E 0000          add byte ptr ds:[eax],al
00402050 0000          add byte ptr ds:[eax],al
00402052 0000          add byte ptr ds:[eax],al
00402054 0000          add byte ptr ds:[eax],al
00402056 0000          add byte ptr ds:[eax],al
00402058 0000          add byte ptr ds:[eax],al
0040205A 0000          add byte ptr ds:[eax],al
0040205C 0000          add byte ptr ds:[eax],al
0040205E 0000          add byte ptr ds:[eax],al
00402060 0000          add byte ptr ds:[eax],al
00402062 0000          add byte ptr ds:[eax],al
```

Copiați linia de cod de la adresa 0040201A și inserați linia de cod la adresa 0040203A.

Copiem adresa de unde începe codul clonat:

The screenshot shows the assembly view of the debugger. A context menu is open over the assembly code. Numbered circles indicate specific steps: '3' points to the 'Copy' option in the context menu, and '4' points to the address 0040201A in the assembly code. The assembly code includes several pushes of addresses (e.g., 0040201A, 0040203A) and calls to MessageBoxA and ExitProcess.

File View Debug Tracing Plugins Favourites Options Help Feb 19 2024 (TitanEngine)

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Hide FPU

00402000 90 nop
00402001 90 nop
00402002 90 nop
00402003 90 nop
00402004 90 nop
00402005 90 nop
00402006 6A 00 push 0
00402008 68 00104000 push test.401000
0040200D 68 12104000 push test.401012
00402012 6A 00 push 0
00402014 90 nop
00402015 90 nop
00402016 90 nop
00402017 90 nop
00402018 90 nop
00402019 90 nop
0040201A FF15 80304000 call dword ptr ds:[<MessageBoxA>]
00402020 6A 00 push 0
00402022 FF15 60304000 call dword ptr ds:[<ExitProcess>]
00402028 0000 add byte ptr ds:[eax],al
0040202A 0000 add byte ptr ds:[eax],al
0040202C 6A 00 push 0
0040202E 68 00104000 push test.401000
00402033 68 12104000 push test.401012
00402038 6A 00 push 0
0040203A FF15 80304000 call dword ptr ds:[<MessageBoxA>]

Binary Copy Selection Ctrl+C
Selection to File
Selection (No Bytes)
Selection to File (No Bytes)
Address Alt+Ins
RVA
File
Header
Disassembly

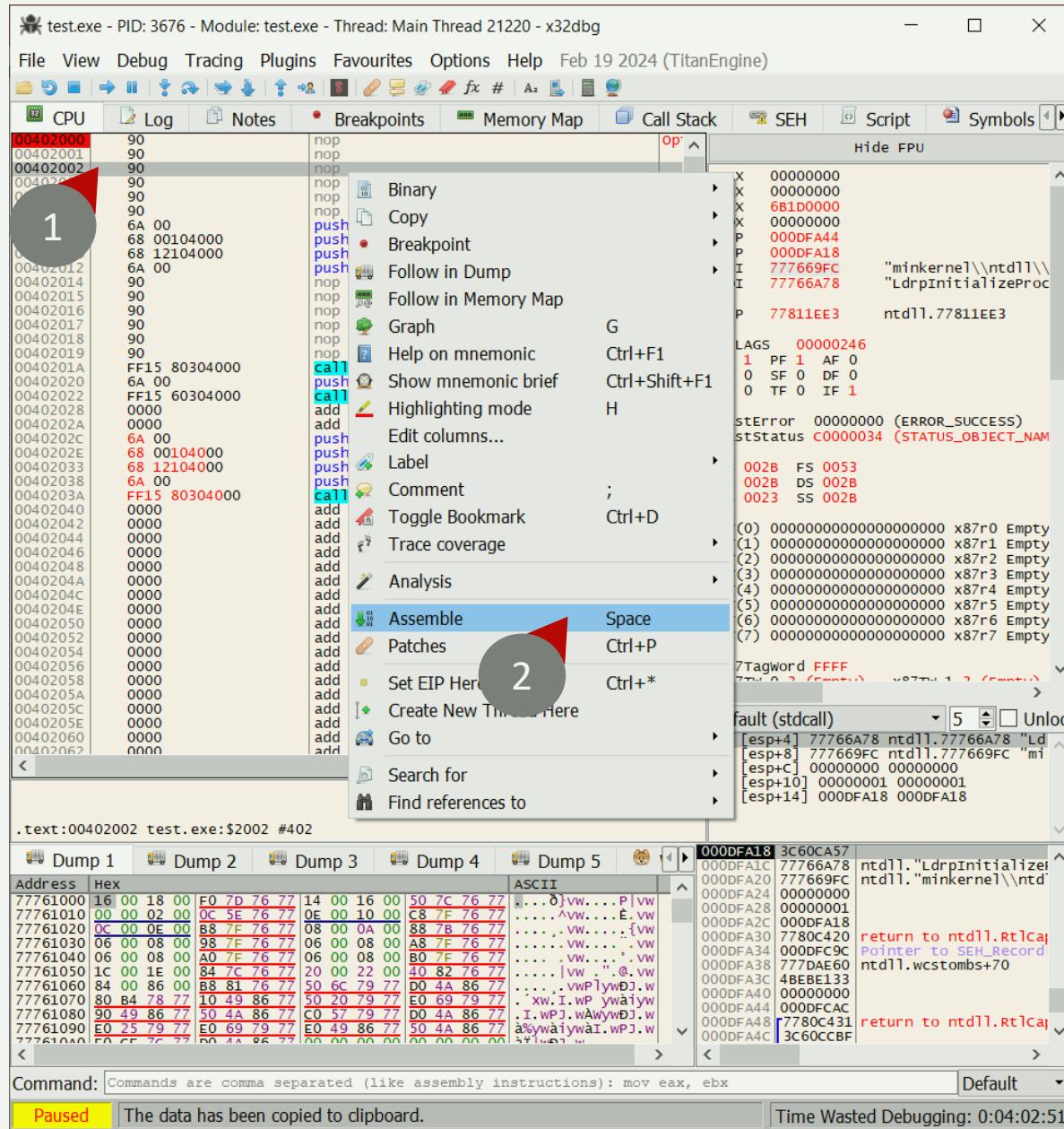
Graph
Help on mnemonic Ctrl+F1
Show mnemonic brief Ctrl+Shift+F1
Highlighting mode H
Edit columns...
Label
Comment ;
Toggle Bookmark Ctrl+D
Trace coverage

Analysis
Assemble Space
Patches Ctrl+P
Set EIP Here Ctrl+\*
Create New Thread Here
Go to
Search for
Find references to

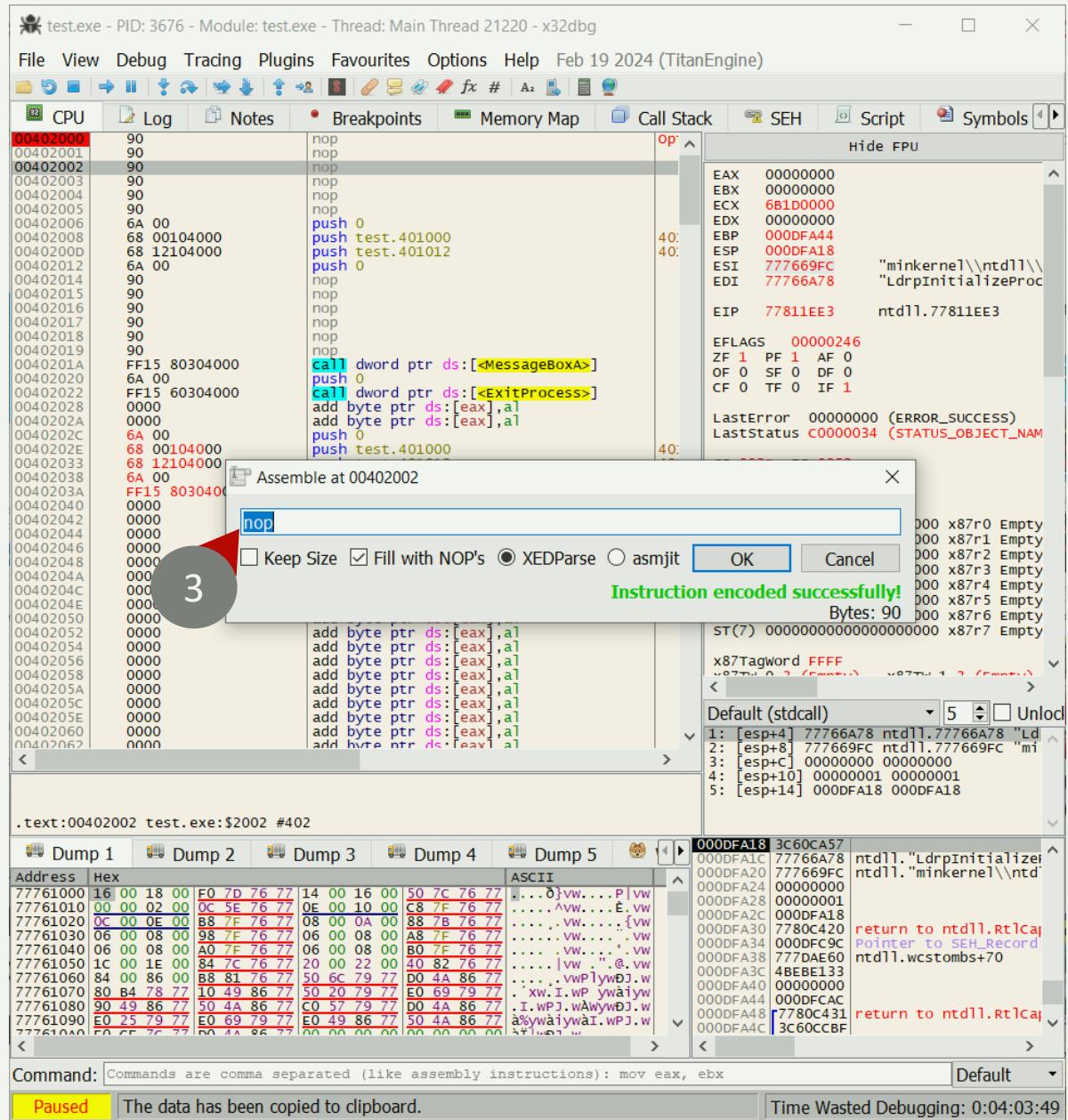
Command: Commands are copied to clipboard. Paused The data has been copied to clipboard.

