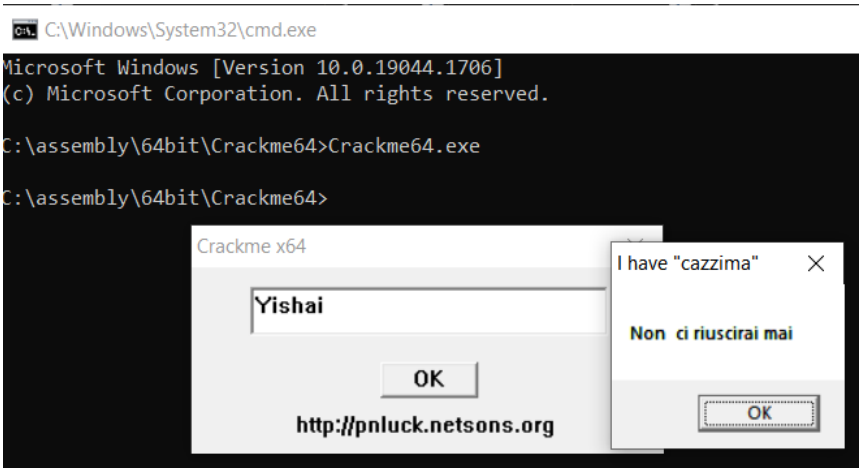
בס"ד

Crackme64_SolutionWay – ישי לוטווק

דבר ראשון נריץ בcmd ונראה מה קורה:



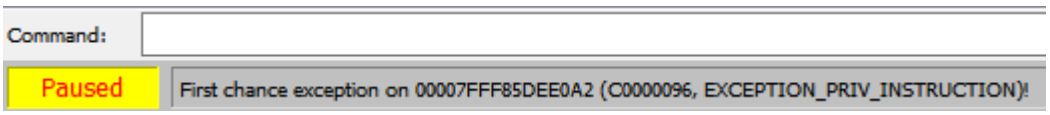
נעלה את הקובץ לx64dbg נריץ אותו לentryPoint, ונחפש את המחרוזת "Non ci riuscirai mai":

Address	Disassembly	String
00007FF683E41A1F	lea r8,qword ptr ds:[7FF683E48280]	L"I have \"cazzima\" "
00007FF683E41A3A	lea r8,qword ptr ds:[7FF683E482D8]	L"Good Job Bros"
00007FF683E42505	lea r8,qword ptr ds:[7FF683E488F0]	"\n\n"
00007FF683E424D2	lea r8,qword ptr ds:[7FF683E488F4]	"..."
00007FF683E42479	lea r8,qword ptr ds:[7FF683E488F8]	"<program name unknown>"
00007FF683E4242E	lea r8,qword ptr ds:[7FF683E48910]	"Runtime Error!\n\nProgram: "
00007FF683E44C15	lea rax,qword ptr ds:[7FF683E4BD00]	&"Sun"
00007FF683E41FFC	lea rcx,qword ptr ds:[7FF683E48328]	L"mscoree.dll"
00007FF683E44086	lea rcx,qword ptr ds:[7FF683E48A48]	"USER32.DLL"
00007FF683E41A26	lea rdx,qword ptr ds:[7FF683E482A8]	L"Non ci riuscirai mai"
00007FF683E41A41	lea rdx,qword ptr ds:[7FF683E482F8]	L"Azz O_O"
00007FF683E4200E	lea rdx,qword ptr ds:[7FF683E48318]	"CorExitProcess"
00007FF683E4255F	lea rdx,qword ptr ds:[7FF683E488C8]	"Microsoft Visual C++ Runtime Library"
00007FF683E4412C	lea rdx,qword ptr ds:[7FF683E489D8]	"GetProcessWindowStation"
00007FF683E440FE	lea rdx,qword ptr ds:[7FF683E489F0]	"GetUserObjectInformationA"
00007FF683E440DF	lea rdx,qword ptr ds:[7FF683E48A10]	"GetLastActivePopup"
00007FF683E440C0	lea rdx,qword ptr ds:[7FF683E48A28]	"GetActiveWindow"
00007FF683E4409F	lea rdx,qword ptr ds:[7FF683E48A38]	"MessageBoxA"

נלחץ על השורה המודגשת, ונסתכל על הקוד שמתקבל:

ניתן לראות שצד שמאל הוא האזור הטוב שבו מודפסת הודעת ההצלחה.

