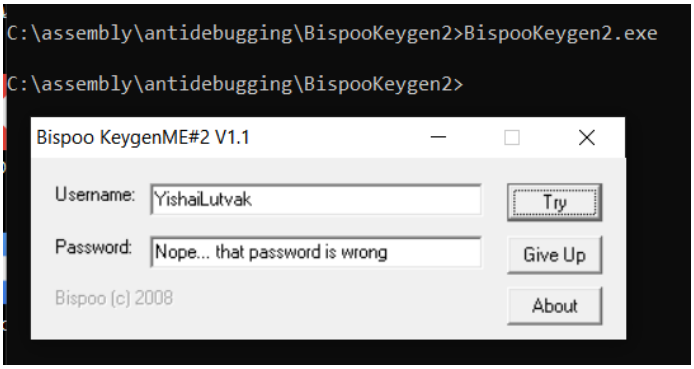
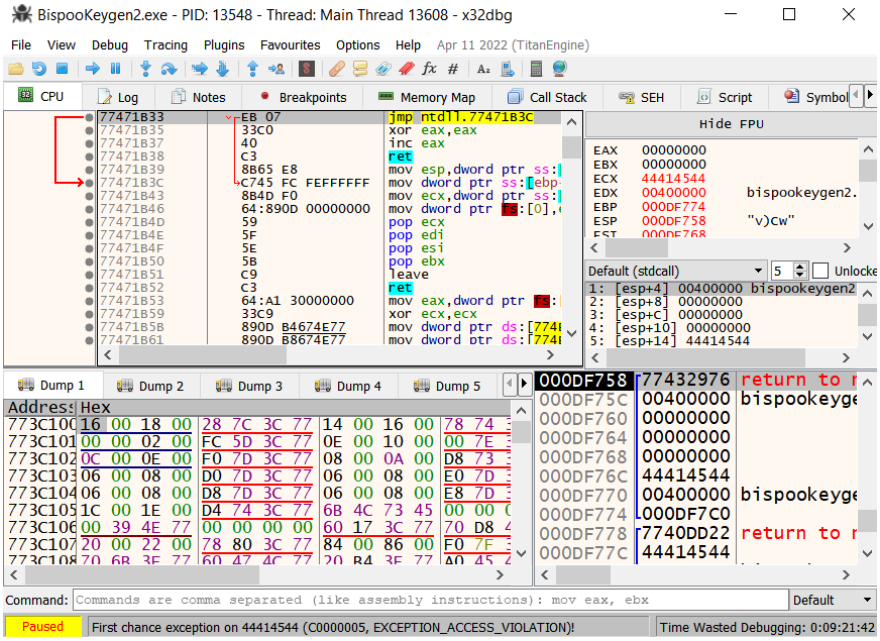
בס"ד

הדרך לפיתרון - BispooKeygen2

נריץ את הקובץ דרך הcmd:

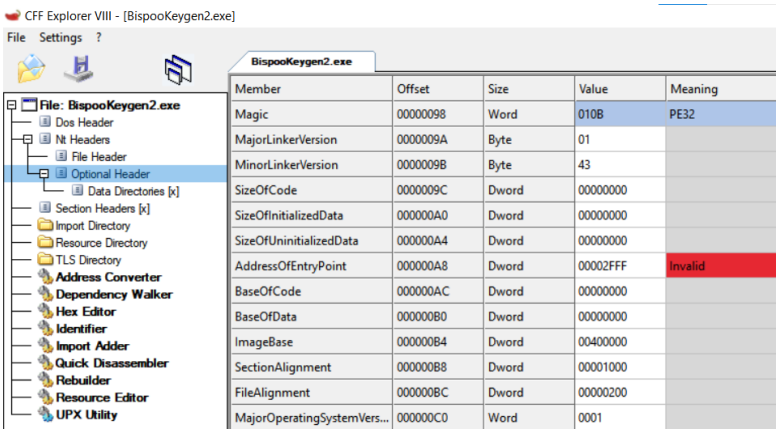


נפתח את הקובץ בexe32dbg וננסה להריץ:



ניתן לראות בשורה האחרונה למטה שאירעה שגיאה.