Time Wasted Debugging: 0:04:08:19

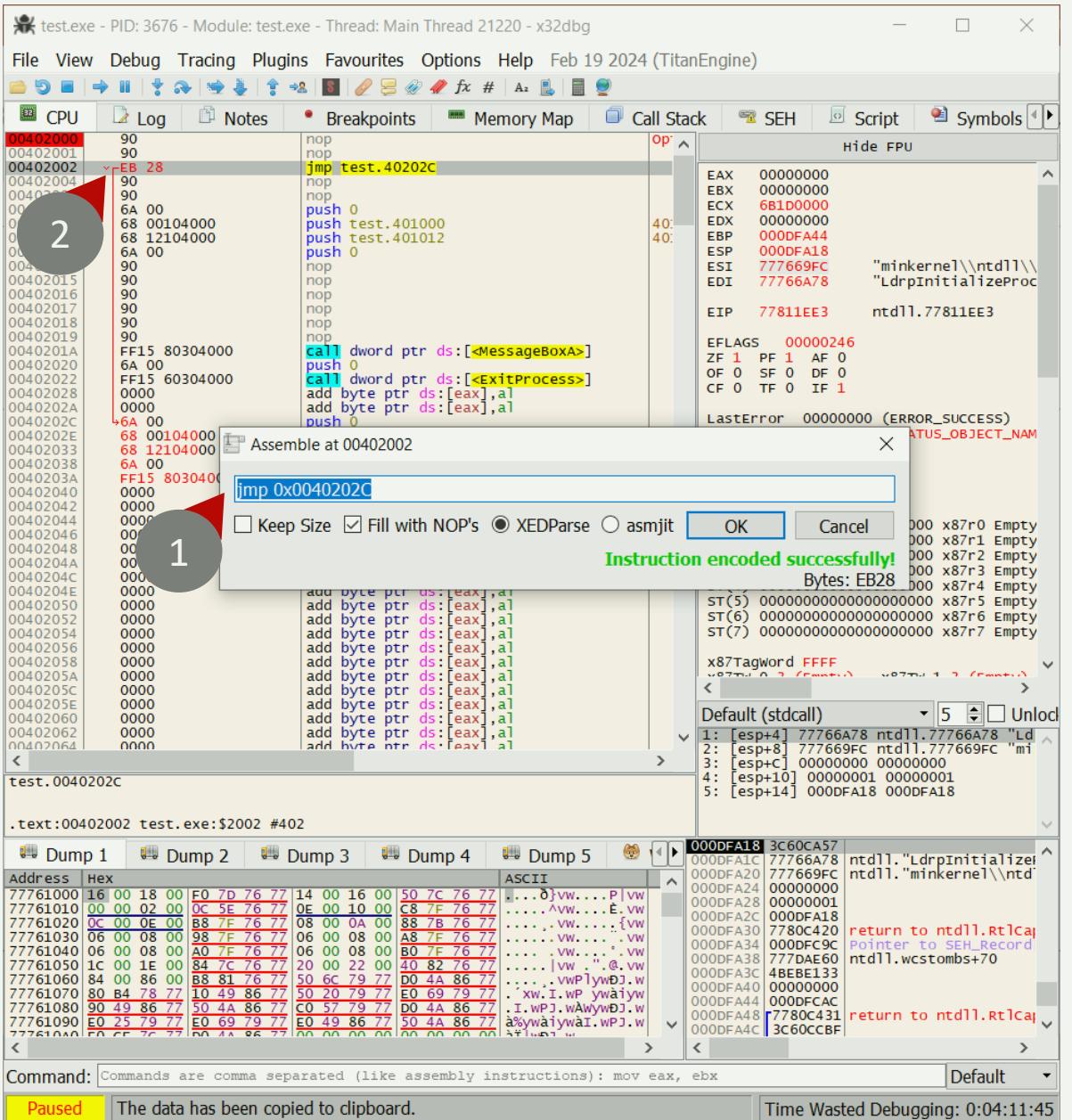
Solicităm o introducere cu instrucțiuni:



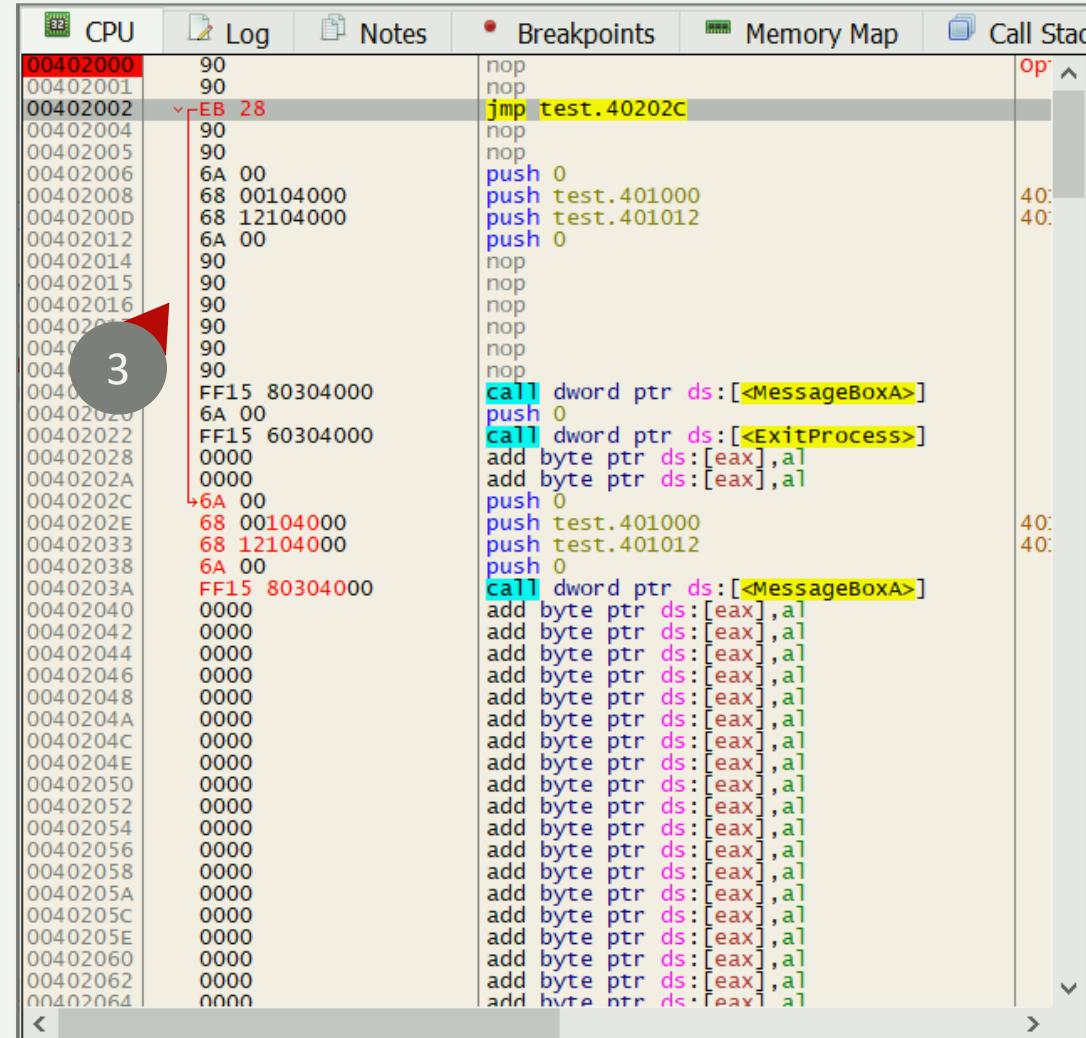
Apare fereastra unde instrucțunea **nop** este înlocuită cu **jmp**:



Introducem un salt necondiționat la adresa copiată anterior:

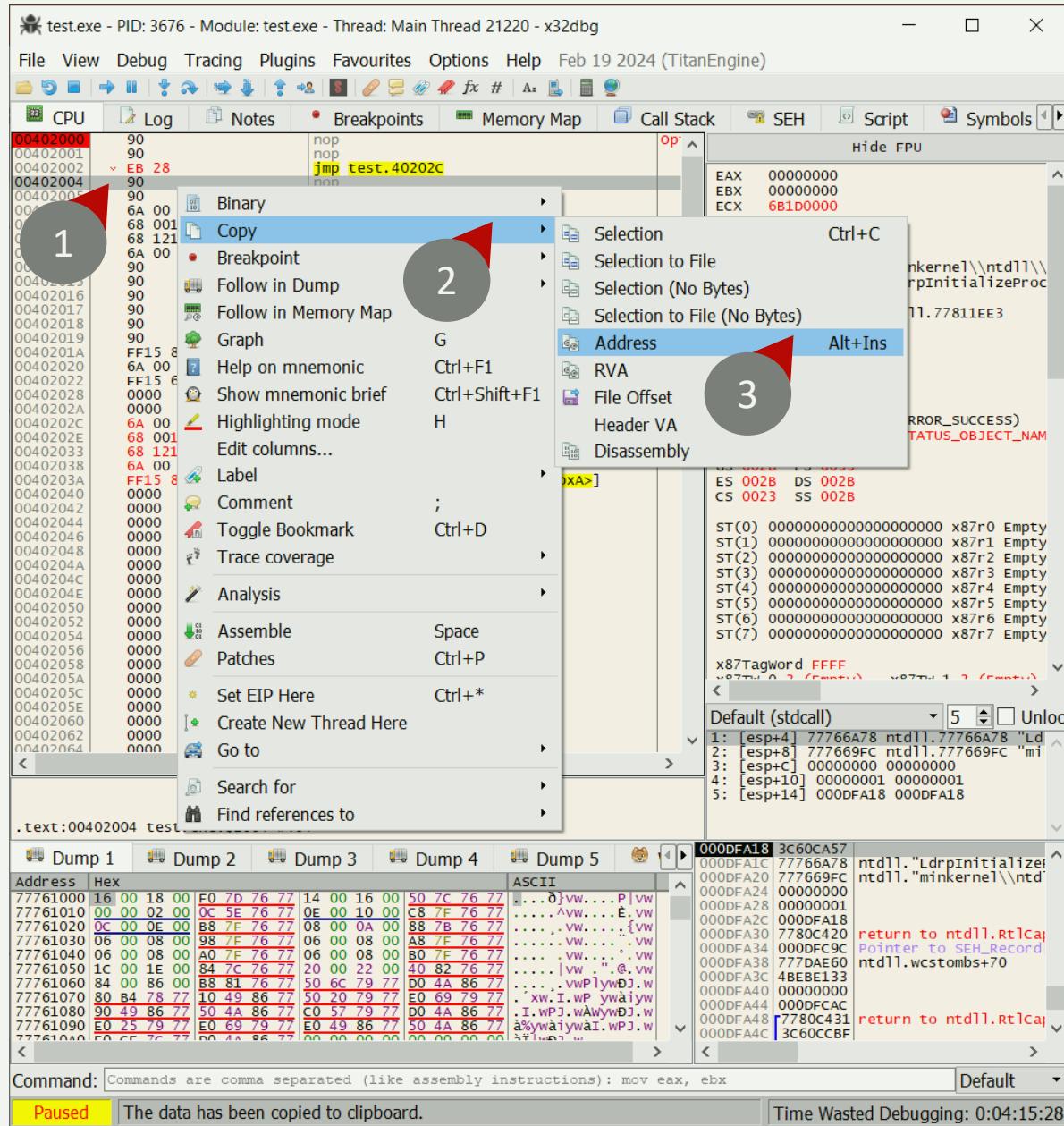


Clonăm această linie imediat după codul clonat anterior:

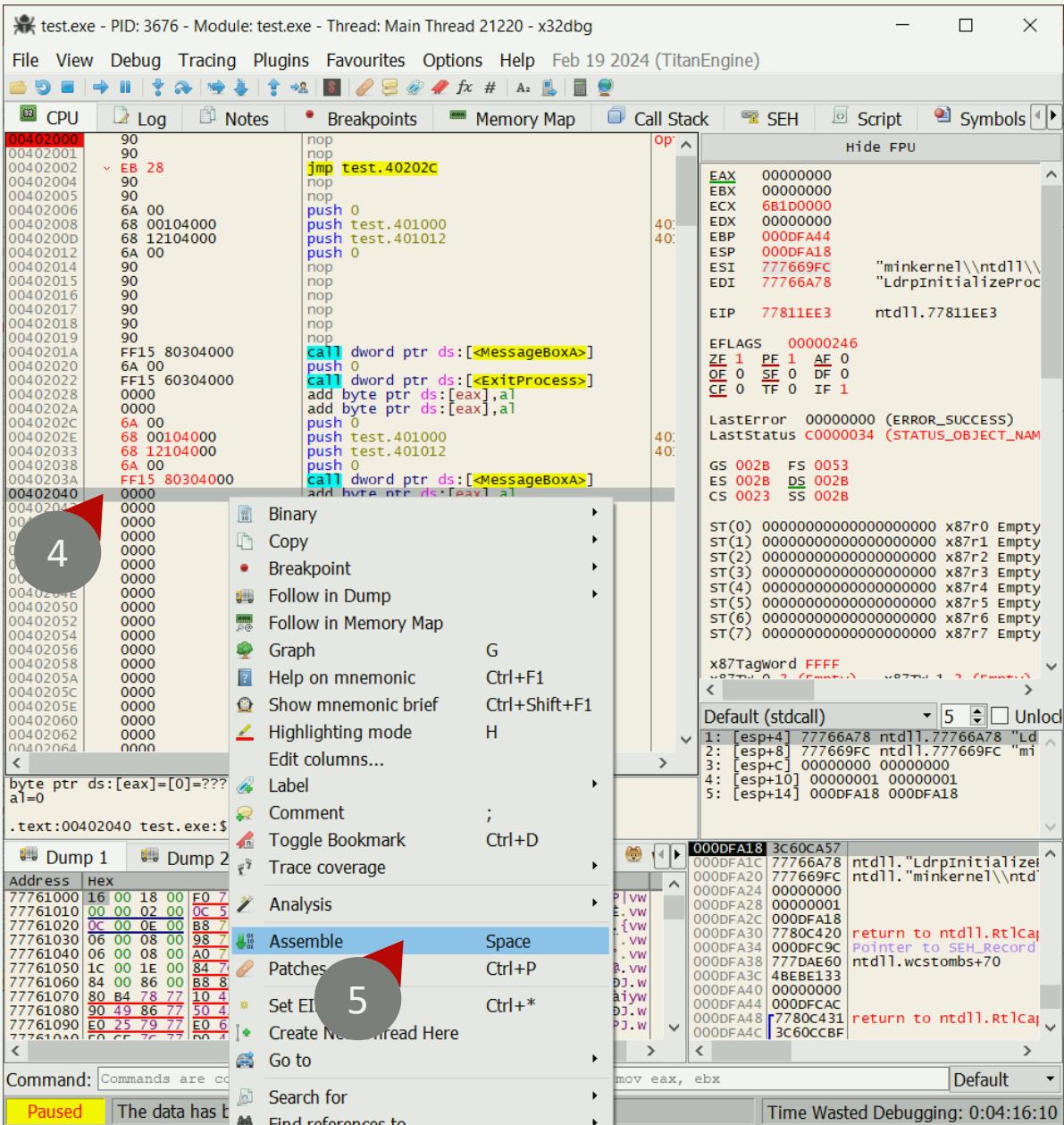


Facem un salt de la adresa 00402002 la adresa 0040202C.