נריץ את הקוד בדיבאגר:



אוייש. כנראה יש מנגנון אנטידיבאג שגורם לקוד לקרוס לפני הקטע המעניין:

נלחץ על * ונסתכל על השורה שבה הקוד קורס:

```
kernel32.00007FFF85DEE0A2
in al,dx
sub byte ptr ds:[rax-1],cl
adc eax,64925
nop dword ptr ds:[rax+rax],eax
int3
int3
int3
```

נלך אחורה דרך כתובות החזרה במחסנית הקריאות:

```
0000000000EFFAC8 00007FF683E41ABC return to crackme64.00007FF683E41ABC from ???
0000000000EFFAD0 00007FF683E48250 crackme64.00007FF683E48250
0000000000EFFAD8 00007FF683E41FBD return to crackme64.00007FF683E41FBD from ???
0000000000EFFAE0 00007FF683E48240 crackme64.00007FF683E48240
0000000000EFFAE8 00007FF683E40000 crackme64.00007FF683E40000
0000000000EFFAF0 00007FF683E48201 crackme64.00007FF683E48201
0000000000EFFAF8 0000000000000000
0000000000EFFB00 00007FF683E48230 crackme64.00007FF683E48230
0000000000EFFB08 00007FF683E420B9 return to crackme64.00007FF683E420B9 from ???
0000000000EFFB10 0000000000000000
0000000000EFFB18 00007FF683E42D82 return to crackme64.00007FF683E42D82 from crackme64.00007FF683E42B24
0000000000EFFB20 0000000002E816B0
0000000000EFFB28 00007FF683E42AD9 return to crackme64.00007FF683E42AD9 from crackme64.00007FF683E445A4
0000000000EFFB30 0000000000000001
0000000000EFFB38 00007FF683E42116 return to crackme64.00007FF683E42116 from crackme64.00007FF683E42094
0000000000EFFB40 0000000000000001
0000000000EFFB48 0000000002E83038 &L"ZES_ENABLE_SYSMAN=1"
0000000000EFFB50 0000000000000001
0000000000EFFB58 00007FF683E40000 crackme64.00007FF683E40000
0000000000EFFB60 0000000000000001
0000000000EFFB68 00007FF683E41D87 return to crackme64.00007FF683E41D87 from crackme64.00007FF683E420D0
0000000000EFFB70 0000000000000000
0000000000EFFB78 0000000000000000
0000000000EFFB80 00007FF683E40000 crackme64.00007FF683E40000
0000000000EFFB88 0000000000000001
```

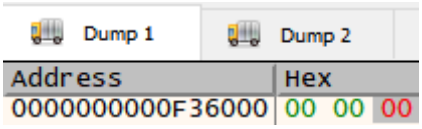
```
crackme64.00007FF683E41A70
 sub rsp,38
 mov rax,qword ptr ds:[<&FatalExit>]
 add rax,2
 mov qword ptr ds:[7FF683E4C040],rax
 mov rax,qword ptr gs:[30]
 mov rax,qword ptr ds:[rax+60]
 movzx eax,byte ptr ds:[rax+2]
 mov byte ptr ss:[rsp+20],al
 movzx eax,byte ptr ss:[rsp+20]
 test eax,eax
 jne crackme64.7FF683E41AB4

crackme64.00007FF683E41AB4
 xor ecx,ecx
 call qword ptr ds:[7FF683E4C040]

crackme64.00007FF683E41AA4
 lea rax,qword ptr ds:[7FF683E41AD0]
 mov qword ptr ds:[7FF683E4B000],rax
 jmp crackme64.7FF683E41ABC

crackme64.00007FF683E41ABC
 xor eax,eax
 add rsp,38
 ret
```

השורה של הxor (זאת שמודגשת) היא כתובת החזרה. כלומר הפקודה call בסוף הבלוק משמאל קראה לפונקציה שבתוכה היינו כרגע ובה אירעה הקריסה.

