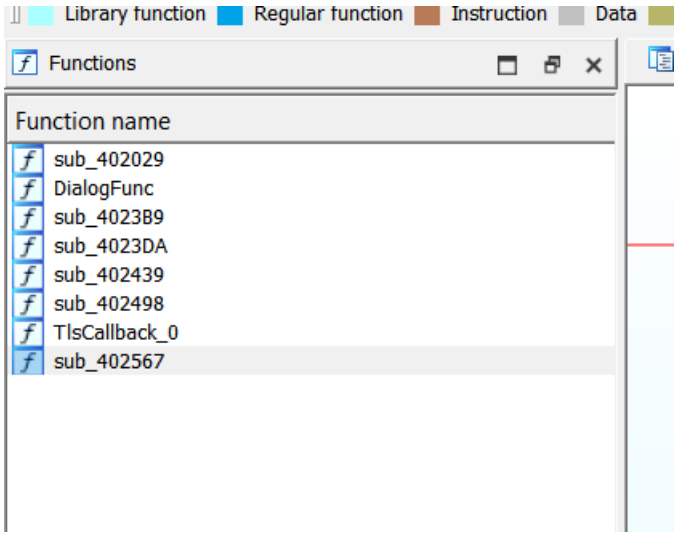
Write Up Anti Debug

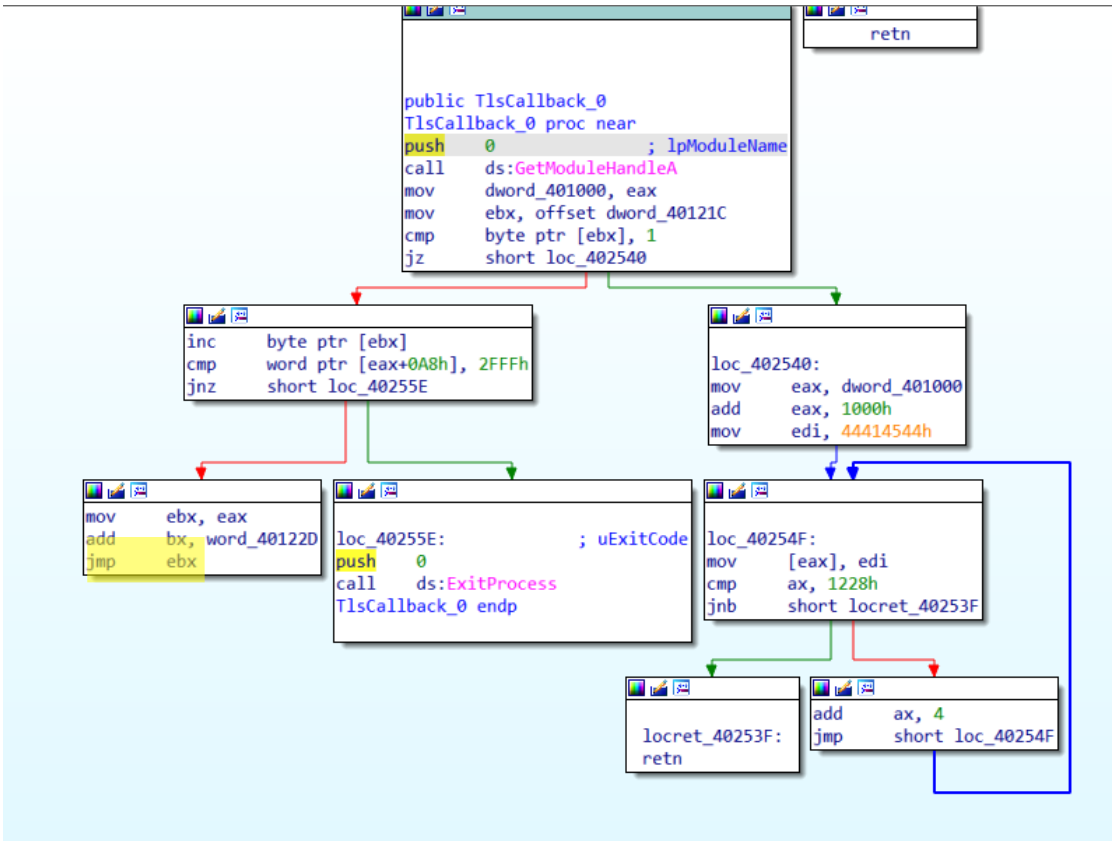
(1

את התרגיל אני אנתח עם IDA32X