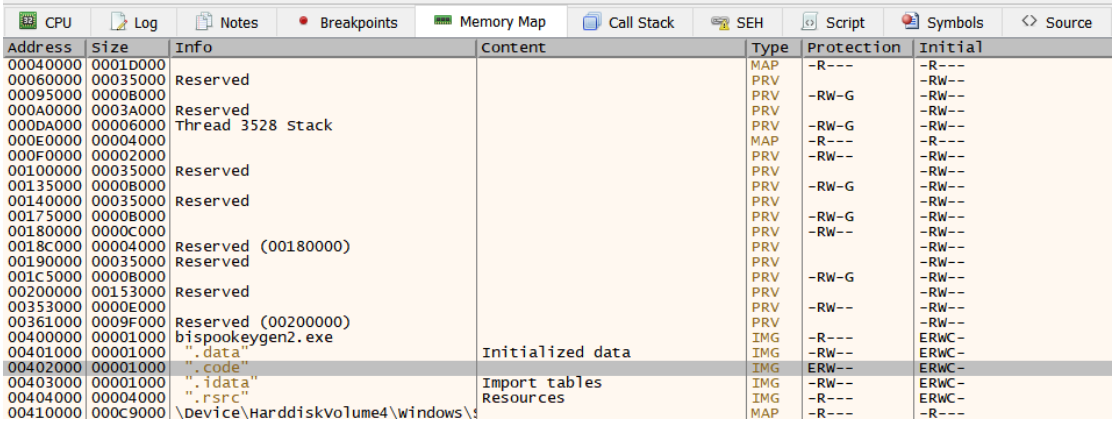
נפתח את הקובץ בCffExplorer:



ניתן לראות שהEntryPoint לא תקין (הערך האמיתי של נקודת ההתחלה היא:

ImageBase + AddressOfEntryPoint כלומר 0x402FFF)

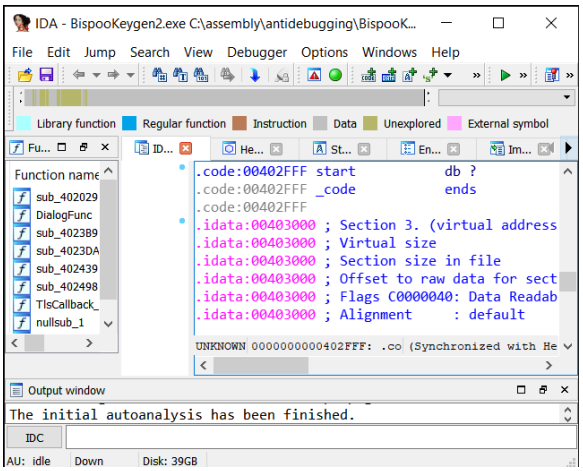
נסתכל בex32dbg בחלון של Memory Map:



Address	Size	Info	Content	Type	Protection	Initial
00040000	0001D000			MAP	-R---	-R---
00060000	00035000	Reserved		PRV		-RW--
00095000	0000B000			PRV	-RW-G	-RW--
000A0000	0003A000	Reserved		PRV		-RW--
000DA000	00006000	Thread 3528 Stack		PRV	-RW-G	-RW--
000E0000	00004000			MAP	-R---	-R---
000F0000	00002000			PRV	-RW--	-RW--
00100000	00035000	Reserved		PRV		-RW--
00135000	0000B000			PRV	-RW-G	-RW--
00140000	00035000	Reserved		PRV		-RW--
00175000	0000B000			PRV	-RW-G	-RW--
00180000	0000C000			PRV	-RW--	-RW--
0018C000	00004000	Reserved (00180000)		PRV		-RW--
00190000	00035000	Reserved		PRV		-RW--
001C5000	0000B000			PRV	-RW-G	-RW--
00200000	00153000	Reserved		PRV		-RW--
00353000	0000E000			PRV	-RW--	-RW--
00361000	0009F000	Reserved (00200000)		PRV		-RW--
00400000	00001000	bispookeygen2.exe		IMG	-R---	ERWC-
00401000	00001000	".data"	Initialized data	IMG	-RW--	ERWC-
00402000	00001000	".code"		IMG	ERW--	ERWC-
00403000	00001000	".idata"	Import tables	IMG	-RW--	ERWC-
00404000	00004000	".rsrc"	Resources	IMG	-R---	ERWC-
00410000	000C9000	\Device\HarddiskVolume4\Windows\		MAP	-R---	-R---

ניתן לראות שאזור הקוד של Bispookeygen2 מתחיל ב0x402000 ונגמר ב0x4021000.

נפתח את הקובץ בida:



ניתן לראות שמשהו לא הגיוני פה. אזור הקוד נפתח ונסגר מיד בלי כלום.