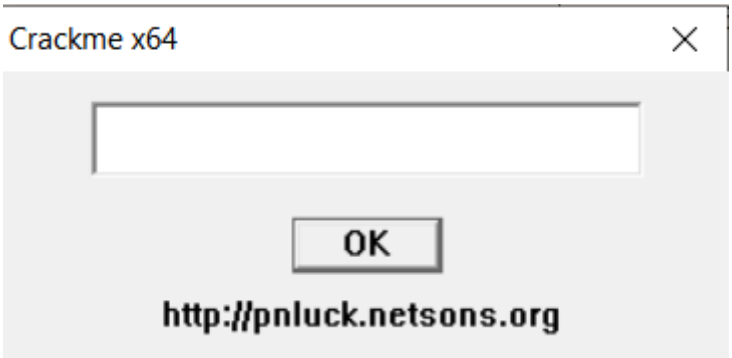
(נשים bp על השורה לפני ההתפצלות כדי שלא נשכח את הנקודה הזו אך נסמן את הbp כdisable כדי שלא יפריע לנו – הריבוע הירוק בתמונה למעלה)

ייתכן שמגנון האנטי דיבאג גורם לקוד להגיע לענף השמאלי ולקרוס במקום ללכת דרך הענף הימני שבו בכלל אין קריאה לפונקציה.

בוא ננסה רגע להעלות את הקובץ מחדש, ללכת לpeb+2 ולאפס אותו, ואז להמשיך את הריצה. אולי זה יעזור ואז נבין שכנראה יש איזו פונקציה שבודקת אם אנחנו מדבגים את הקוד באמצעות הערך שנמצא בבית השלישי של peb..



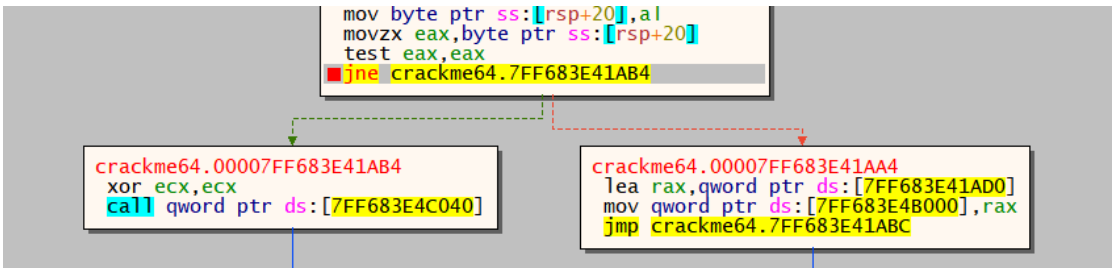
ונריץ:



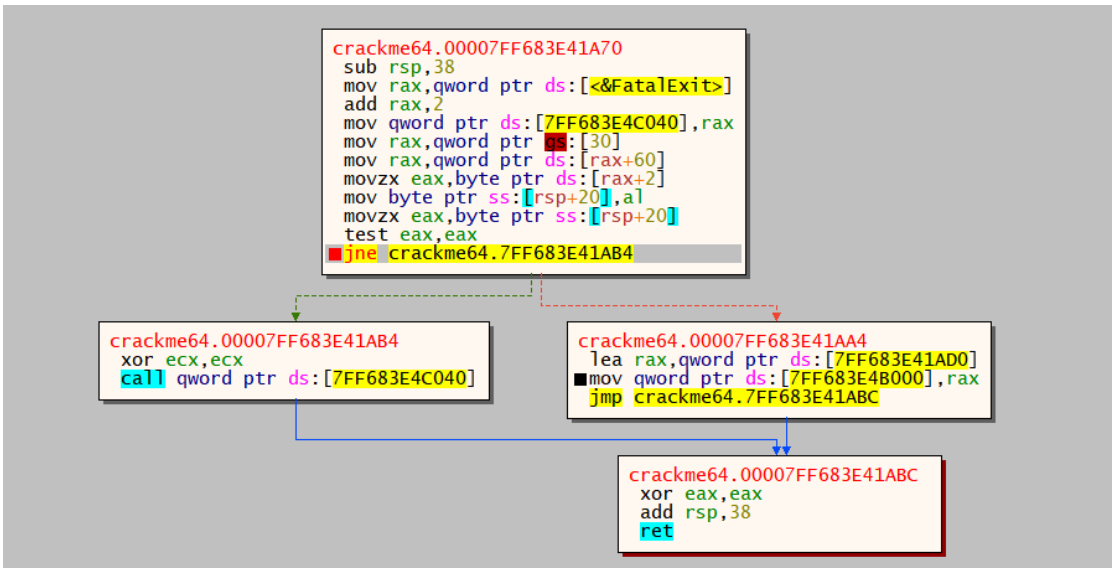
יש!! החלון נפתח!

