מבחן מועד ב ברברסינג:

1. תחילה אנו צריכים למצוא את המain בתוכנית. ניכנס לstart_0:

```
; Attributes: thunk

public start
start proc near
jmp     start_0
start endp
```

עכשיו לsub_412260:

```
; Attributes: bp-based frame

start_0 proc near
push    ebp
mov     ebp, esp
call    sub_412260
pop     ebp
retn
start_0 endp
```

עכשיו לsub_412280 (הפונקציה האחרונה כמובן):

```
; Attributes: bp-based frame

sub_412260 proc near
push    ebp
mov     ebp, esp
call    sub_4112DF
call    sub_412280
pop     ebp
retn
sub_412260 endp
```

נרד עד למטה לפני הexit_code וניכנס לפונקציה sub_412550 (הזו שאחרי נוסתה והיא לא המain):

```
; Attributes: bp-based frame

sub_412550 proc near
push    ebp
mov     ebp, esp
call    j__get_initial_narrow_environment
push    eax
call    j___p___argv
mov     eax, [eax]
push    eax
call    j___p___argc
mov     ecx, [eax]
push    ecx
call    sub_4112C6
add     esp, 0Ch
pop     ebp
retn
sub_412550 endp
```

כאן כמובן אנו רואים את שלושת הפרמטרים המועברים לmain. ניכנס לפונקציה sub_4112c6:

```
; Attributes: bp-based frame

; int __cdecl main(int argc, const char **argv, const char **envp)
main proc near

var_E4= byte ptr -0E4h
var_20= dword ptr -20h
var_14= dword ptr -14h
lpParameter= dword ptr -8
argc= dword ptr  8
argv= dword ptr  0Ch
envp= dword ptr  10h

push    ebp
mov     ebp, esp
sub     esp, 0E4h
push    ebx
push    esi
push    edi
lea     edi, [ebp+var_E4]
mov     ecx, 39h
mov     eax, 0CCCCCCCCh
rep stosd
mov     ecx, offset unk_41C04A
call    sub_41122B
mov     [ebp+lpParameter], 0
mov     esi, esp
push    0               ; lpThreadId
push    0               ; dwCreationFlags
mov     eax, [ebp+lpParameter]
push    eax             ; lpParameter
```

ניכנס לפונקציה sub_411235:

```
push    ebp
mov     ebp, esp
sub     esp, 0E4h
push    ebx
push    esi
push    edi
lea     edi, [ebp+var_E4]
mov     ecx, 39h
mov     eax, 0CCCCCCCCh
rep stosd
mov     ecx, offset unk_5CC04A
call    sub_5C122B
mov     [ebp+lpParameter], 0
mov     esi, esp
push    0               ; lpThreadId
push    0               ; dwCreationFlags
mov     eax, [ebp+lpParameter]
push    eax             ; lpParameter
push    offset StartAddress ; lpStartAddress
push    0               ; dwStackSize
push    0               ; lpThreadAttributes
call    ds:CreateThread
cmp     esi, esp
call    sub_5C1235
mov     [ebp+var_14], eax
cmp     [ebp+argc], 2
jz      short loc_5C1C9C
```

ניכנס:

```
; Attributes: thunk

sub_5C1235 proc near
jmp     sub_5C2070
sub_5C1235 endp
```

כנ"ל:

```
loc_5C2075:
push    ebp
mov     ebp, esp
sub     esp, 0
push    eax
bndstx  [esp+eax+8+var_8], bnd0
push    edx
push    ebx
push    esi
push    edi
mov     eax, [ebp+4]
push    0
push    eax
call    sub_5C128A
add     esp, 8
pop     edi
pop     esi
pop     ebx
pop     edx
pop     eax
bndldx  bnd0, [esp+eax+4+var_8]
mov     esp, ebp
pop     ebp
bnd retn
sub_5C2070 endp
```

ניכנס לפונקציה sub_5C128A:

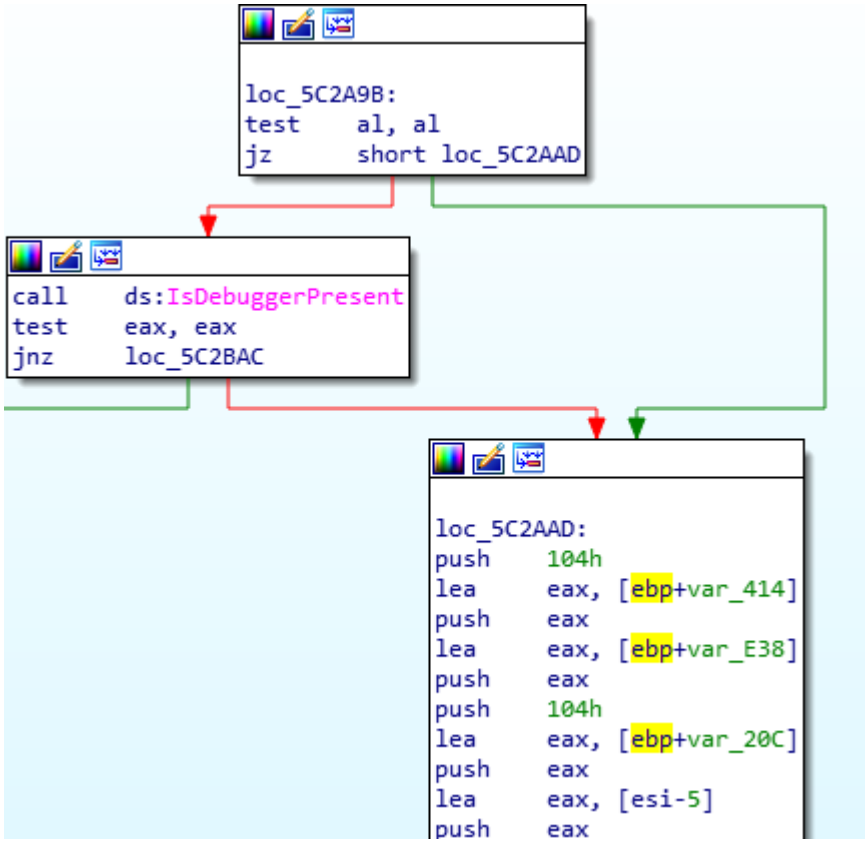
```
; Attributes: thunk

sub_5C128A proc near
jmp     sub_5C27C0
sub_5C128A endp
```

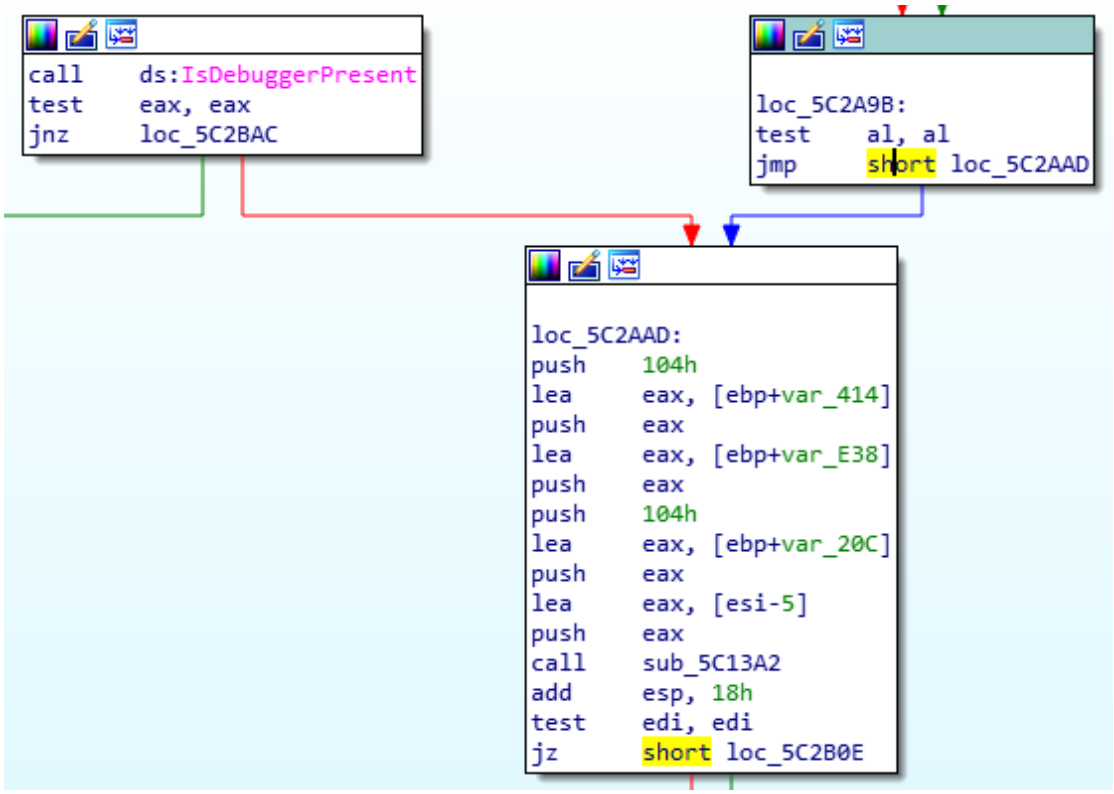
כנ"ל:

```
push    edx             ; lpMultiByteStr
push    eax             ; int
push    ecx             ; int
push    [ebp+arg_0]     ; int
call    sub_5C29C0
add     esp, 10h
pop     ebp
retn
```