אם נגלול למעלה נראה את אזור הקוד האמתי:

```
IDA View-A   Hex View-1   Structures   Enums   Imports
.data:0040122F                 align 1000h
.data:0040122F _data           ends
.data:0040122F
.code:00402000 ; Section 2. (virtual address 00002000)
.code:00402000 ; Virtual size                  : 00000688 (   1672.)
.code:00402000 ; Section size in file          : 00000800 (   2048.)
.code:00402000 ; Offset to raw data for section: 00000800
.code:00402000 ; Flags E0000020: Text Executable Readable Writable
.code:00402000 ; Alignment     : default
.code:00402000 ; ===========================================================================
.code:00402000
.code:00402000 ; Segment type: Pure code
.code:00402000 ; Segment permissions: Read/Write/Execute
.code:00402000 _code           segment para public 'CODE' use32
.code:00402000                 assume cs:_code
.code:00402000                 ;org 402000h
.code:00402000                 assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing
.code:00402000                 call    sub_402439
.code:00402005                 push    0
.code:00402007                 push    offset sub_402029
.code:0040200C                 push    0
.code:0040200E                 push    25h
.code:00402010                 push    eax
.code:00402011                 call    ds:DialogBoxParamA
.code:00402017                 push    0
.code:00402019                 call    ds:ExitProcess
.code:0040201F ; ---------------------------------------------------------------------------
.code:0040201F                 call    sub_402567
.code:00402024                 call    loc_40266A
.code:00402029
.code:00402029 ; =============== S U B R O U T I N E =======================================
.code:00402029
.code:00402029 ; Attributes: bp-based frame
.code:00402029
.code:00402029 ; int __stdcall sub_402029(HWND hDlg, int, int, int)
.code:00402029 sub_402029      proc near               ; DATA XREF: .code:00402007↑o
0000062F 000000000040122F: .data:0040122F (Synchronized with Hex View-1)
```

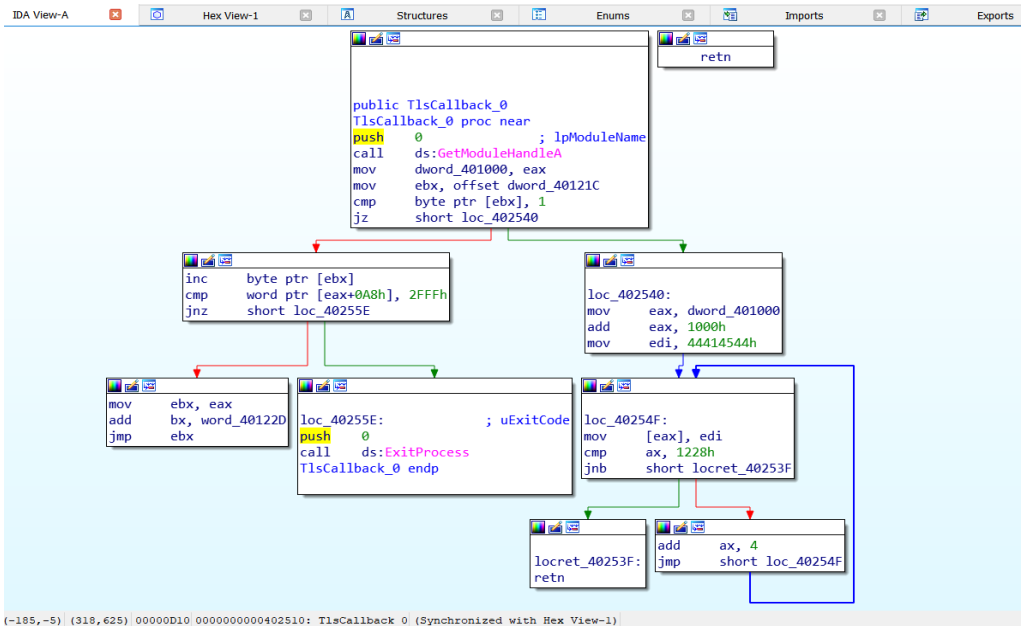
נשים לב לחלון משמאל שמביא לנו את שמות הפונקציות:

Functions window

Function name	Segment	Start	Length	Locals	Arguments	R	F	L	S	B	T	=
sub_402029	.code	0000000000402029	0000034F	00000010	0000000C	R	.	.	.	B	T	.
DialogFunc	.code	0000000000402378	00000041	00000010	0000000C	R	.	.	.	B	T	.
sub_4023B9	.code	00000000004023B9	00000021	00000004	00000004	R	.	.	.	B	.	.
sub_4023DA	.code	00000000004023DA	0000005F	00000004	00000004	R	.	.	.	B	.	.
sub_402439	.code	0000000000402439	0000005F	00000000	00000000	R	.	.	.	.	.	.
sub_402498	.code	0000000000402498	00000078	00000000	00000000	R	.	.	.	.	.	.
TlsCallback_0	.code	0000000000402510	00000056	00000000	00000000	R	.	.	.	.	.	.
nullsub_1	.code	0000000000402566	00000001	00000000	00000000	R	.	.	.	.	.	.
sub_402567	.code	0000000000402567	000000EE	00000000	00000000	R	.	.	.	.	.	.

ניתן לראות בין הפונקציות את הפונקציה TlsCallback_0.