נרצה לבדוק אם בהתפצלות בה פנינו קודם שמאלה וקרסנו הקוד עכשיו התנהג באופן שונה.

נחזור לנקודה הזו ונשים bp ואחר כך נחזור על מה שעשינו כעת (עם peb וכו') שוב.

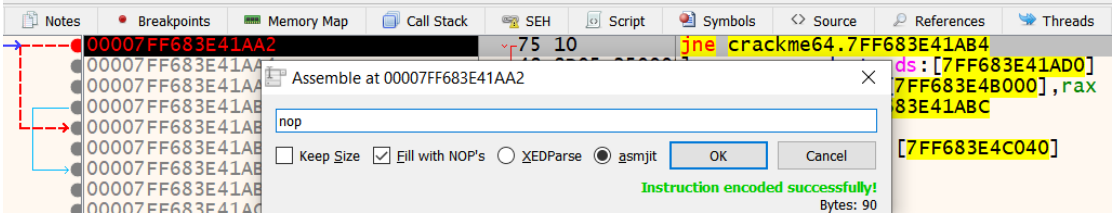


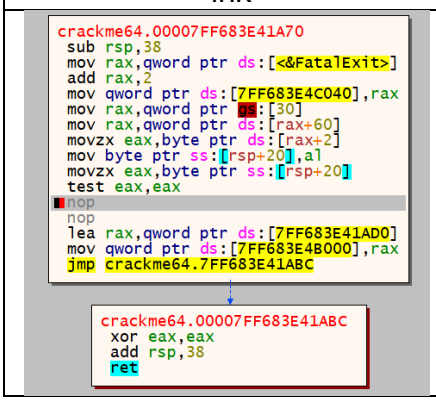
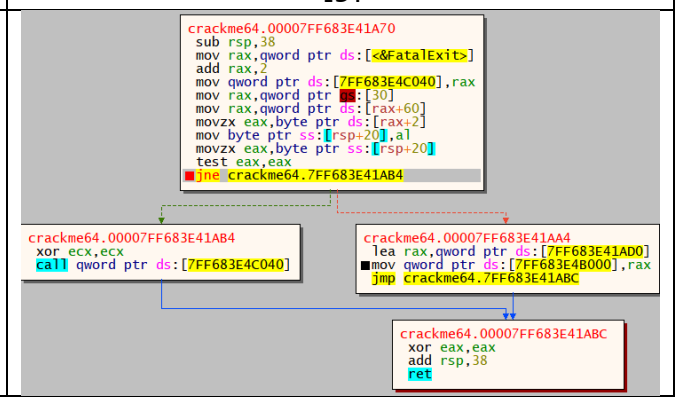
הקוד עצר בנקודה המיוחלת. עכשיו נצעד צעד צעד ונראה מה קורה:



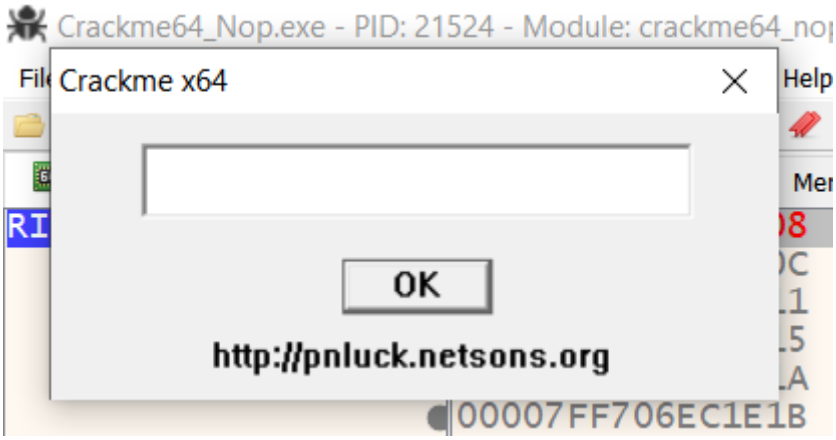
ניתן לראות את הריבוע השחור בענף הימני. כלומר זה ששינינו את הערך בבית השלישי בpeb ל0 גרם לקוד ללכת לענף הימני ולא כמו מקודם – לענף השמאלי (שגורם לקריסה).