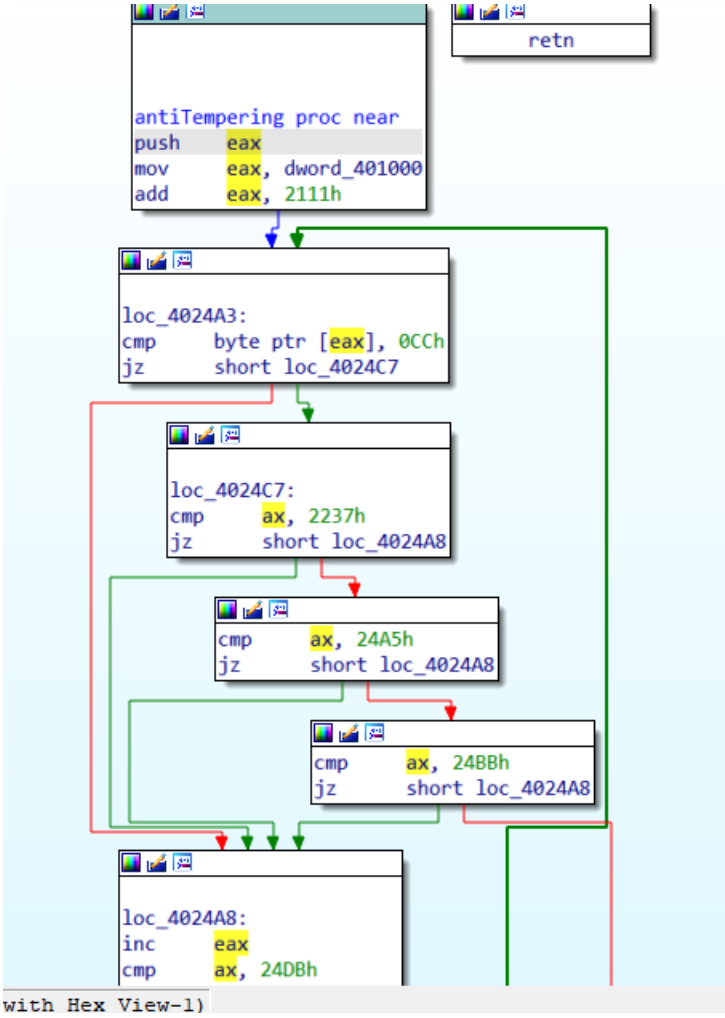
יש לנו מספר קטן מאוד של פונקציות,

אפשר ככה לנתח כל אחת ולראות מה מעניין אותנו:

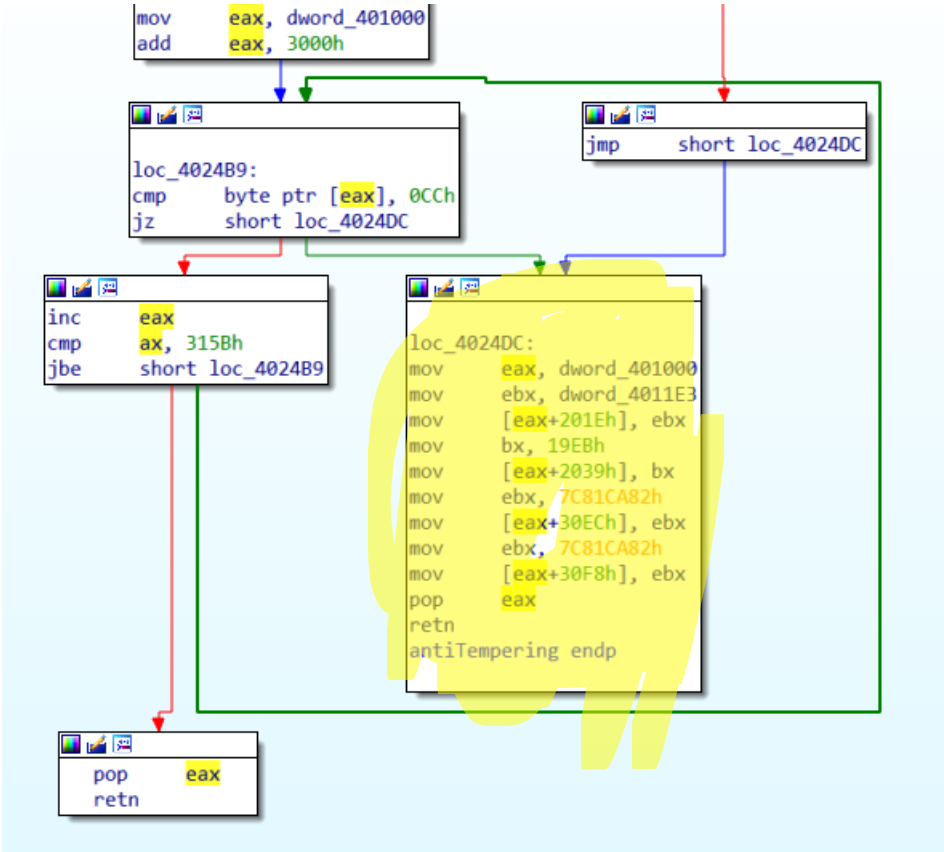


זה הTLS אין כאן קוד זדוני שרוצה לבדוק שאנחנו לא מדבגים אותו אבל יש קפיצה לקוד מצד שמאל

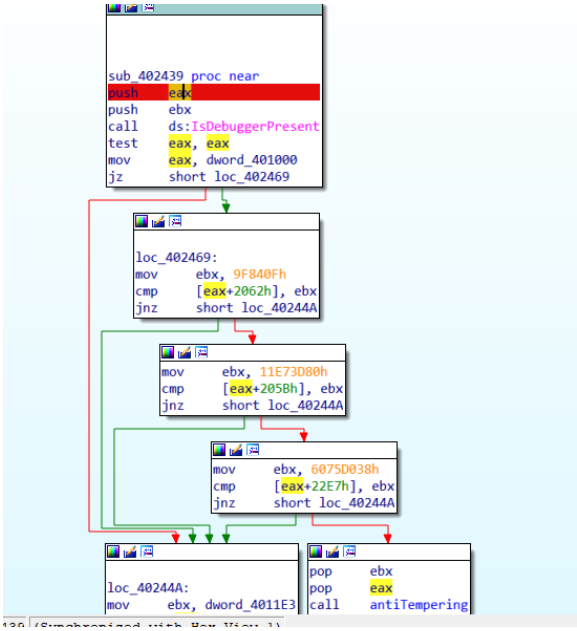
הפונקציה המעניינת הבאה:



כאן אפשר לראות שיש בדיקה שלא שינינו את הקוד של הפונקציה שלנו במקומות אסטרטגיים.



במקרה שכן שינינו את הקוד התוכנית תלך לאזור המודגש ושם היא תשנה את הקוד באזור מסוים(על פי ההרצה בהמשך זה כדי לצאת מהתהליך)



כאן יש לנו את הבדיקה הראשית אם הדיבאגר פעיל, שאפשר לעבוד על זה בעזרת שינוי של הביט.

ואחר כך יש בדיקה של שורות קוד שלא שינינו ואם כן אנחנו נצטרך לעזוב.

כל מנגנון בעיקרון אפשר לשנות בפני עצמו, למשל את הבדיקה אם הדיבאגר פעיל אפשר לדרוס בפני עצמו. או את הבדיקה של הבתים אפשר להמשיך הלאה בלי לבדוק את התנאי או לא לשנות את התוכנית אם זיהינו שאנחנו מדובאגים ע"י תהליך זר.

2) אני בחרתי דווקא לדרוס את הקריאות שיש לפונקציה של הבדיקה. בשביל לעשות את זה פתחתי

```
bispookeygen2.00402000
call bispookeygen2.402439
push 0
push bispookeygen2.402029
push 0
push 25
push eax
call dword ptr ds:[<&DialogBoxParamA>]
push 0
call dword ptr ds:[<&ExitProcess>]
call bispookeygen2.402567
call bispookeygen2.40266A
push ebp
mov ebp,esp
push ebx
push esi
push edi
cmp dword ptr ss:[ebp+C],110
je bispookeygen2.402056
```

בשלב הזה את הX32 וראיתי שיש קריאות לפני הקוד המרכזי לפונקציה של הבדיקה, אך אבוי! הפונקציה שנקראת ראשונה היא הזאת שבודקת האם אני בכלל בתהליך דיבוג, ואכן מה שקורה אחרי שאני מריץ אותה זה:

```
bispookeygen2.00402000
call bispookeygen2.402439
push 0
push bispookeygen2.402029
push 0
push 25
push eax
call dword ptr ds:[<&ExitProcess>]
push 0
call dword ptr ds:[<&ExitProcess>]
call bispookeygen2.402567
call bispookeygen2.40266A
push ebp
mov ebp,esp
push ebx
push esi
push edi
cmp dword ptr ss:[ebp+C],110
jmp bispookeygen2.402051

bispookeygen2.00402051
jmp bispookeygen2.402107

bispookeygen2.00402107
pop edi
pop esi
pop ebx
leave
ret 10
```