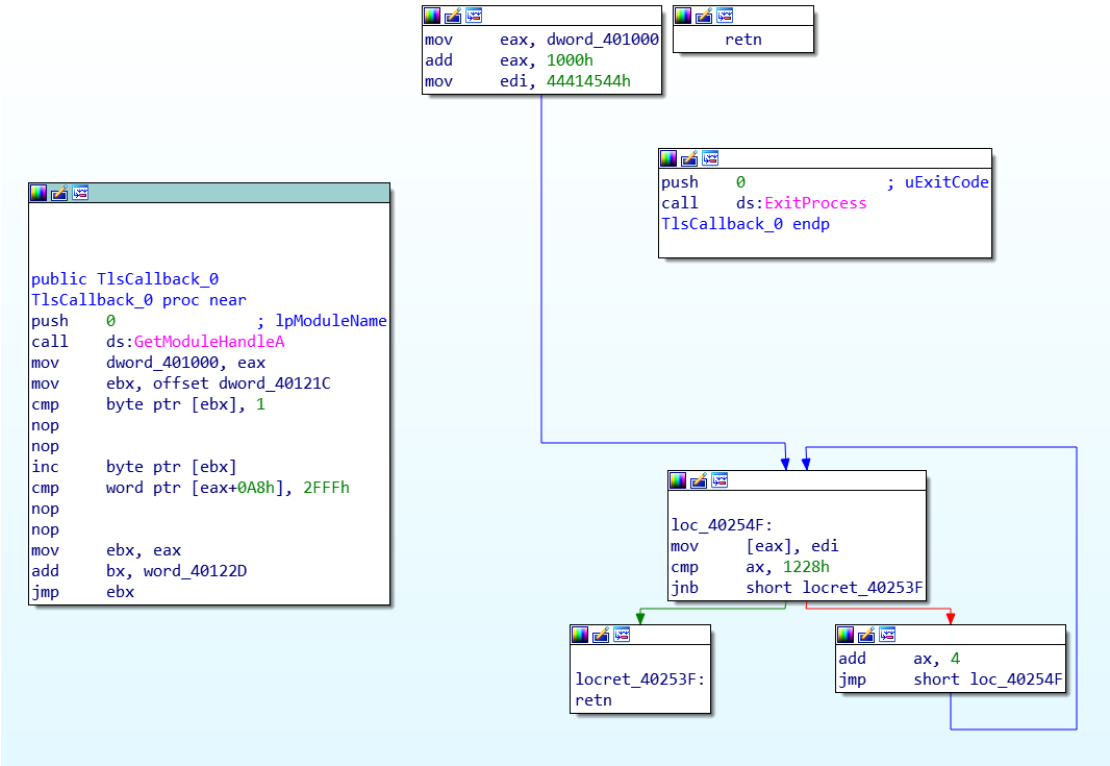
נכנס אליה ונראה מה היא עושה:



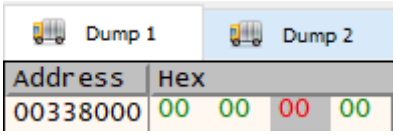
בענף השמאלי ביותר יש עדכון של הEntryPoint לערך הנכון (401000+1000=402000).

בענף הימני ביותר יש דריסה של אזור הdata. בענף האמצעי יש קריאה לפונקציה ExitProcess.

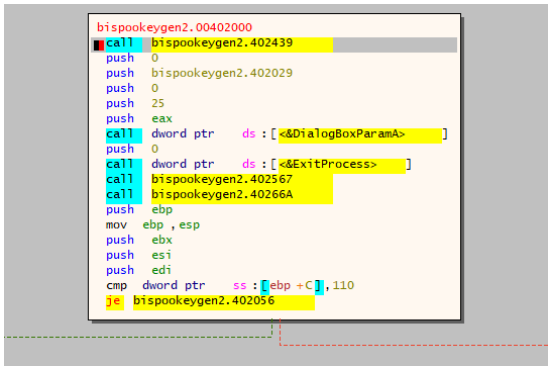
נפצפץ את הקוד באופן המתואר בתמונה הבאה, כך שתמיד נעבור דרך הענף השמאלי ביותר:



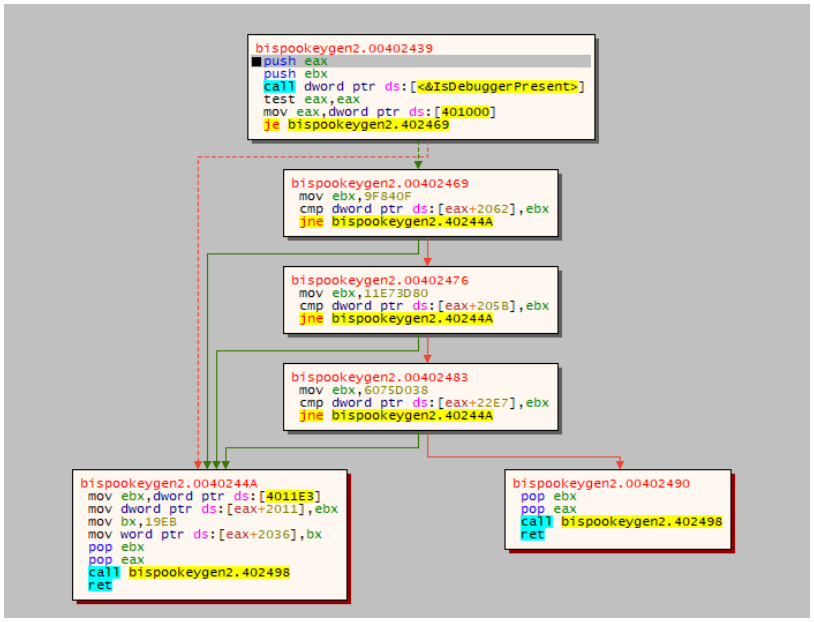
נפתח את הקובץ בexe32dbg, ודבר ראשון נשנה את הpeb ליתר ביטחון:



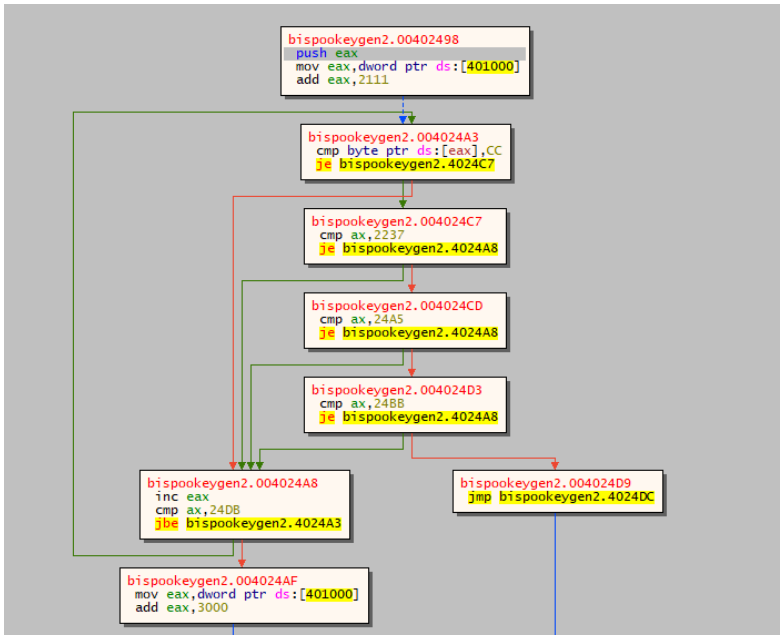
נשים bp בכתובת 0x402000 ונריץ:



נכנס לתוך הפונקציה בשורה הראשונה:



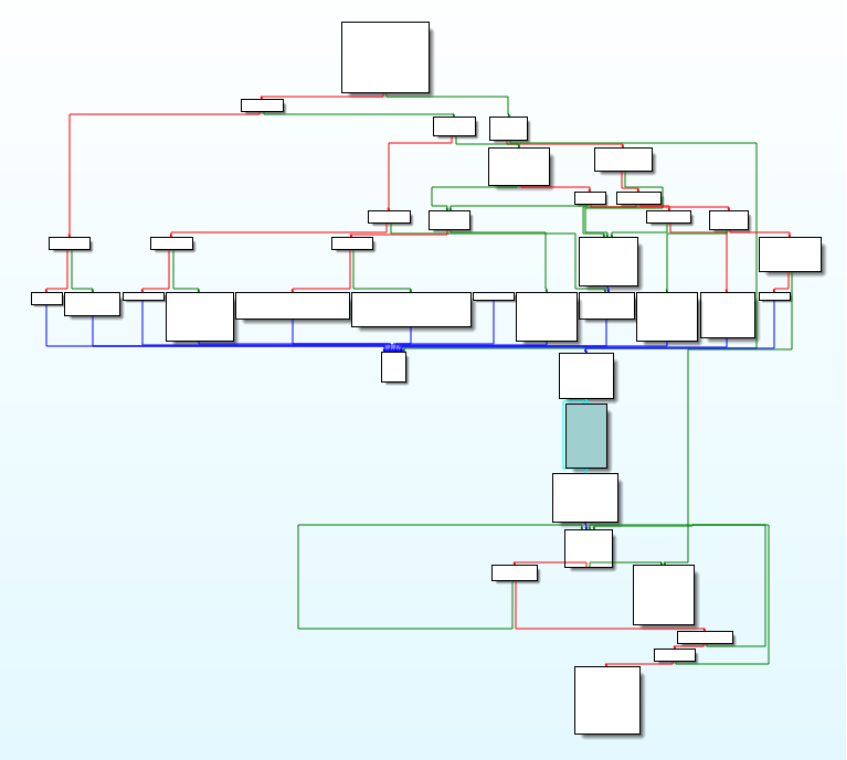
בשורה השלישית ניתן לראות שקוראים לפונקציה IsDebuggerPresent, כלומר זוהי פונקציה שבודקת אם אנו מדבגים את הקוד. בהמשך ניתן לראות בדיקות אם קוד בכתובות מסוימות נשאר כפי שהוא או פוצפץ. אם ניכנס לפונקציה בענף הימני ביותר למטה נראה בדיקות נוספות (CC) שקשורות לantidebug:



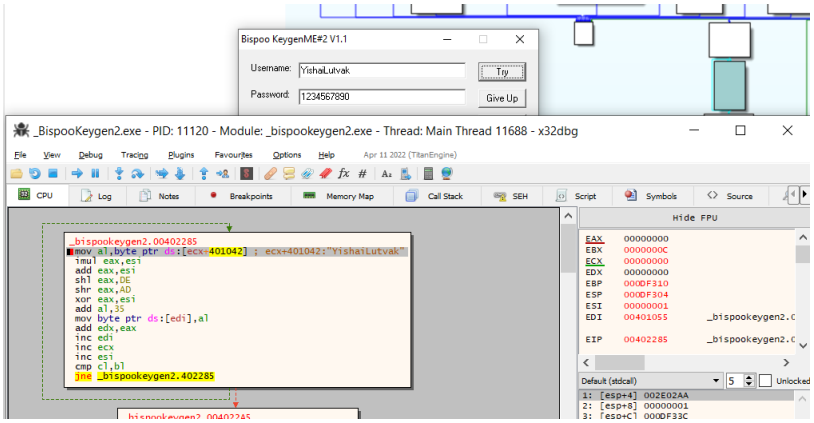
נפצפץ את הקוד על ידי שינוי הפקודה הראשונה בפונקציה לפקודה ret:

לפני	אחרי
00402439 50 push eax 0040243A 53 push ebx 0040243B FF15 763 call dword ptr ds:[<&IsDebuggerPresent>] 00402441 85C0 test eax,eax 00402443 A1 00104 mov eax,dword ptr ds:[401000] 00402448 74 1F je bispookeygen2.402469 0040244A 8B1D E31 mov ebx,dword ptr ds:[4011E3] 00402450 8998 112 mov dword ptr ds:[eax+2011],ebx	00402439 C3 ret 0040243A 53 push ebx 0040243B FF15 763 call dword ptr ds:[<&IsDebuggerPresent>] 00402441 85C0 test eax,eax 00402443 A1 00104 mov eax,dword ptr ds:[401000] 00402448 74 1F je _bispookeygen2.402469 0040244A 8B1D E31 mov ebx,dword ptr ds:[4011E3] 00402450 8998 112 mov dword ptr ds:[eax+2011],ebx

נפתח את הפונקציה שנשלחת לDialogBoxParamA בתור פרמטר (הפונקציה בכתובת 0x402029) באמצעות ida, כדי לראות איפה כדאי לשים בה bp לפני שנריץ את הקובץ המפוצפץ שלנו בx32dbg.



ממבנה הקוד וסקירה קצרה נראה שכדאי לשים bp בתחילת המלבן שסימנו בכחול. שם יש לולאה שאולי תהיה מעניינת, בנוסף נמלא את שם המשתמש והסיסמא כמתואר בתמונה הבאה:

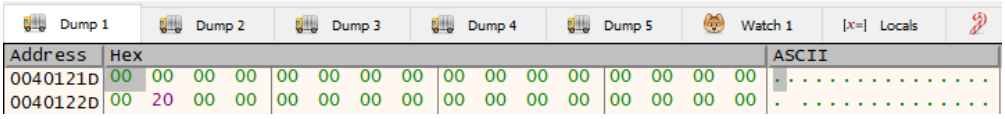


נשים לב ששם המשתמש נכנס לזיכרון בכתובת 0x401042 (ניתן לראות זאת בשורה הראשונה במלבן הראשון בתמונה להלן), והסיסמא נכנסה לזיכרון בכתובת 0x401004 (ניתן לראות זאת בשורה הראשונה במלבן השלישי בתמונה להלן).