נבצע פיצפוץ קטן כדי שלא נצטרך בכל הרצה לשנות את הבית השלישי בpeb:



אחרי	לפני
	

נשמור את הקובץ החדש המפוצפץ בשם Crackme64_Nop.exe ונריץ אותו בדיבאגר לראות אם עכשיו זה יעבוד כמו שצריך:



עובד!

יכול להיות שסיימנו לנטרל את מנגנון האנטי דיבאג. ננסה להמשיך ואם נראה שיש עוד מנגנונים נעצור ונטפל בהם.

כרגע זה נראה שהייתה איפה שהוא פונקציית isDebuggerPresent שגרמה לקוד ללכת בענף השמאלי שהראינו בתמונות לעיל (אותו ענף שגורם לקוד לקרוס).

עכשיו נשים bp בנקודה של ההתפצלות בין האזור הטוב לאזור הרע,

נכניס סיסמא ונריץ את הקוד:

נשים לב שהתנאי בודק אם מה שיש בתוך הרגיסטר rax שווה ל0,

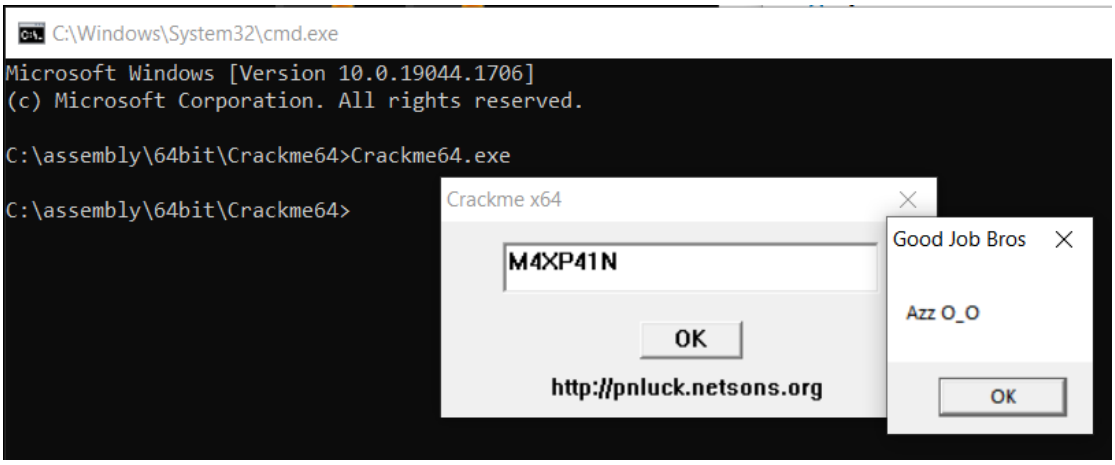
אם הוא 0 הולכים לצד ימין – הצד הרע, אחרת הולכים לצד הטוב.