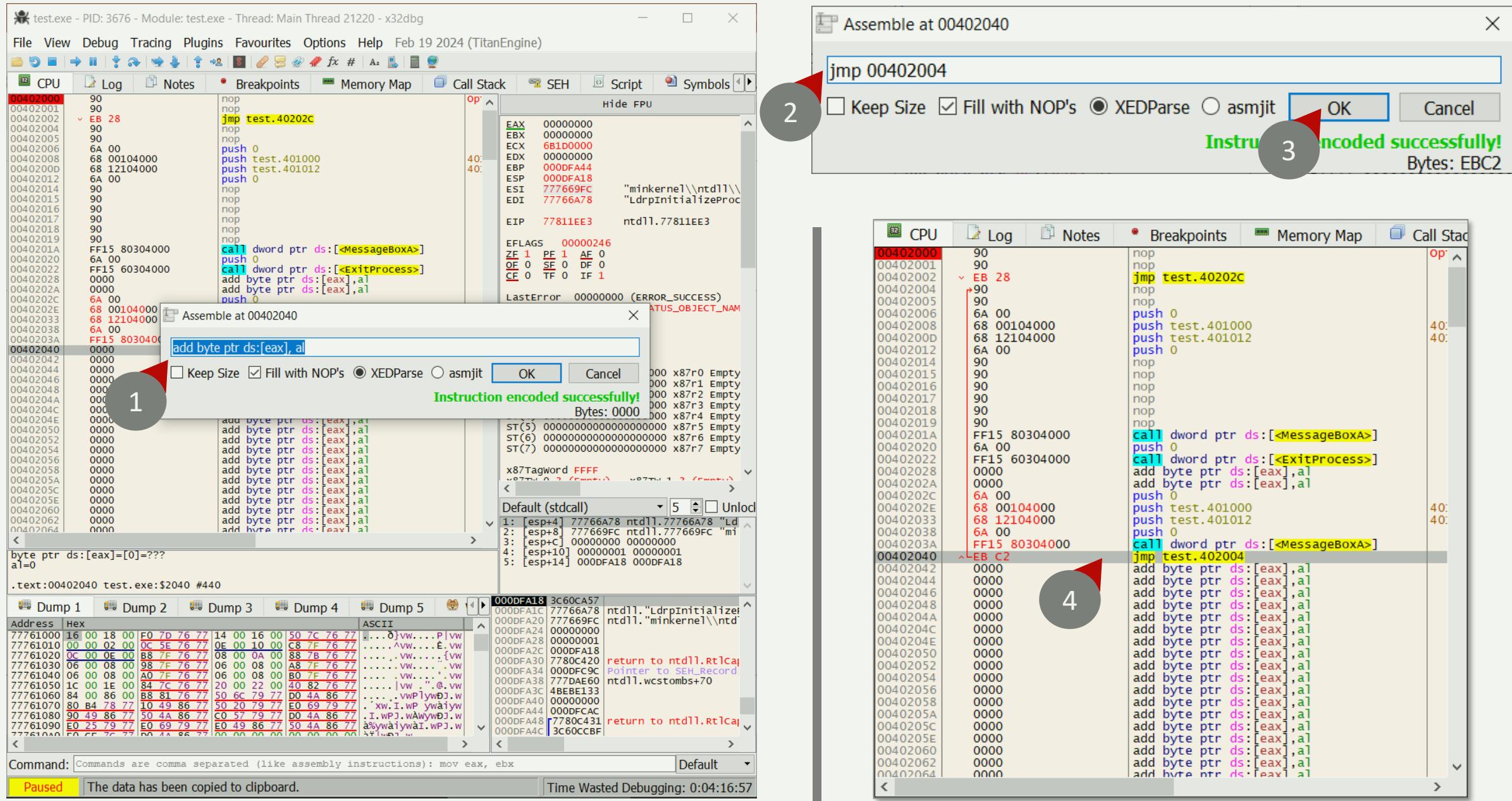
Adresa după saltul initial la codul nostru este copiată:



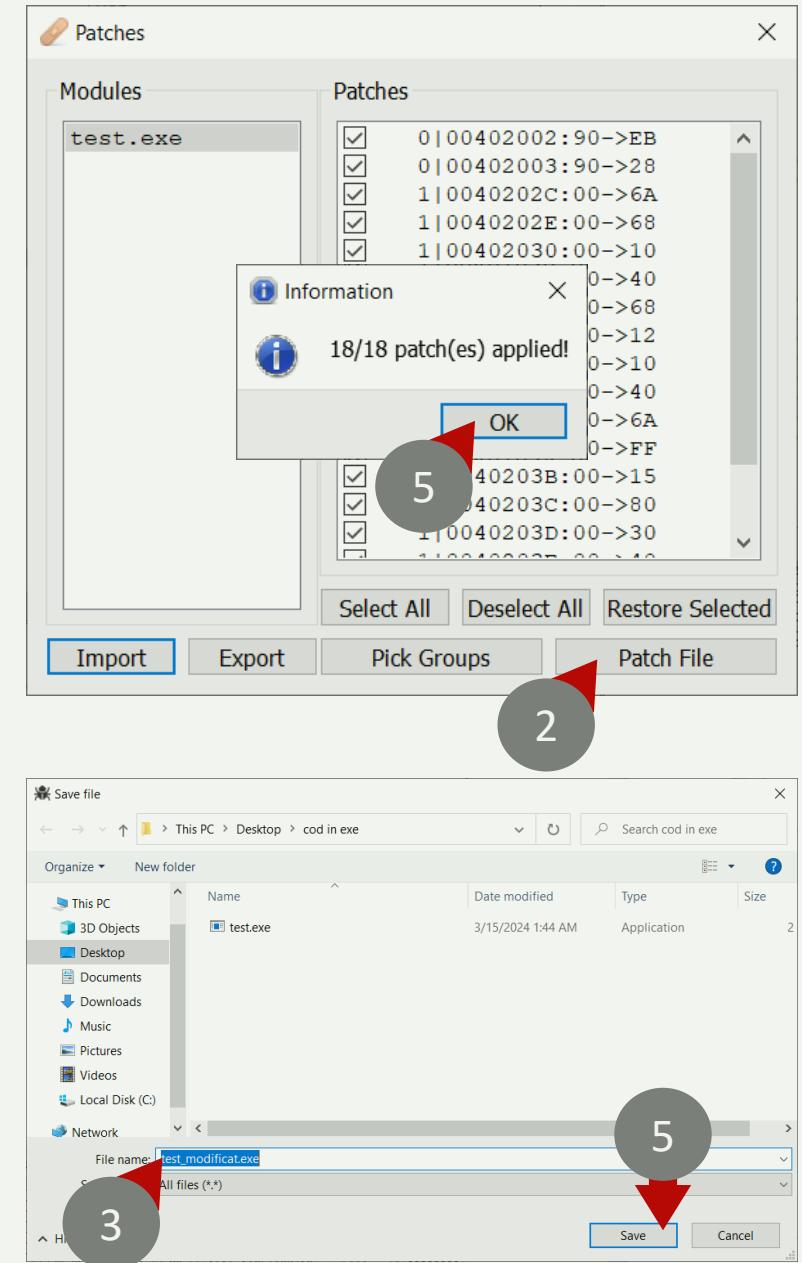
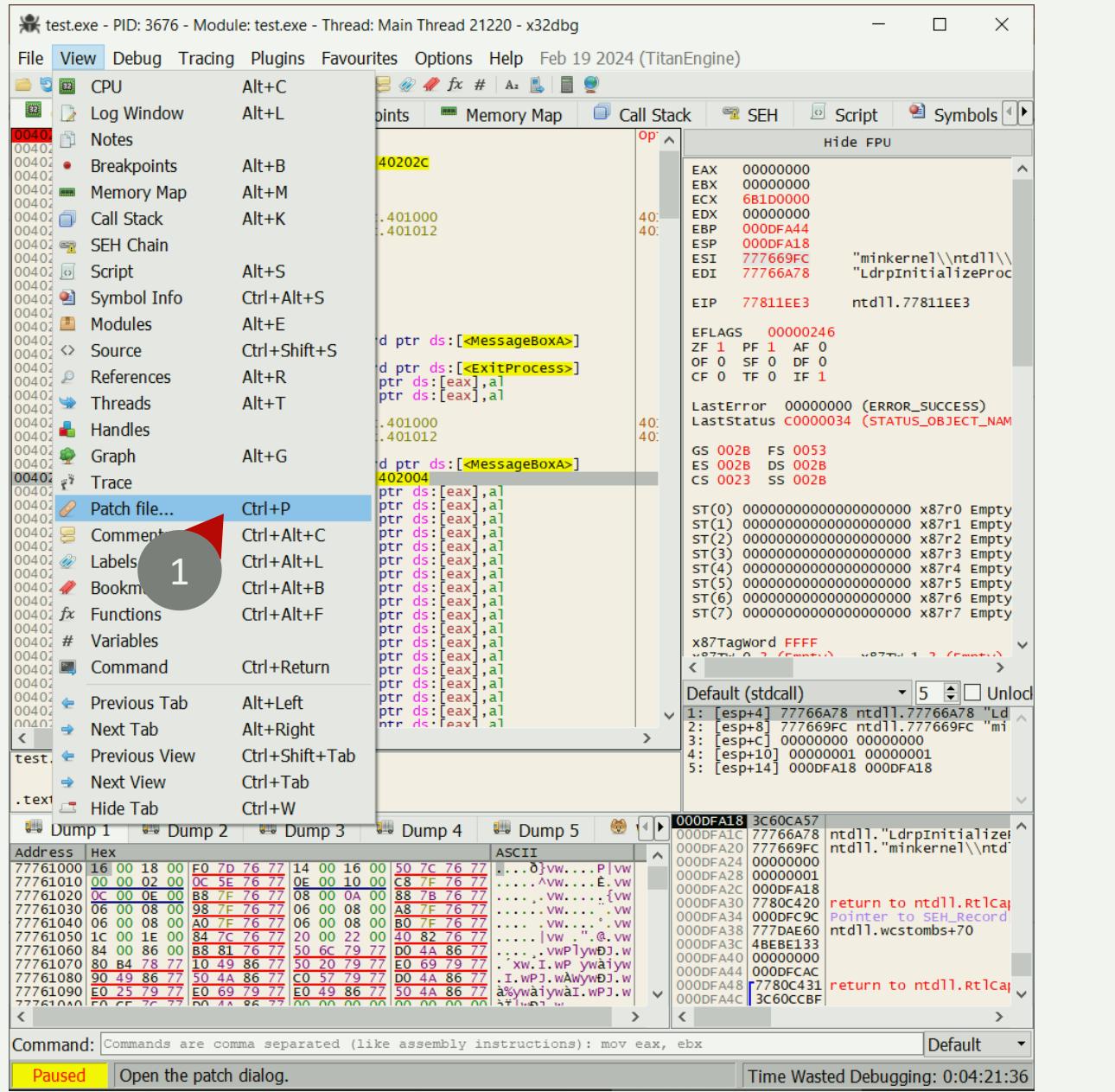
După codul clonat, se solicită o introducere de instrucțiuni:



Codul redundant este înlocuit cu un salt la adresa copiată:



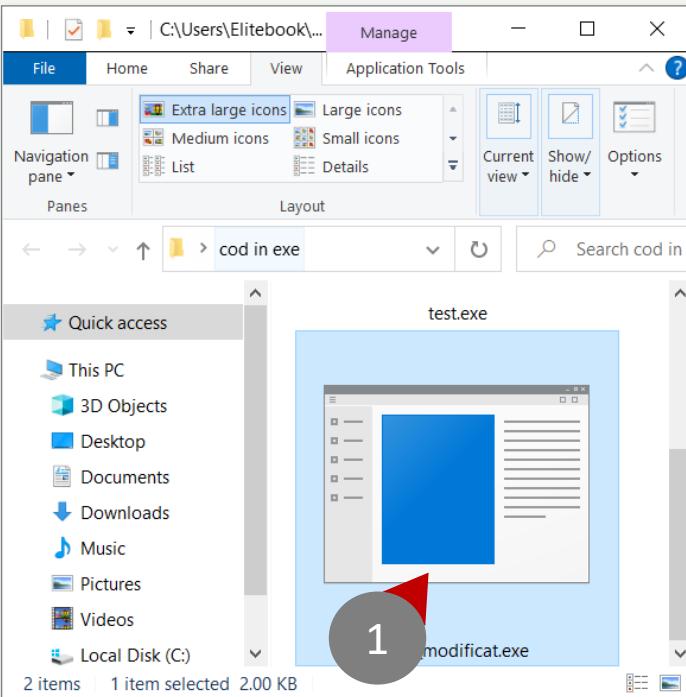
Noua versiune a executabilului este salvată:



C.8.4

# ADAUGAREA DE STRUCTURI DE DATE IN EXECUTABILE





1. Schimbați punctul de intrare la un alt segment
2. Rulați segmentul în regiunea selectată
3. Salt la adresa de sub punctul de intrare

Selectați adresa și urmărim eticheta în *dump* la adresa specificată:

```

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols
00402000 90 nop
00402001 90 nop
00402002 EB 28 jmp test_modificat.40202C
00402004 90 nop
00402005 90 nop
00402006 6A 00 push 0
00402008 68 00104000 push test_modificat.401000
0040200D 68 12104000 push test_modificat.401012
00402012 6A 00 push 0
00402014 90 nop
00402015 90 nop
00402016 90 nop
00402017 90 nop
00402018 90 nop
00402019 90 nop
0040201A FF15 80304000 call dword ptr ds:[MessageBoxA]
00402020 6A 00 push 0
00402022 FF15 60304000 call dword ptr ds:[ExitProcess]
00402028 0000 add byte ptr ds:[eax],al
0040202A 0000 add byte ptr ds:[eax],al
0040202C 6A 00 push 0
0040202E 68 00104000 push test_modificat.401000
0040202F 68 00104000 push test_modificat.401012