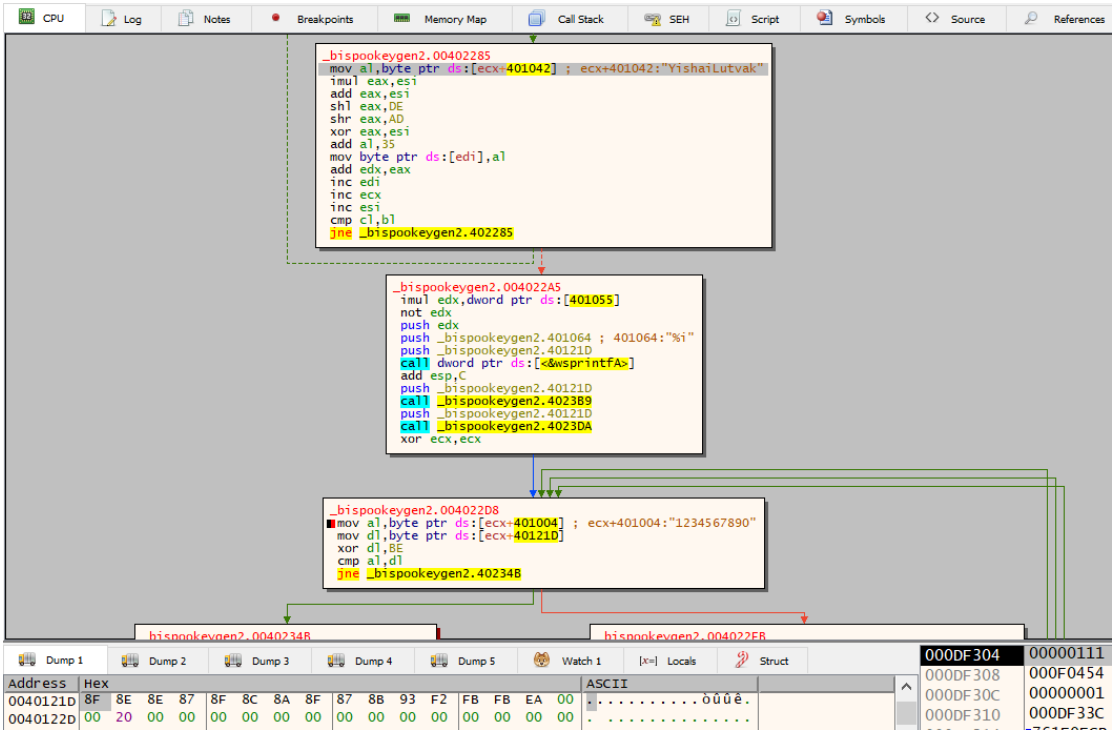
אם נסתכל נסקור את שאר הקוד באזור נוכל להבין שהקטע הקריטי ביותר נמצא במלבן השלישי לעיל. שם נעשית בדיקה האם הסיסמא מתאימה למחרוזת שנמצאת בזיכרון בכתובת 0x40121D **לאחר שאנו מבצעים על כל תו בה פעולת xor עם הערך BE** (ecx עולה בכל איטרציה ב1 כדי לרוץ במקביל על שתי המחרוזות עד הסוף).

מן הסתם המחרוזת שנמצאת בכתובת 0x40121D מושפעת משם המשתמש שהזנו, כך שלכל שם משתמש תהיה סיסמא שונה.

נסתכל מה יש כרגע בכתובת 0x40121D:



נסיר את הbp ששמנו בשורה הראשונה במלבן **הראשון** (שהופיע בתמונה לעיל), ונשים bp חדש בשורה הראשונה במלבן **השלישי** (שהופיע בתמונה לעיל) ונריץ:



ניתן לראות בתמונה כי הזיכרון בכתובת 0x40121D התמלא ואורכו הוא 15 בתים.

נכתוב סקריפט בפייתון שיבצע xor על הערכים האלו עם BE ויחזיר לנו את המחרוזת שיוצאת:

```
BispooKeygen2.py
cryptYishaiLutvak = 0x8f, 0x8e, 0x8e, 0x87,\
        0x8f, 0x8c, 0x8a, 0x8f,\
        0x87, 0x8b, 0x93, 0xf2,\
        0xfb, 0xfb, 0xea,

password = ''
for ch in cryptYishaiLutvak:
    password += (chr(int(ch) ^ 0xBE))
print(password)
```

נקבל את הסיסמא הבאה:

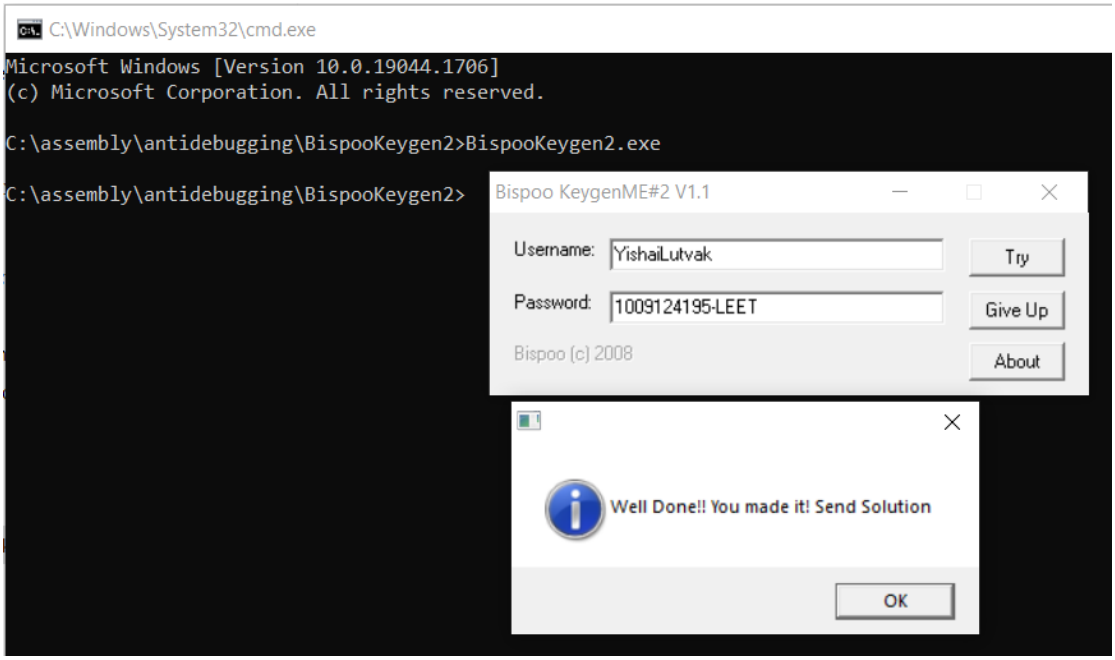
```
Run: BispooKeygen2
C:\Networks\Python3.8\python.exe C:/assembly/antidebugging/BispooKeygen2/BispooKeygen2.py
1009124195-LEET

Process finished with exit code 0
```