כפי המתואר בתמונה האחרונה, נכנסנו לפונקציה sub_5C27C0 ולאחר מכן ירדנו עד למטה לפונקציה האחרונה. כעת ניכנס לפונקציה sub_5C29C0 :



ומצאנו את האנטי-דיבאג. האנטידיבאג הוא IsDebuggerPresent.
כדי לעקוף (כמובן) את מנגנון האנטידיבאג ניגש לloc_5C2A9B (מוצג בתמונה האחרונה)
ושם נשנה את התנאי jz short loc_5C2AAD ל jmp short loc_5C2AAD:



2. בשאלה זו אנו אמורים למצוא את הסיסמה שתביא לנו ציון מושלם, ללא עקיפה דרך patching.

תחילה אנו רואים כי לאחר הכנסת הסיסמה התוכנית בודקת את אורך הסיסמה.
לאחר מכן נלך בעקבות החץ הירוק לכאן:

```
loc_411CCB:
mov     eax, 4
shl     eax, 0
mov     ecx, [ebp+argv]
mov     edx, [ecx+eax]
push    edx             ; Str1
call    sub_4110D7
add     esp, 4
```

ניכנס לפונקציה sub_4110d7 :

```
mov     eax, [ebp+var_3C]
movsx   ecx, [ebp+eax+Str]
xor     ecx, 7
mov     edx, [ebp+var_3C]
mov     [ebp+edx+Str], cl
jmp     short loc 41190E
```

כפי המוצג בתמונה האחרונה כאן, אנו "מנווטים" קצת באלגוריתם של פונקציה זו בכדי לראות מה היא עושה עם המחרוזת. נשים breakpoint ונריץ, וכאשר נגיע לשלב זה נקבל את המחרוזת:
NKhqbUbqbutni'
אקח את מחרוזת זו לpython, אעשה XOR עם 7 כמו שנעשה בפונקציה ונגיע למחרוזת הבאה:

```
Python 3.8.5 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> a = list("NKhqbUbqbutni'")
>>> a
['N', 'K', 'h', 'q', 'b', 'U', 'b', 'q', 'b', 'u', 't', 'n', 'i', "'"]
>>> map(lambda x: x^7, a)
<map object at 0x034D9F40>
>>> print(list(map(lambda x: ord(x)^7,a)))
[73, 76, 111, 118, 101, 82, 101, 118, 101, 114, 115, 105, 110, 32]
>>> b = (list(map(lambda x: ord(x)^7,a)))
>>> print(list(maplambda x: chr(x), b)))
SyntaxError: invalid syntax
>>> print(list(lambda x: chr(x), b)))
SyntaxError: unmatched ')'
>>> print(list(lambda x: chr(x), b))
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    print(list(lambda x: chr(x), b))
TypeError: list expected at most 1 argument, got 2
>>> print(list(map(lambda x: chr(x), b)))
['I', 'L', 'o', 'v', 'e', 'R', 'e', 'v', 'e', 'r', 's', 'i', 'n', ' ']
>>> |
```

לכן המחרוזת שיצאה לנו הינה: ILoveReversing