Binary Copy Breakpoint Follow in Dump Follow in Disassembler Follow in Memory Map Graph Help on mnemonic Show mnemonic brief Ctrl+F1 Ctrl+Shift+F1 H Highlighting mode Edit columns... Label Comment Toggle Bookmark Trace coverage Analysis Assemble Patches Set EIP Here Ctrl+* Create New Thread Here Go to Search for Find references to

```

Selected Address: test\_modificat.00401000

Default (stdcall) 5 Unlocked

Dump 4 Dump 5

000DFA18 6EAC2CD1 nt!LdrpInitialize nt!minkernel\ntdll

000DFA1C 77766A78 nt!L77766A78 "Ld

000DFA20 777669FC nt!L777669FC "mi

000DFA28 00000001 nt!wcstombs+70

000DFA2C 000DFA18

000DFA30 7780C420

000DFA34 000DFC9C

000DFA38 777DAE60

000DFA3C 192707B5

000DFA40 00000000

000DFA44 000DFC4AC

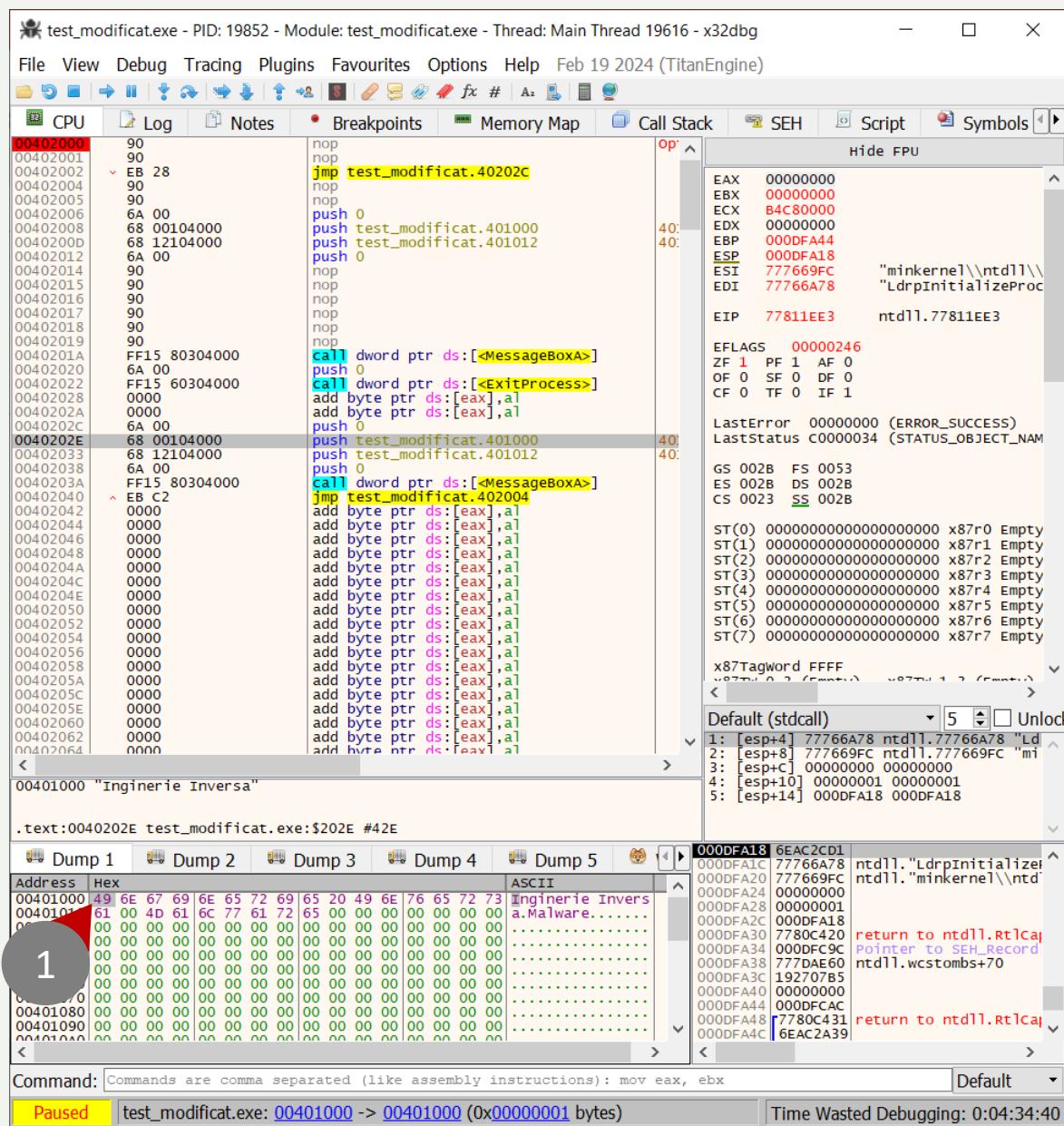
000DFA48 7780C431

000DFA4C 6EAC2A39

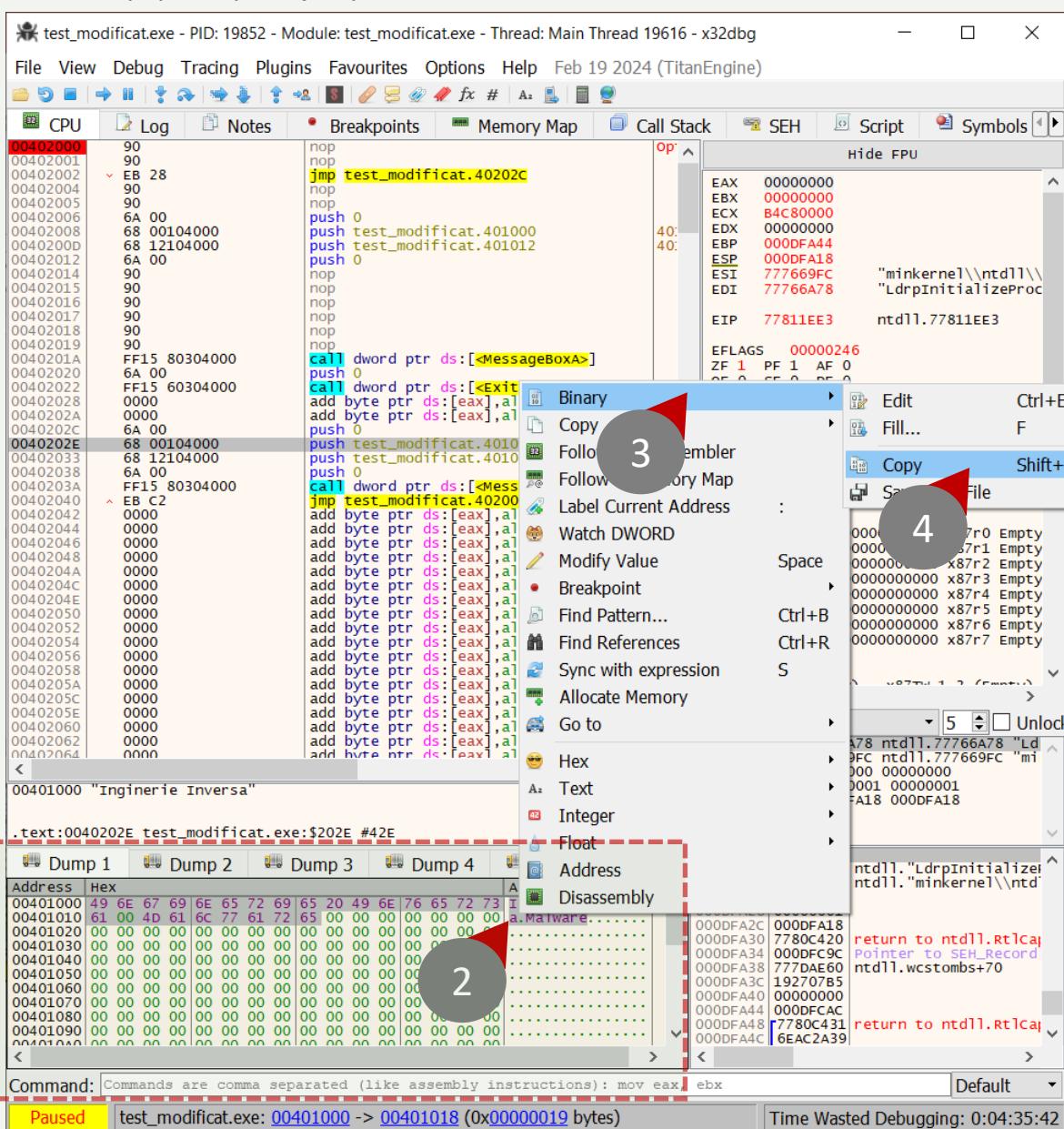
assembly instructions): mov eax, ebx

Time Wasted Debugging: 0:04:32:45

Ajungem la adresa 0040100 în *dump* și observăm sirul:



Selectati sirul si copiati datele:



Găsim zona liberă și dublăm datele folosind comanda paste:

The screenshot shows the x32dbg debugger interface with the following details:

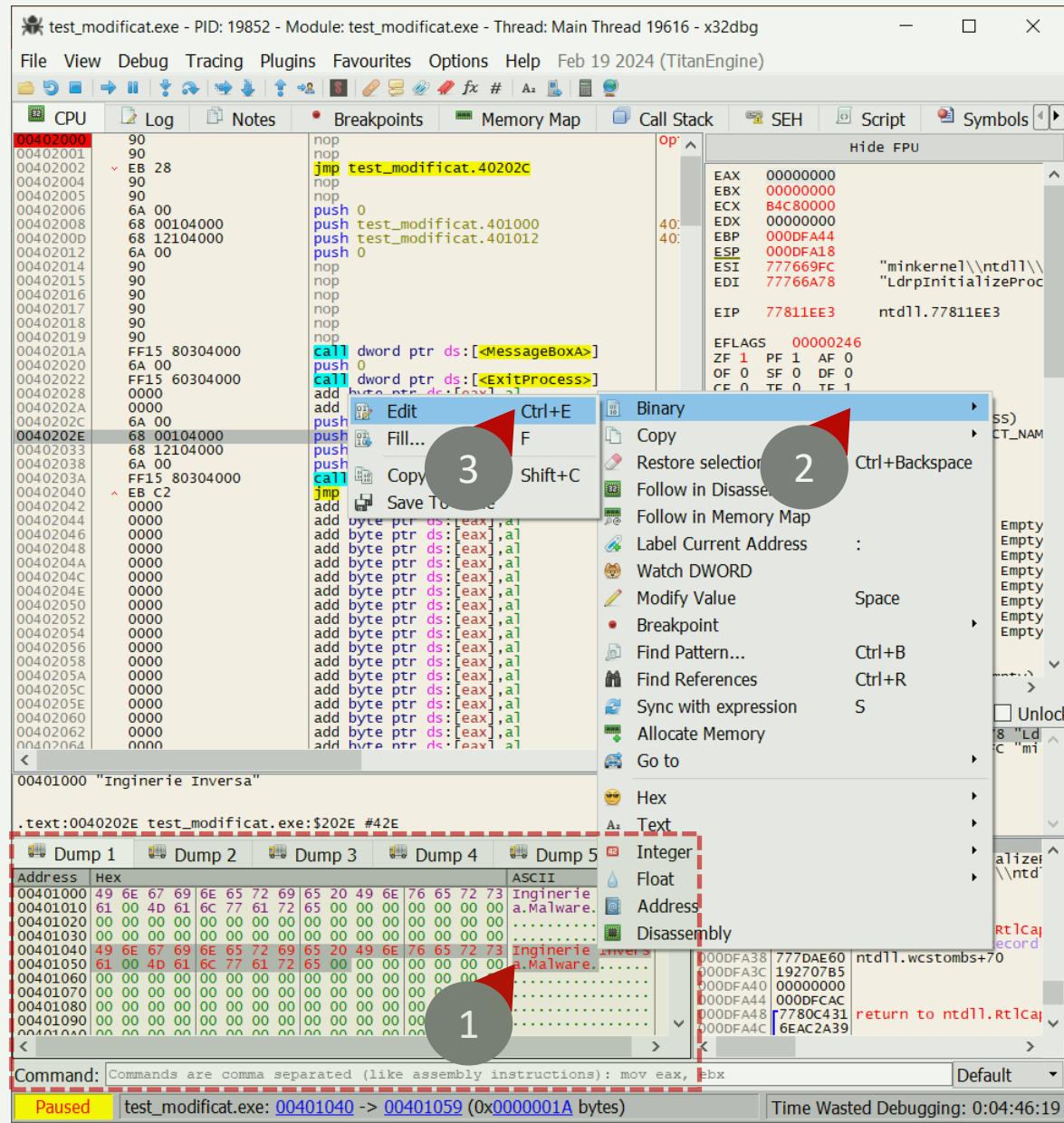
- Top Bar:** test\_modificat.exe - PID: 19852 - Module: test\_modificat.exe - Thread: Main Thread 19616 - x32dbg
- Menu Bar:** File View Debug Tracing Plugins Favourites Options Help Feb 19 2024 (TitanEngine)
- Toolbars:** CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols
- Registers:** EAX, EBX, ECX, EDX, EBP, ESP, ESI, EDI, EIP, EFLAGS, Last Error, CPU Registers, Stack Registers.