ייתכן מאוד שהרגיסטר rax מכיל את הערך המוחזר מהפונקציה האחרונה שנקראת לפני ההתפצלות.
ניכנס אליה רגע ונראה מה היא מכילה:

```
crackme64_nop.00007FF706EC1AD0
 mov qword ptr ss:[rsp+8],rcx ; [rsp+8]:L"OK"
 sub rsp,18
 mov dword ptr ss:[rsp],0 ; [rsp]:L"Yishai"
 mov rax,qword ptr ss:[rsp+20]
 movzx eax,word ptr ds:[rax]
 cmp eax,4D ; 4D:'M'
 jne crackme64_nop.7FF706EC1B55

crackme64_nop.00007FF706EC1AED
 mov rax,qword ptr ss:[rsp+20]
 movzx eax,word ptr ds:[rax+2]
 cmp eax,34 ; 34:'4'
 jne crackme64_nop.7FF706EC1B55

crackme64_nop.00007FF706EC1AFB
 mov rax,qword ptr ss:[rsp+20]
 movzx eax,word ptr ds:[rax+4]
 cmp eax,58 ; 58:'X'
 jne crackme64_nop.7FF706EC1B55

crackme64_nop.00007FF706EC1B09
 mov rax,qword ptr ss:[rsp+20]
 movzx eax,word ptr ds:[rax+6]
 cmp eax,50 ; 50:'P'
 jne crackme64_nop.7FF706EC1B55

crackme64_nop.00007FF706EC1B17
 mov rax,qword ptr ss:[rsp+20]
 movzx eax,word ptr ds:[rax+8]
 cmp eax,34 ; 34:'4'
 jne crackme64_nop.7FF706EC1B55

crackme64_nop.00007FF706EC1B25
 mov rax,qword ptr ss:[rsp+20]
 movzx eax,word ptr ds:[rax+A]
 cmp eax,31 ; 31:'1'
 jne crackme64_nop.7FF706EC1B55

crackme64_nop.00007FF706EC1B33
 mov rax,qword ptr ss:[rsp+20]
 movzx eax,word ptr ds:[rax+C]
 cmp eax,4E ; 4E:'N'
 jne crackme64_nop.7FF706EC1B55

crackme64_nop.00007FF706EC1B41
 mov rax,qword ptr ss:[rsp+20]
 movzx eax,word ptr ds:[rax+E]
 test eax,eax
 jne crackme64_nop.7FF706EC1B55
```

זה נראה שבודקים פה עם הסיסמא היא: **M4XP41N**.

ננסה את מזלנו - נפתח את הcmd ונריץ את הקובץ עם הסיסמה הזו:

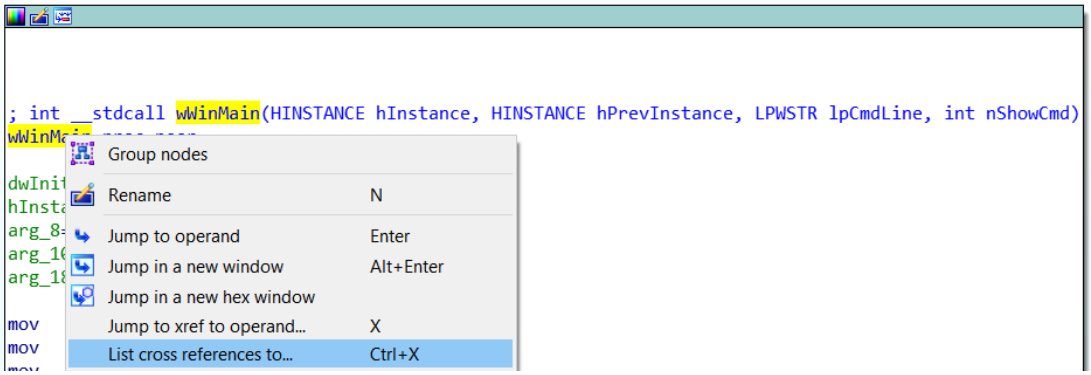


ב"ה!