והתוכנית שלנו תצא מיד אחרי הקריאה לפונקציה הזאת, כמובן שאנחנו לא רוצים לצאת מיד ולכן אני דרסתי את הקריאה לפונקציה(גם כי לא היה אף קטע קריטי שיש לבצע בפונקציה.)

```
bispoon keygen intesting.00402000
nop
nop
nop
nop
nop
push 0
push bispoon keygen intesting.402029
push 0
push 25
push eax
call dword ptr ds:[<&DialogBoxParamA>]
push 0
```

ככה עשיתי לשאר הקריאות שיש(עוד איזה 2-3 פעמים)

איך גיליתי את הסיסמה:

```
bispoon keygen intesting.004022D8
mov al,byte ptr ds:[ecx+401004] ; ecx+401004:"123456"
mov dl,byte ptr ds:[ecx+40121D]
xor dl,BE
cmp al,dl
jne bispoon keygen intesting.40234B
```

זיהיתי שעושים פה השוואה בין מה שיוצא

מהקוד הקודם לסיסמה שמכניסים

```
Hex                                                ASCII
31 32 33 34 35 36 00 00 00 00 00 00 00 00 00 00    123456.
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
00 00 00 F4 00 00 00 10 25 40 00 F4 00 00 00
8D 86 87 8D 87 8E 8D 8E 88 8F 93 F2 FB FB EA
```

למעשה אנחנו משווים בין 2 המחרוזות האלה

רק שאנחנו עושים קסור של BE עם ההאש שיוצא בלמטה

אז יש לי סקריפט פייתון שמחשב את הסיסמה על פי ההאש:

```
print("".join(list(map(lambda a: chr((int(a,16)^int("be",16))), ['8d','86','87', '8d',
                                                                 '87' ,'8e' ,'8d', '8e',
                                                                 '88', '8f','93',
                                                                  'F2','fb', 'FB', 'EA']))))
```

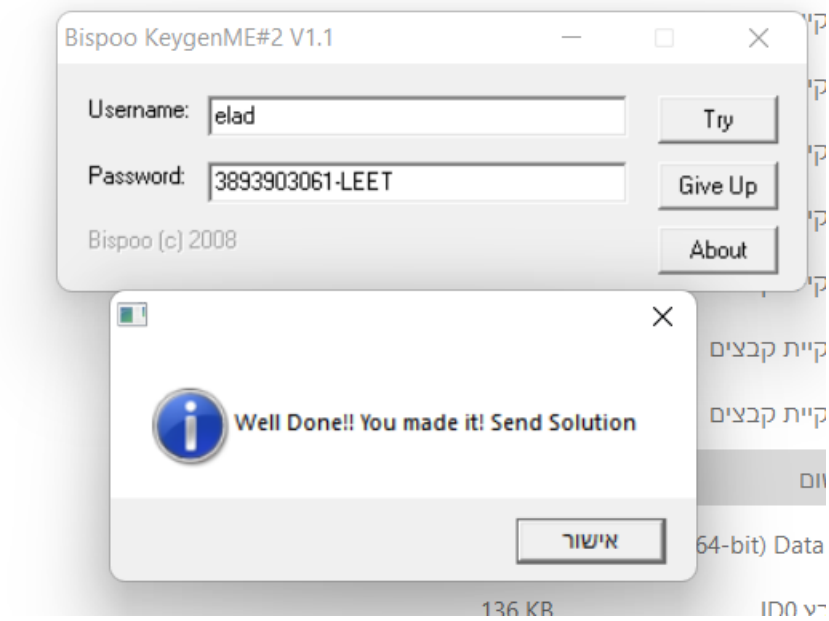
למעשה הקלט יהיה הבתים והוא מחשב קסור של BE עם כל אחד מתווים ומדפיס את הסיסמה.

הנה הפלט של ההרצה

```
[51, 49, 52, 56, 51, 55, 50, 48, 54, 49, 45, 76, 69, 69, 84]
3893903061-LEET

Process finished with exit code 0
```

והסיסמה כשאני מריץ את זה



אגב מלהריץ עם עוד קלטים אני חושב שהסיסמה תלויה יותר באורך וגם שהיא תמיד עם הLEET בסוף וגם ההאש תמיד נראה דומה, אבל מצד שני המטרה שלי הייתה למצוא שם וסיסמה, ולמעשה אין טעם לחקור את קטעי הקוד המיותרים