- Stack:** Shows memory dump from 00401000 to 00401050, highlighting assembly instructions like `jmp test_modificat.40202C` and `call dword ptr ds:[<MessageBoxA>]`.
- Context Menu (Open at 00401040):**
  - Binary (highlighted with circle 2)
  - Copy
  - Follow in Disass
  - Follow in Memory
  - Label Current Address
  - Watch DWORD
  - Modify Value
  - Breakpoint
  - Find Pattern...
  - Find References
  - Sync with expression
  - Allocate Memory
  - Go to
  - Space
  - Ctrl+B
  - Ctrl+R
  - S
- Assembly View:** Shows assembly code for the current thread, including `jmp test_modificat.40202C`, `push 0`, `push test_modificat.401000`, `push test_modificat.401012`, `push 0`, `nop`, `push 0`, `call dword ptr ds:[<MessageBoxA>]`, `push 0`, `call dword ptr ds:[<ExitProcess>]`, `add byte ptr ds:[eax],al`, and `add byte ptr ds:[eax],al`.
- Registers View:** Shows CPU registers (EAX-EIP) and stack registers (ESI-EDI).
- Registers View (Bottom):** Shows CPU Registers, Stack Registers, and Stack Dump.
- Registers View (Bottom Left):** Shows Dump 4 and Dump 5.
- Registers View (Bottom Right):** Shows Return Address, CPU Registers, Stack Registers, and Stack Dump.
- Bottom Bar:** Command: Commands are comma separated (like assembly instructions): mov eax, ebx, Default
- Status Bar:** Paused test\_modificat.exe: 00401040 -> 00401040 (0x00000001 bytes), Time Wasted Debugging: 0:04:37:50

Avem date clonate la diferite adrese:

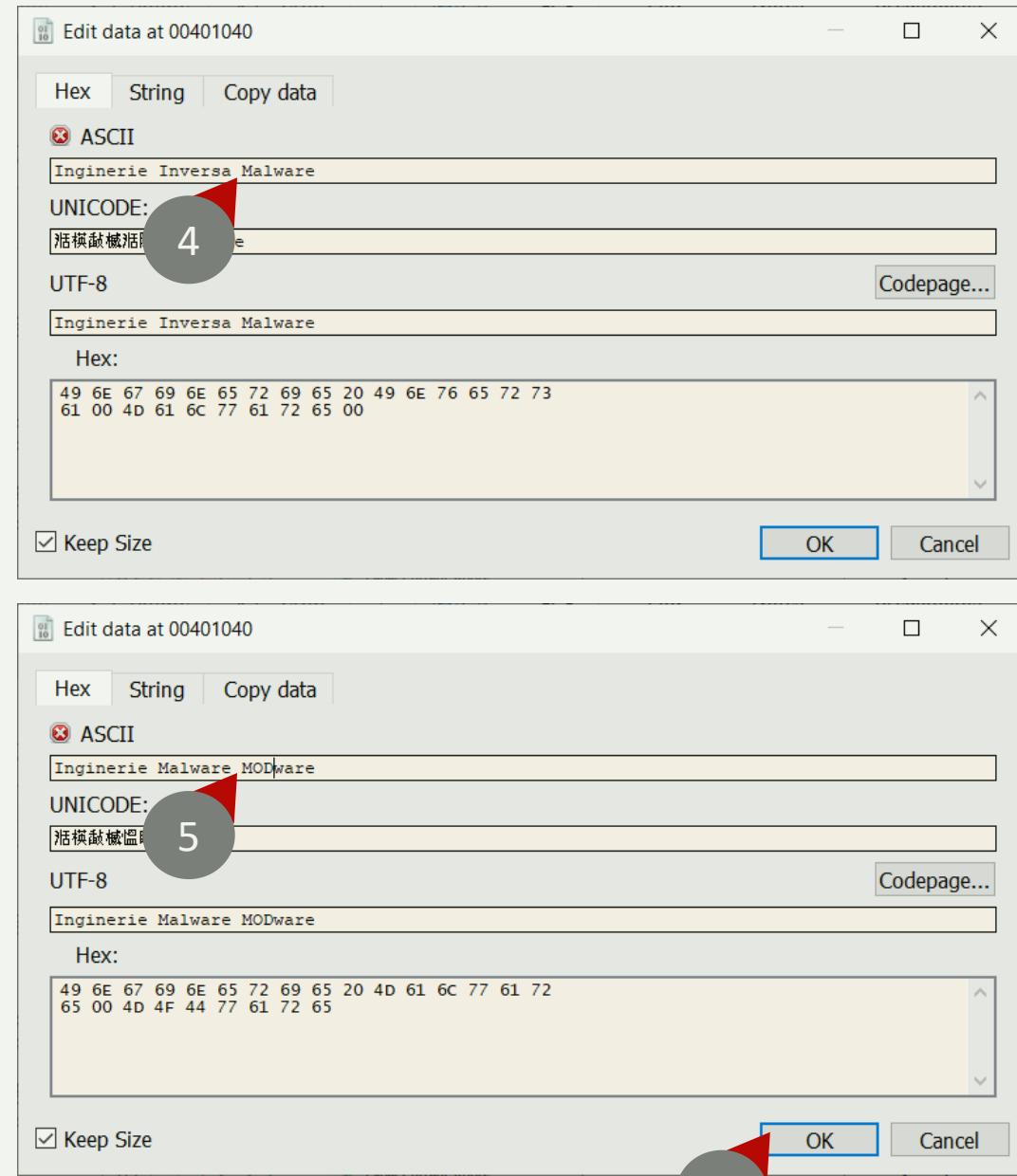
Clonăm datele pentru a diversifica argumentele pentru prima și a doua casetă de mesaj.



Editați sirul în dump la noua adresă unde se află clona:



Schimbați sirul, dar mențineți dimensiunea pentru moment:

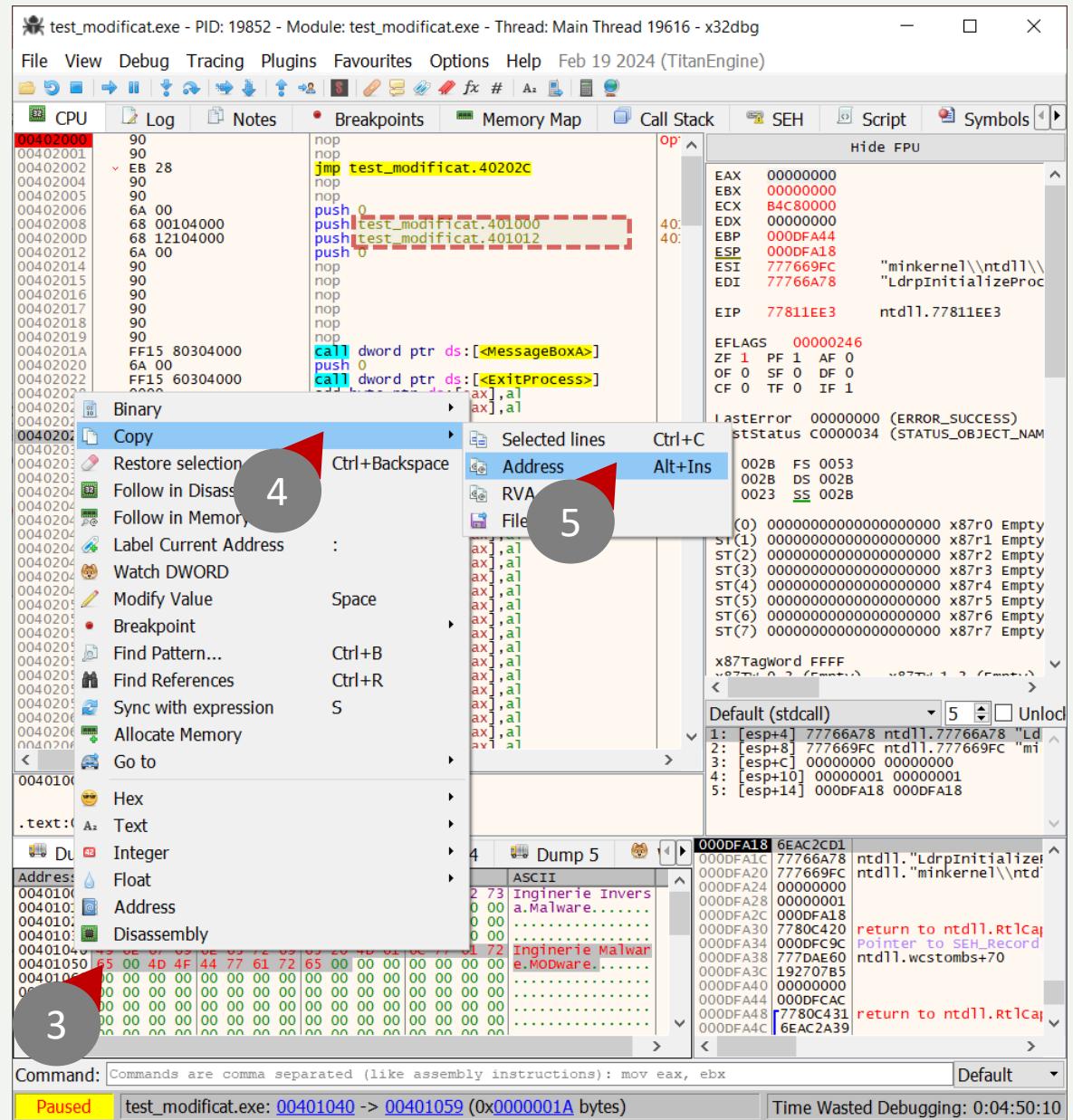


Ambele șiruri din clonă sunt editate:

- “Inginierie Inversa” devine “Inginierie Malware”
  - “Malware” devine “MODware”

Putem scrie un sir mai mare atata timp cat deplasam adresa pentru cea de-a doua eticheta!