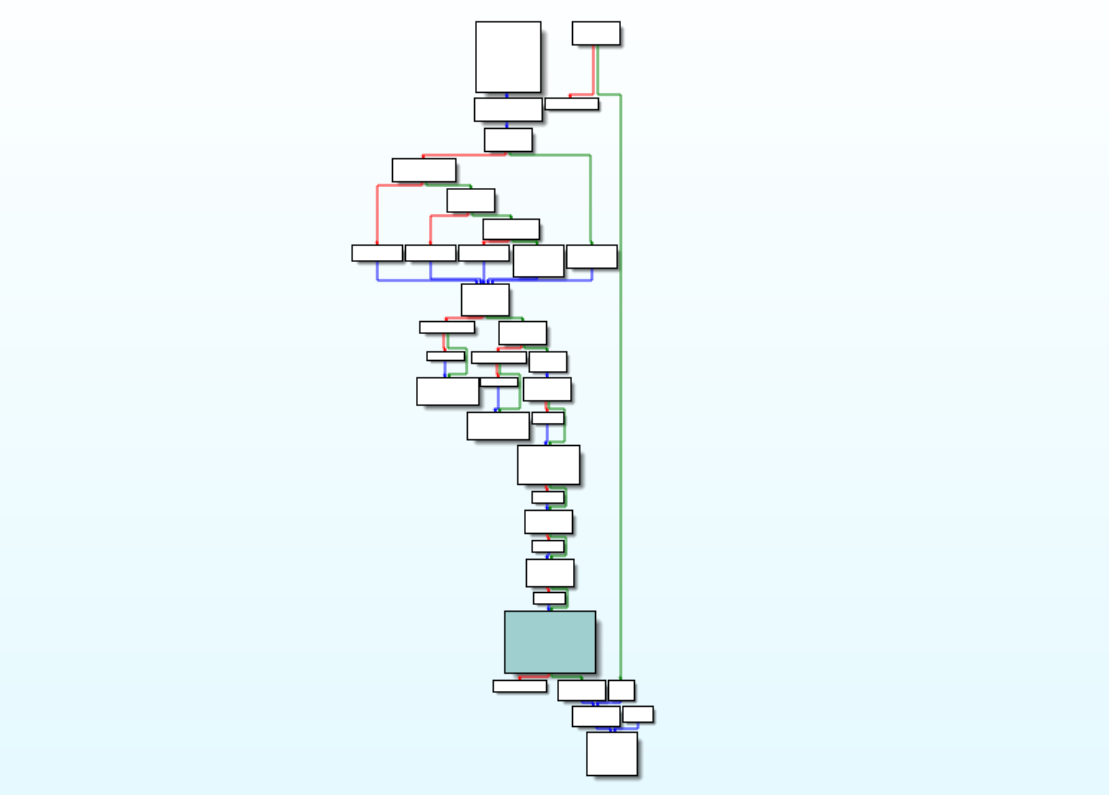
עכשיו נבדוק איפה המיין. בטח בida זה יהיה יותר פשוט. נפתח את הקובץ בida:

```
; int __stdcall wWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPWSTR lpCmdLine, int nShowCmd)
wWinMain proc near

dwInitParam= qword ptr -18h
hInstance= qword ptr  8
arg_8= qword ptr  10h
arg_10= qword ptr  18h
arg_18= dword ptr  20h

mov     [rsp+arg_18], r9d
mov     [rsp+arg_10], r8
mov     [rsp+arg_8], rdx
mov     [rsp+hInstance], rcx
sub     rsp, 38h
mov     [rsp+38h+dwInitParam], 0 ; dwInitParam
lea     r9, DialogFunc  ; lpDialogFunc
xor     r8d, r8d        ; hWndParent
mov     edx, 67h        ; lpTemplateName
mov     rcx, [rsp+38h+hInstance] ; hInstance
call    cs:DialogBoxParamW
add     rsp, 38h
retn
wWinMain endp
```

נלך אחורה באמצעות החלון הבא:



נגיע למבנה הבא כאשר אנו נמצאים במלבן המסומן בכחול:



המלבן הכחול מקרוב, ניתן לראות את הקריאה לפונקציה main מסומנת בצהוב:

```
loc_140001D92:
call    _wwincmdln
test    byte ptr [rsp+0A8h+StartupInfo.dwFlags], dil
movzx   edx, [rsp+0A8h+StartupInfo.wShowWindow]
mov     r9d, 0Ah
cmovnz  r9d, edx          ; nShowCmd
mov     r8, rax           ; lpCmdLine
xor     edx, edx          ; hPrevInstance
mov     rcx, rsi          ; hInstance
call    wWinMain
mov     edi, eax
mov     [rsp+0A8h+var_88], eax
test    ebx, ebx
jnz     short loc_140001DC9
```