נריץ את הקובץ המקורי (לפני הפצפוץ) בcmd

כאשר נזין את שם המשתמש: "YishaiLutvak"

ואת הסיסמא: "1009124195-LEET"

ונבדוק מה קורה:

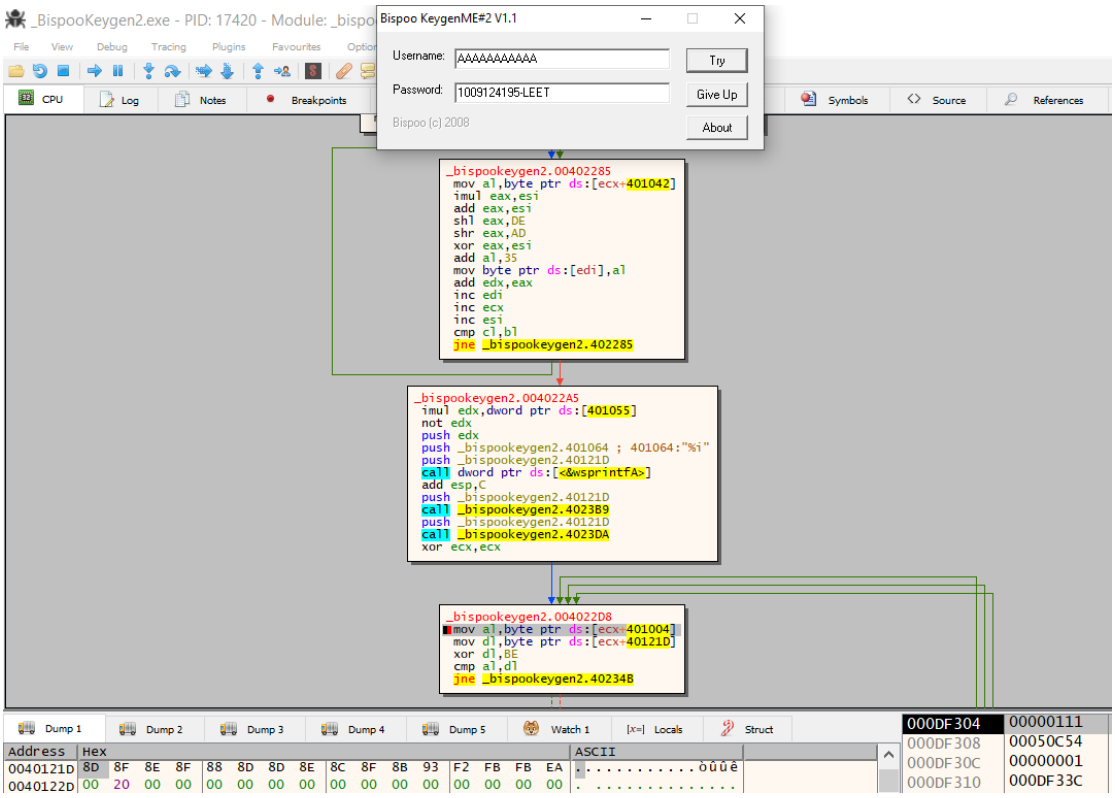


זה נראה שהצלחנו!!

נוודא רק שההנחה שתוכן הזיכרון בכתובת 0x40121D מושפע משם המשתמש נכונה:

נריץ שוב את הקובץ המפוצפץ בx32dbg, כאשר אנו משאירים את הbp ששמנו בכתובת 0x4022D8

ומזינים שם משתמש שונה:



כבר מהאורך של המחרוזת ניתן להבחין שהיא שונה ממה שהיה לנו קודם.

מכאן נראה בסבירות גבוהה כי ההנחה שלנו שתוכן הזיכרון בכתובת 0x40121D מושפע משם המשתמש אכן נכונה.