Copiați adresa la care se află șirul din clonă:



Schimbăm adresa etichetei originale cu adresa clonei modificate:

The screenshot shows the x32dbg debugger interface. The assembly window displays the following code:

```
00402000 90          nop
00402001 90          nop
00402002 EB 28        jmp test_modificat.40202C
00402003 90          nop
00402004 90          nop
00402005 90          nop
00402006 6A 00        push 0
00402007 68 00104000  push test_modificat.401000
00402008 68 12104000  push test_modificat.401012
00402009 6A 00        push 0
00402010 90          nop
00402011 90          nop
00402012 90          nop
00402013 90          nop
00402014 90          nop
00402015 90          nop
00402016 90          nop
00402017 90          nop
00402018 90          nop
00402019 90          nop
0040201A FF15 80304000 call dword ptr ds:[<MessageBoxA>]
0040201B 6A 00        push 0
0040201C FF15 60304000 call dword ptr ds:[<ExitProcess>]
0040201D 0000          add byte ptr ds:[eax],al
0040201E 0000          add byte ptr ds:[eax],al
0040201F 6A 00        push 0
00402020 68 00104000  push test_modificat.401000
00402021 68 12104000  push test_modificat.401012
00402022 6A 00        push 0
00402023 EB C2        jmp test_modificat.402004
00402024 0000          add byte ptr ds:[eax],al
00402025 0000          add byte ptr ds:[eax],al
00402026 0000          add byte ptr ds:[eax],al
00402027 0000          add byte ptr ds:[eax],al
00402028 0000          add byte ptr ds:[eax],al
00402029 0000          add byte ptr ds:[eax],al
0040202A 0000          add byte ptr ds:[eax],al
0040202B 0000          add byte ptr ds:[eax],al
0040202C 0000          add byte ptr ds:[eax],al
0040202D 0000          add byte ptr ds:[eax],al
0040202E 0000          add byte ptr ds:[eax],al
0040202F 0000          add byte ptr ds:[eax],al
00402030 0000          add byte ptr ds:[eax],al
00402031 0000          add byte ptr ds:[eax],al
00402032 0000          add byte ptr ds:[eax],al
00402033 0000          add byte ptr ds:[eax],al
00402034 0000          add byte ptr ds:[eax],al
00402035 0000          add byte ptr ds:[eax],al
00402036 0000          add byte ptr ds:[eax],al
00402037 0000          add byte ptr ds:[eax],al
00402038 0000          add byte ptr ds:[eax],al
00402039 0000          add byte ptr ds:[eax],al
0040203A EB 00        jmp test_modificat.402004
0040203B 0000          add byte ptr ds:[eax],al
0040203C 0000          add byte ptr ds:[eax],al
0040203D 0000          add byte ptr ds:[eax],al
0040203E 0000          add byte ptr ds:[eax],al
0040203F 0000          add byte ptr ds:[eax],al
00402040 0000          add byte ptr ds:[eax],al
00402041 0000          add byte ptr ds:[eax],al
00402042 0000          add byte ptr ds:[eax],al
00402043 0000          add byte ptr ds:[eax],al
00402044 0000          add byte ptr ds:[eax],al
00402045 0000          add byte ptr ds:[eax],al
00402046 0000          add byte ptr ds:[eax],al
00402047 0000          add byte ptr ds:[eax],al
00402048 0000          add byte ptr ds:[eax],al
00402049 0000          add byte ptr ds:[eax],al
0040204A 0000          add byte ptr ds:[eax],al
0040204B 0000          add byte ptr ds:[eax],al
0040204C 0000          add byte ptr ds:[eax],al
0040204D 0000          add byte ptr ds:[eax],al
0040204E 0000          add byte ptr ds:[eax],al
0040204F 0000          add byte ptr ds:[eax],al
00402050 0000          add byte ptr ds:[eax],al
00402051 0000          add byte ptr ds:[eax],al
00402052 0000          add byte ptr ds:[eax],al
00402053 0000          add byte ptr ds:[eax],al
00402054 0000          add byte ptr ds:[eax],al
00402055 0000          add byte ptr ds:[eax],al
00402056 0000          add byte ptr ds:[eax],al
00402057 0000          add byte ptr ds:[eax],al
00402058 0000          add byte ptr ds:[eax],al
00402059 0000          add byte ptr ds:[eax],al
0040205A 0000          add byte ptr ds:[eax],al
0040205B 0000          add byte ptr ds:[eax],al
0040205C 0000          add byte ptr ds:[eax],al
0040205D 0000          add byte ptr ds:[eax],al
0040205E 0000          add byte ptr ds:[eax],al
0040205F 0000          add byte ptr ds:[eax],al
00402060 0000          add byte ptr ds:[eax],al
00402061 0000          add byte ptr ds:[eax],al
00402062 0000          add byte ptr ds:[eax],al
00402063 0000          add byte ptr ds:[eax],al
00402064 0000          add byte ptr ds:[eax],al
```

The context menu (number 2) is open over the assembly code, with "Assemble" selected.

The assembly window title bar says: "test\_modificat.exe - PID: 19852 - Module: test\_modificat.exe - Thread: Main Thread 19616 - x32dbg".

The status bar at the bottom says: "Paused" and "The data has been copied to clipboard." Time Wasted Debugging: 0:04:51:54.

Adresa originală este înlocuită cu cea nouă a clonei:

The screenshot shows two assembly windows in the x32dbg debugger.

The top window is titled "Assemble at 0040202E" and contains the instruction: `push 0x401000`. It includes options: "Keep Size" (unchecked), "Fill with NOP's" (checked), "XEDParse" (radio button selected), and "asmjit" (radio button unselected). Buttons for "OK" and "Cancel" are shown. A message at the bottom says: "Instruction encoded successfully! Bytes: 6800104000".

The bottom window is titled "Assemble at 0040202E" and contains the instruction: `push 00401040`. It includes the same set of options. A message at the bottom says: "Instruction encoded successfully! Bytes: 6840104000".

A large cartoon character with the number 32 on its chest is positioned between the two windows.

Observați că noua adresă la al doilea argument este schimbată:

The screenshot shows the x32dbg debugger interface. The assembly tab displays the following code snippet:

```
00402000 90          nop
00402001 90          nop
00402002 EB 28        jmp test_modificat.40202C
00402004 90          nop
00402005 90          nop
00402006 6A 00        push 0
00402008 68 00104000  push test_modificat.401000
0040200D 68 12104000  push test_modificat.401012
00402012 6A 00        push 0
00402014 90          nop
00402015 90          nop
00402016 90          nop
00402017 90          nop
00402018 90          nop
00402019 90          nop
0040201A FF15 80304000 call dword ptr ds:[<MessageBoxA>]
00402020 6A 00        push 0
00402022 FF15 60304000 call dword ptr ds:[<ExitProcess>]
00402028 0000          add byte ptr ds:[eax],al
0040202A 0000          add byte ptr ds:[eax],al
0040202C 6A 00        push 0
0040202E 68 40104000  push test_modificat.401040
00402033 68 12104000  push test_modificat.401012
00402038 6A 00        push 0
0040203A EB C2        jmp test_modificat.402004
00402042 0000          add byte ptr ds:[eax],al
00402044 0000          add byte ptr ds:[eax],al
00402046 0000          add byte ptr ds:[eax],al
00402048 0000          add byte ptr ds:[eax],al
0040204A 0000          add byte ptr ds:[eax],al
0040204C 0000          add byte ptr ds:[eax],al
0040204E 0000          add byte ptr ds:[eax],al
00402050 0000          add byte ptr ds:[eax],al
00402052 0000          add byte ptr ds:[eax],al
00402054 0000          add byte ptr ds:[eax],al
00402056 0000          add byte ptr ds:[eax],al
00402058 0000          add byte ptr ds:[eax],al
0040205A 0000          add byte ptr ds:[eax],al
0040205C 0000          add byte ptr ds:[eax],al
0040205E 0000          add byte ptr ds:[eax],al
00402060 0000          add byte ptr ds:[eax],al
00402062 0000          add byte ptr ds:[eax],al
00402064 0000          add byte ptr ds:[eax],al
```

The memory dump tabs at the bottom show the state of memory at address 00401040, which contains the string "Inginerie Malware".

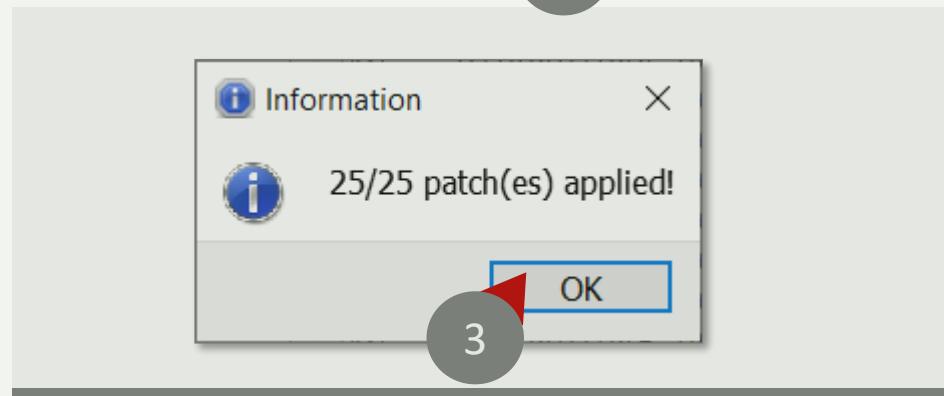
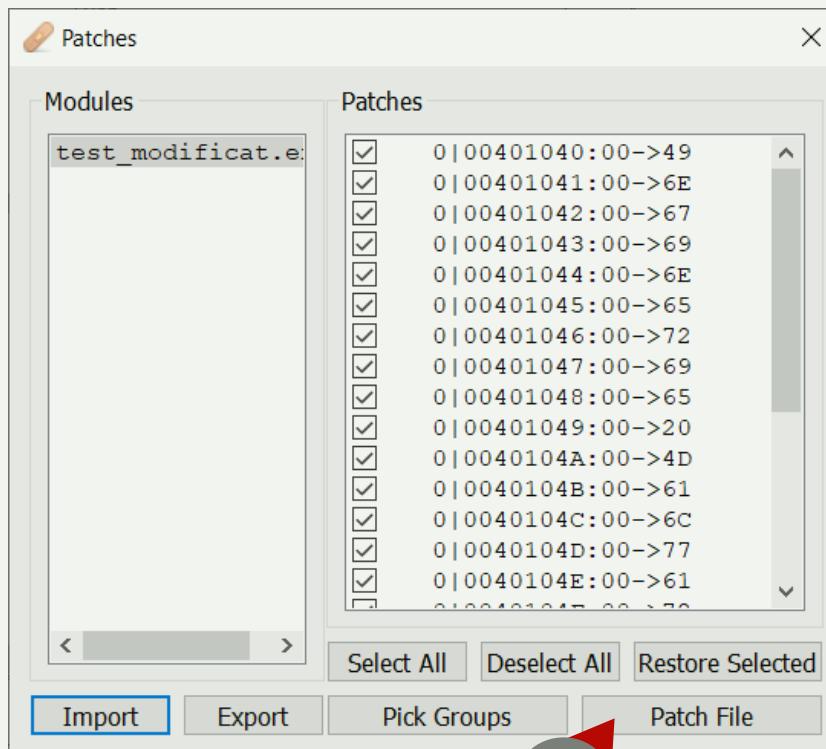
Peticim executabilul:

The screenshot shows the x32dbg debugger interface with the file menu open. The 'Patch file...' option is highlighted with a red arrow and a number 3.

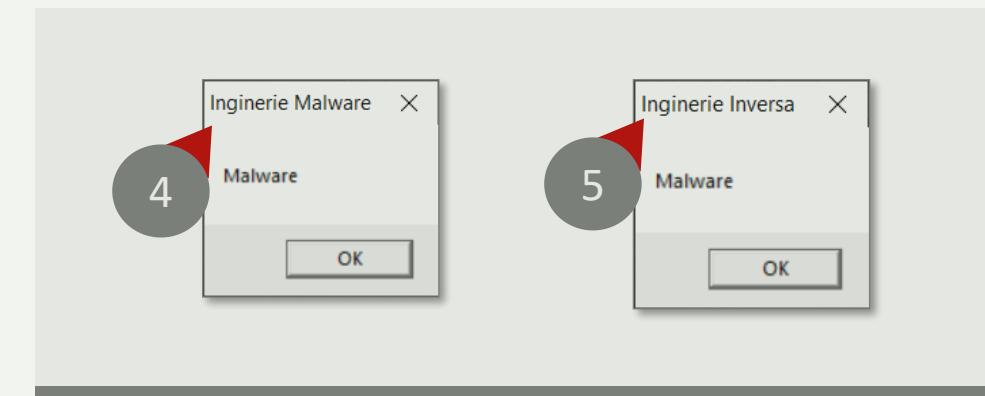
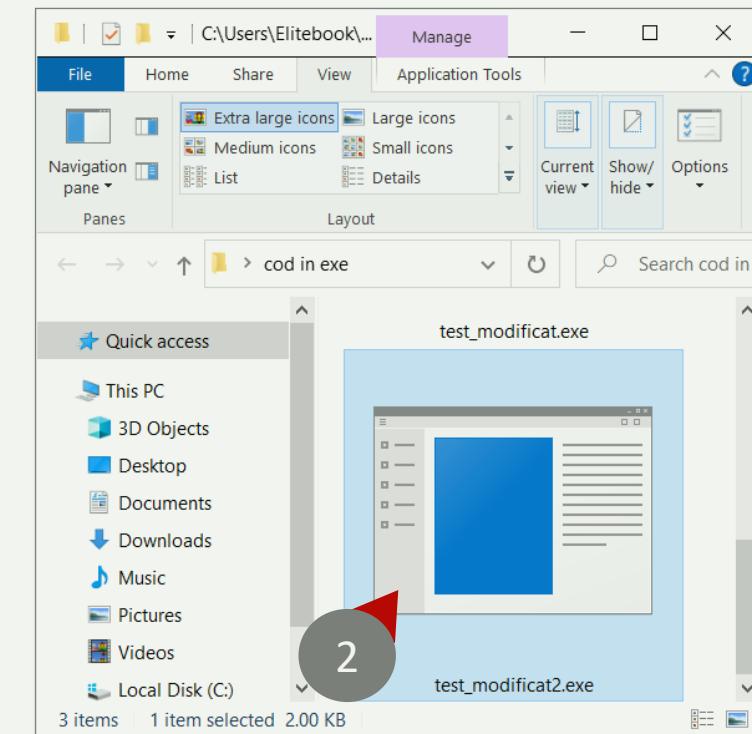
The assembly tab displays the same code as the first screenshot, but the memory dump tabs at the bottom now show the state of memory at address 00401040, which contains the string "Inginerie Inversa. Malware.".