נחזור לחלון שנפתח לנו כשפתחנו את ida:

```
; int __stdcall wWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPWSTR lpCmdLine, int nShowCmd)
wWinMain proc near

dwInitParam= qword ptr -18h
hInstance= qword ptr  8
arg_8= qword ptr  10h
arg_10= qword ptr  18h
arg_18= dword ptr  20h

mov     [rsp+arg_18], r9d
mov     [rsp+arg_10], r8
mov     [rsp+arg_8], rdx
mov     [rsp+hInstance], rcx
sub     rsp, 38h
mov     [rsp+38h+dwInitParam], 0 ; dwInitParam
lea     r9, DialogFunc  ; lpDialogFunc
xor     r8d, r8d        ; hWndParent
mov     edx, 67h        ; lpTemplateName
mov     rcx, [rsp+38h+hInstance] ; hInstance
call    cs:DialogBoxParamW
add     rsp, 38h
retn
wWinMain endp
```

נחפש באינטרנט מה מקבלת הפונקציה DialogBoxParamW:

C++ Copy

```
INT_PTR DialogBoxParamA(
  [in, optional] HINSTANCE hInstance,
  [in]           LPCSTR    lpTemplateName,
  [in, optional] HWND      hWndParent,
  [in, optional] DLGPROC   lpDialogFunc,
  [in]           LPARAM    dwInitParam
);
```

Type: **HWND**

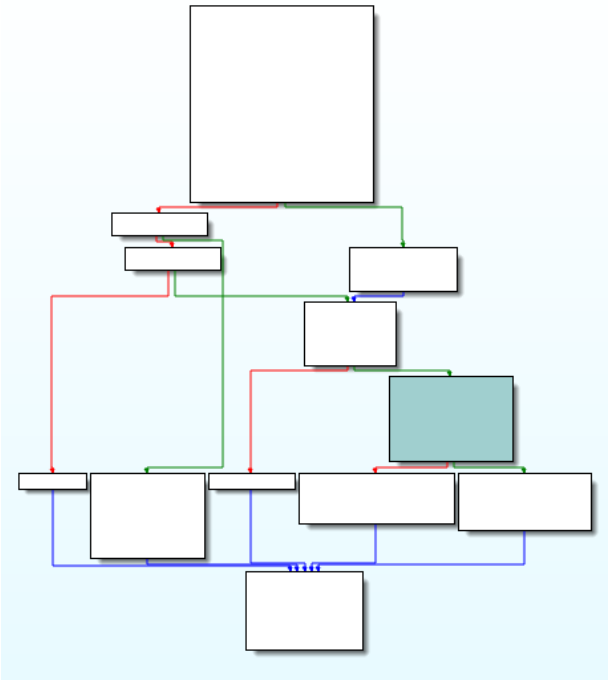
A handle to the window that owns the dialog box.

`[in, optional]` `lpDialogFunc`

נראה שאנו שולחים לה בפרמטר lpDialogFunc את הפונקציה שמנהלת את העניינים..

ניכנס אליה וננסה למצוא את הקוד שראינו בx64dbg:

המבנה של הפונקציה הזו הוא המבנה הבא, כאשר המלבן הכחול מסתיים בתנאי שמפצל בין האזור הרע לאזור הטוב **שכבר ראינו בx64dbg:**



והנה המלבן הכחול מקרוב עם ההתפצלות לאזור הרע ולאזור הטוב:

```
loc_1400019EE:                  ; cchMax
mov     r9d, 14h
lea     r8, [rsp+78h+String] ; lpString
mov     edx, 3E8h               ; nIDDlgItem
mov     rcx, [rsp+78h+hDlg] ; hDlg
call    cs:GetDlgItemTextW
lea     rcx, [rsp+78h+String]
call    cs:off_14000B000
test    rax, rax
jnz     short loc_140001A37
```

```
50  xor     r9d, r9d            ; uType
    lea     r8, Caption         ; "I have \"cazzima\" "
    lea     rdx, Text           ; "Non  ci riuscirai mai"
    xor     ecx, ecx            ; hWnd
    call    cs:MessageBoxW
    jmp     short loc_140001A50
```

```
loc_140001A37:                  ; uType
xor     r9d, r9d
lea     r8, aGoodJobBros ; "Good Job Bros"
lea     rdx, aAzzOO      ; "Azz O_O"
xor     ecx, ecx         ; hWnd
call    cs:MessageBoxW
```

לסיום רק נציין שכנראה שיש קוד שרץ לפני המיין – אותו קוד שאותו פצפצנו כדי לנטרל את מנגנון האנטי דיבאג.

מ.ש.ל.