The command bar at the bottom indicates: "Commands are comma separated (like assembly instructions): mov eax, ebx" and "Paused".

Modificările sunt prezentate orientativ:



Noua versiune poate fi testată:

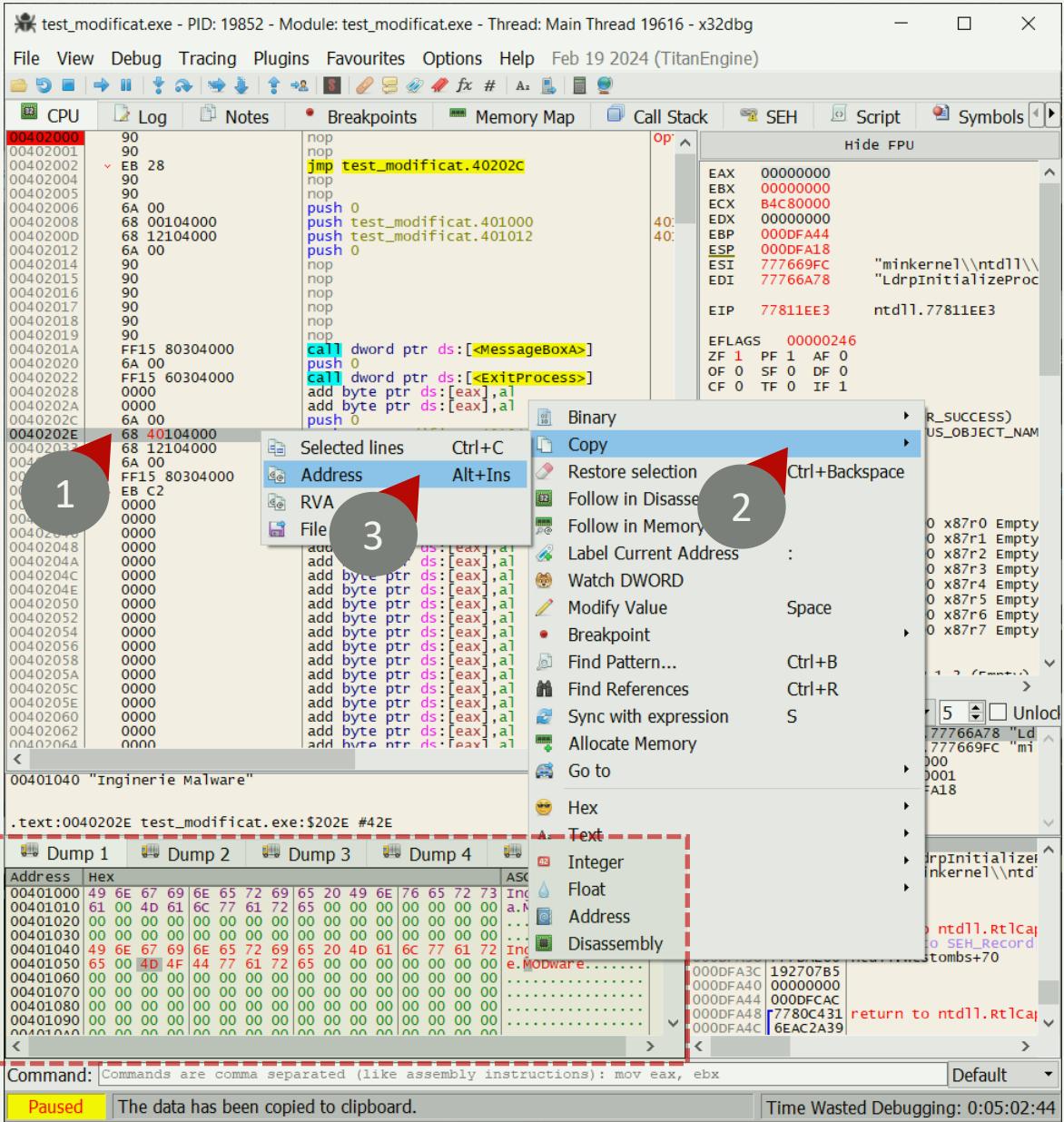


C.8.5

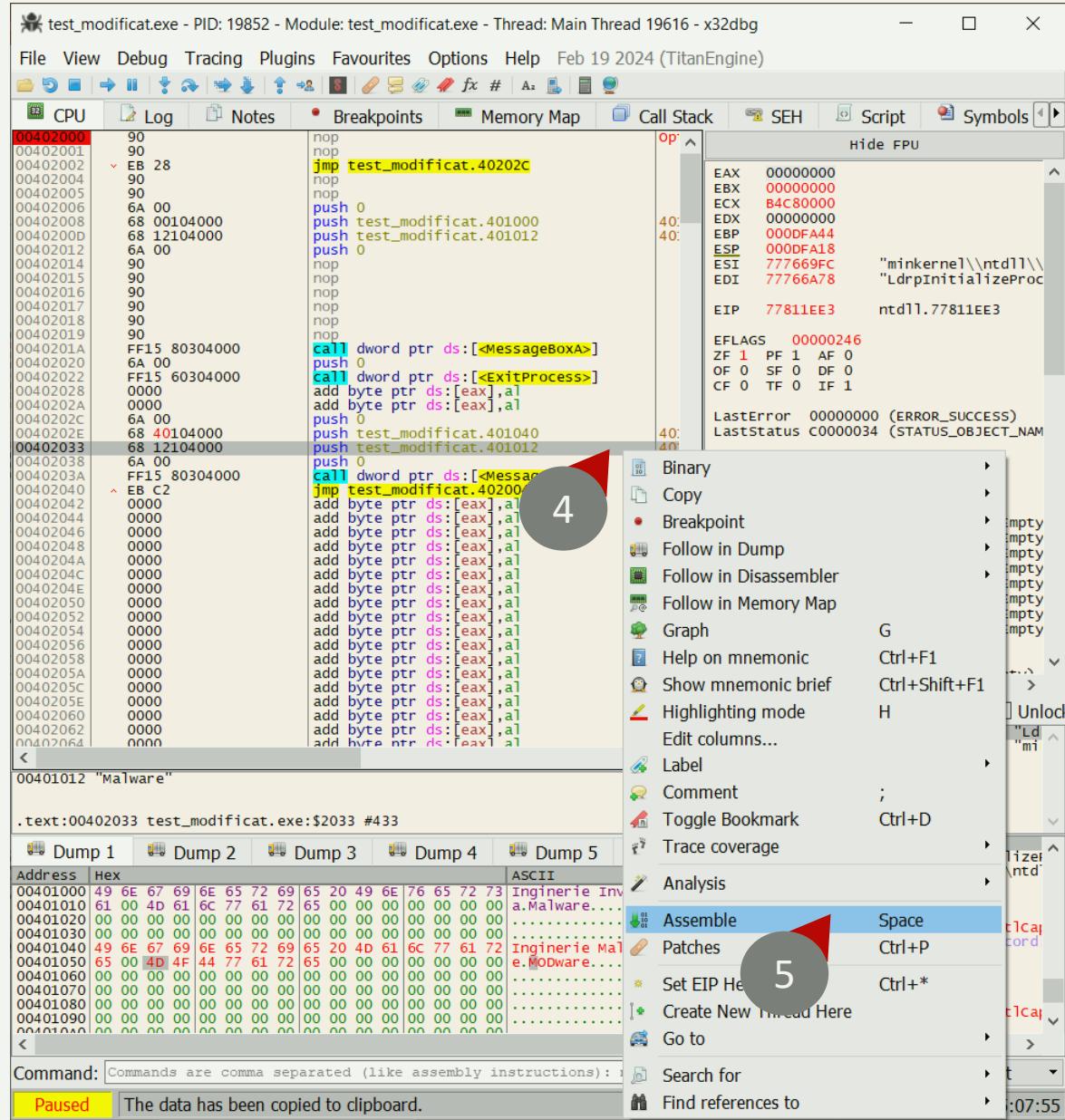
# MODIFICĂRI DE ADRESE CĂTRE ALTE ARGUMENTE



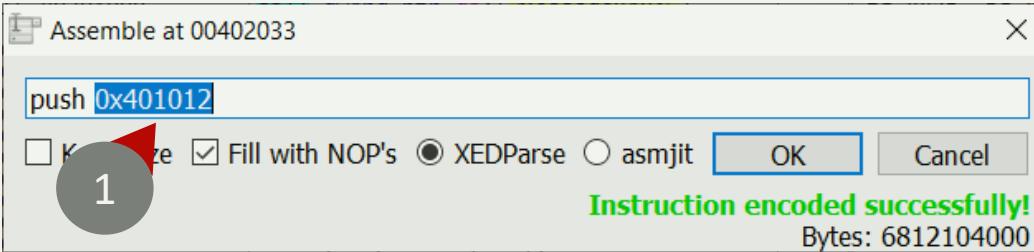
Copiați adresa pentru direcționarea către cel de-al treilea argument:



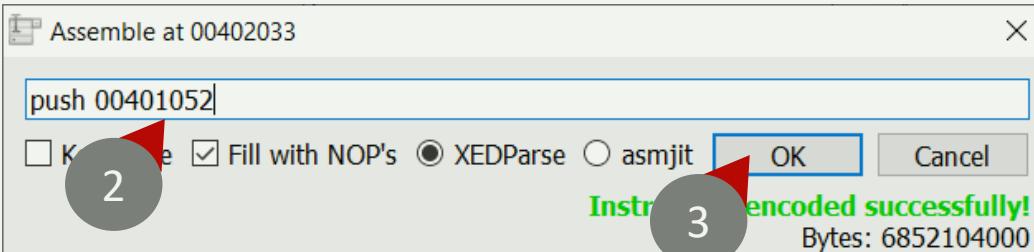
## Schimbarea adresei etichetei:



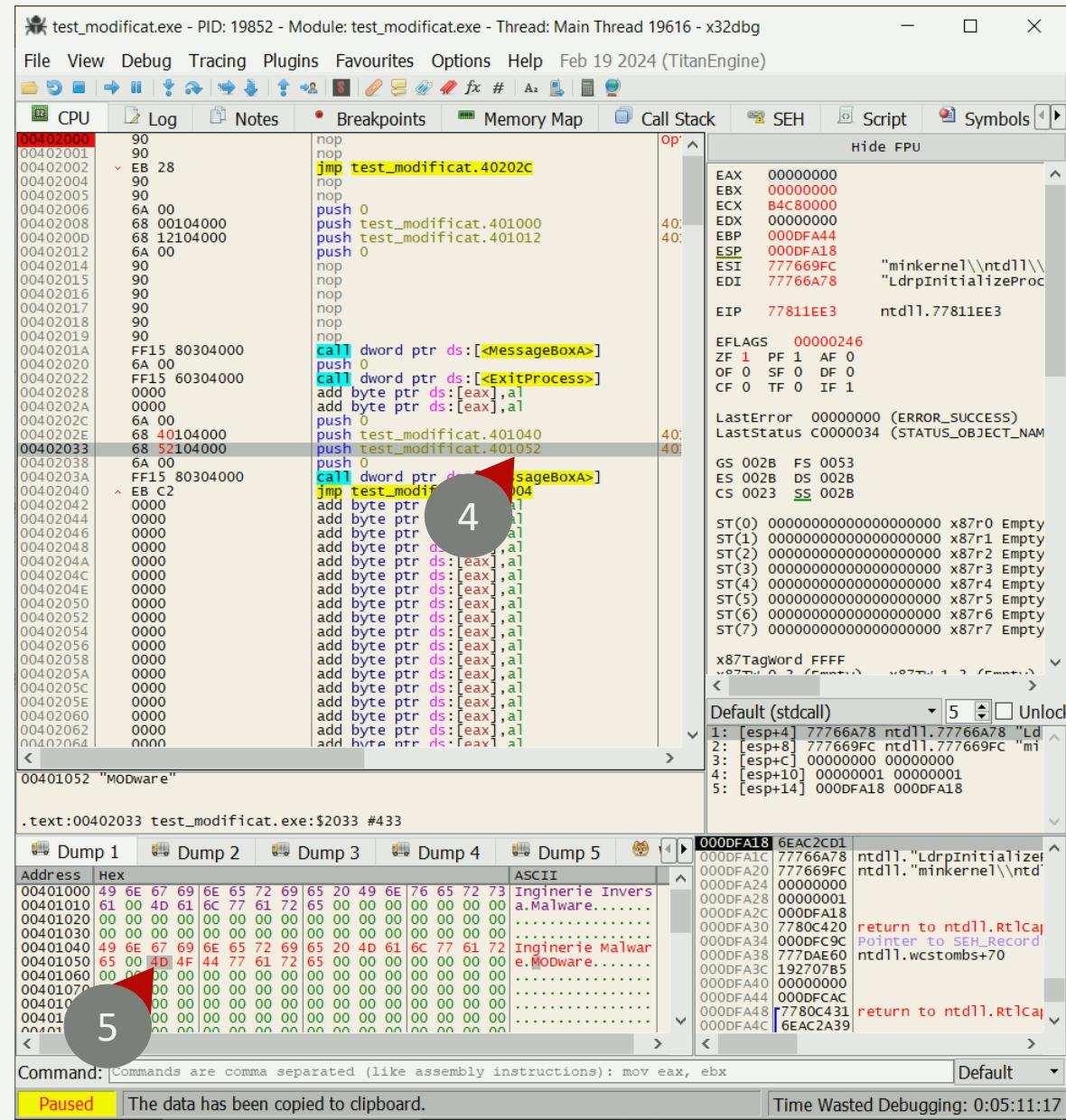
Schimbarea adresei originale:



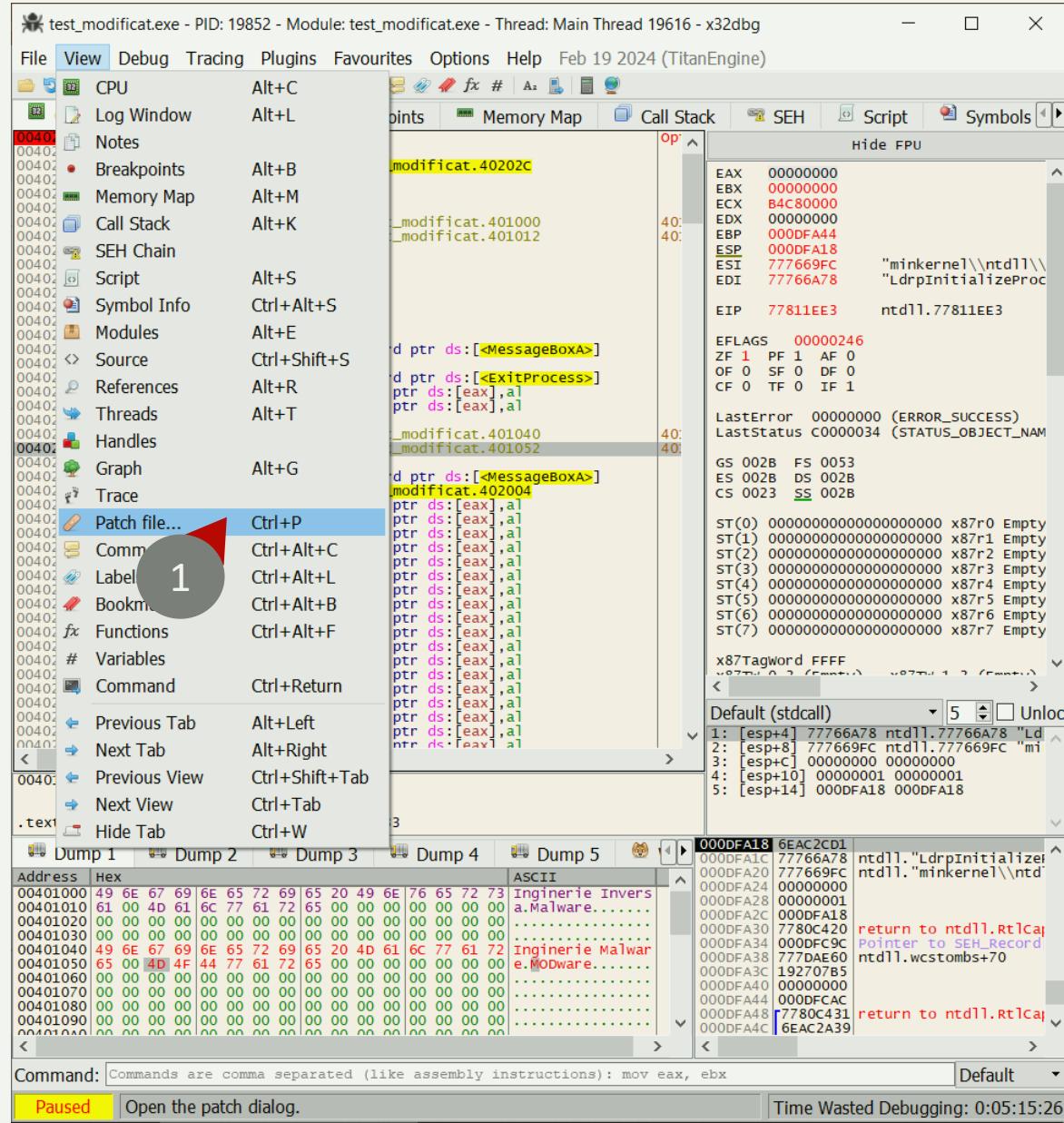
Cu adresa nou din clona:



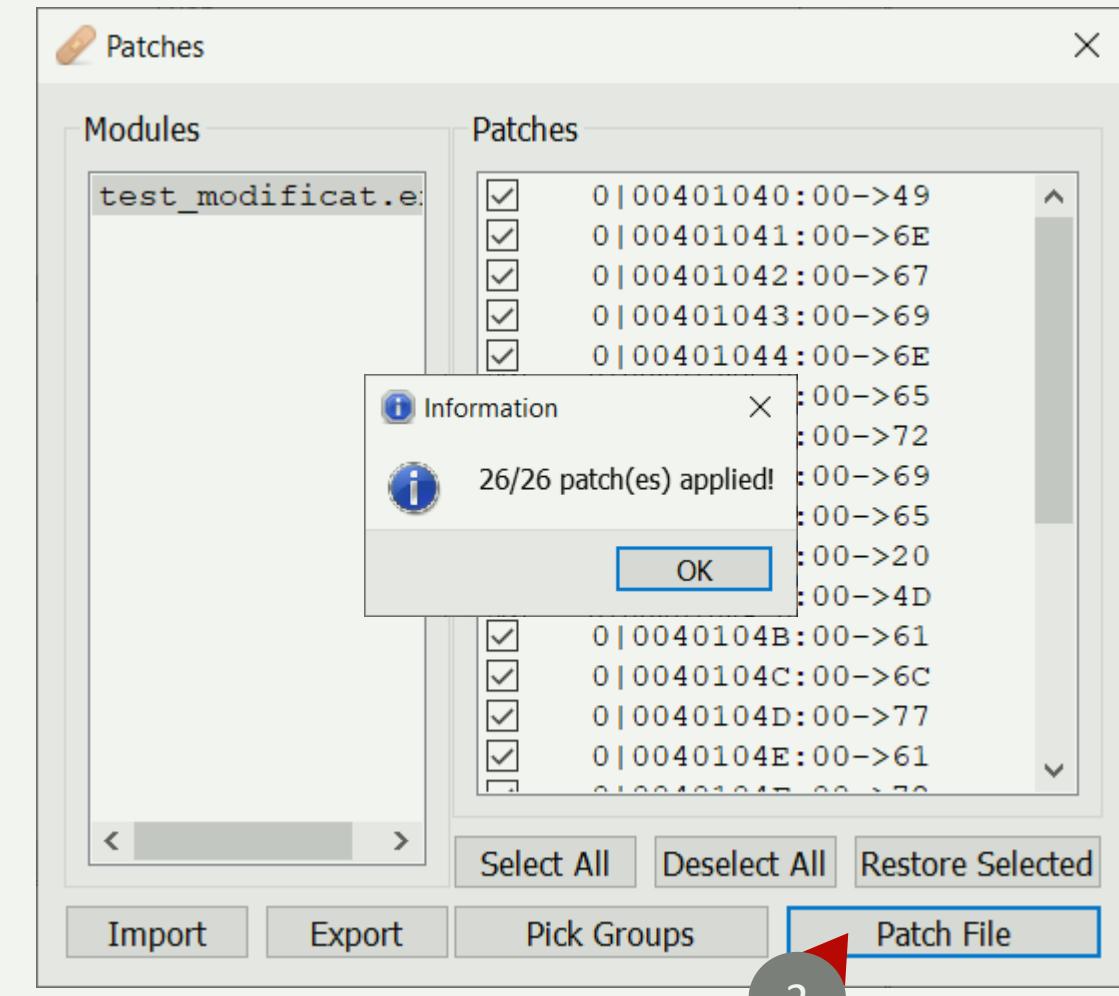
**Adresa a fost schimbată:**



Se trece la peticirea executabilului:



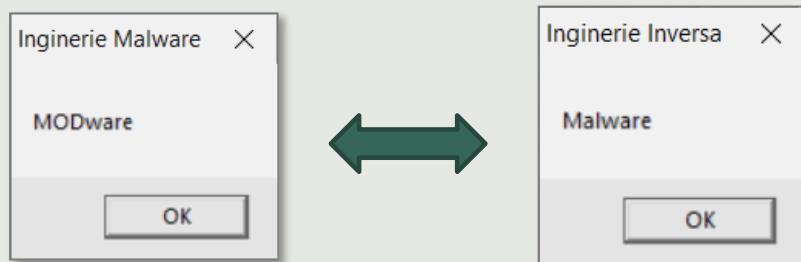
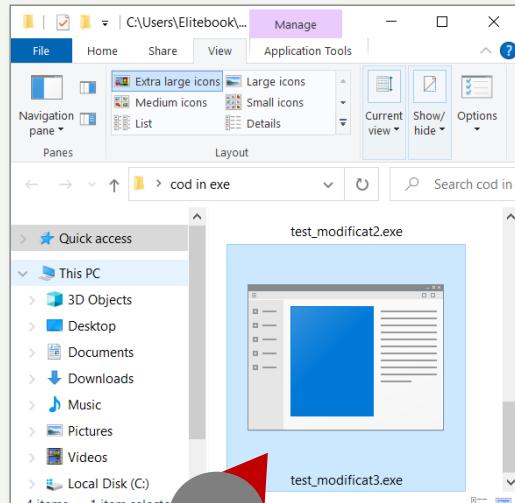
Noua versiune este scrisă pe disc și poate fi testată:



# ÎN CAZUL CODULUI MALIȚIOS

## CE PUTEM REALIZA CU ACESTE TIPURI DE SCHIMBĂRI?

Câteva activități de ghidare:



1. Putem vedea legături între serverele de comandă și control (C&C).
2. Putem dilua infecția mașinilor din rețelele mari creând variante ale aceluiași malware (ex. poate există o luptă în procesul de infecție; unele aplicații malware răspund la C&C original, iar altele la un honeypot).
3. De cele mai multe ori, există un marker pentru infecție (o serie de octeți). Odată ce o mașină este infectată cu malware-ul corectat, malware-ul original nu va infecta acea mașină (este similar cu un **vaccin biologic**).
4. Abaterea de la secțiunea **.text** poate duce la confuzia unui virus dacă se așteaptă la denumirea clasica **.text**

# BIBLIOGRAFIE / RESURSE

- Paul A. Gagniuc. *Antivirus Engines: From Methods to Innovations, Design, and Applications*. Cambridge, MA: Elsevier Syngress, 2024. pp. 1-656.
- Paul A. Gagniuc. *An Introduction to Programming Languages: Simultaneous Learning in Multiple Coding Environments*. Synthesis Lectures on Computer Science. Springer International Publishing, 2023, pp. 1-280.
- Paul A. Gagniuc. *Coding Examples from Simple to Complex - Applications in MATLAB*, Springer, 2024, pp. 1-255.
- Paul A. Gagniuc. *Coding Examples from Simple to Complex - Applications in Python*, Springer, 2024, pp. 1-245.
- Paul A. Gagniuc. *Coding Examples from Simple to Complex - Applications in Javascript*, Springer, 2024, pp. 1-240.
- Paul A. Gagniuc. *Markov chains: from theory to implementation and experimentation*. Hoboken, NJ, John Wiley & Sons, USA, 2017, ISBN: 978-1-119-38755-8.

<https://github.com/gagniuc>

