מחקר Bispoo2 / ברק גונן

כאשר טוענים את הקובץ ל-ida הוא לא מזהה את נקודת ההתחלה.

```
.code:00402FFF start          db ?
.code:00402FFF _code          ends
.code:00402FFF
```

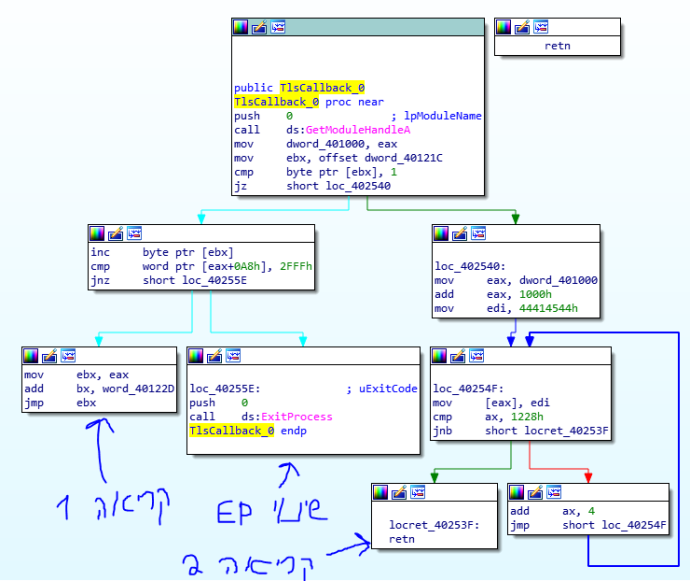
נתקלנו במנגנון obfuscation שנועד להקשות על הדיבוג. אפשר לראות ש-ida מזהה פונקציית TLS, שנקראת לפני ה-main.

פונקציית ה-TLS

הכתובת של הפונקציה היא 0x402510. לכן נטען את התוכנית ב-x32dbg, ונשים נקודת עצירה בכתובת של ה-TLS. מהרצת הקוד אפשר לראות שהקוד מחשב כתובת באמצעות סיכום של כתובת התוכנה בזכרון (0x400000) יחד עם קבוע 0x2000, ואז מתבצעת קפיצה לכתובת הזו- 0x402000.

דברים נוספים שאפשר ללמוד מאנליזה סטטית של פונקציית ה-TLS:

- יש בדיקה האם זו הפעם הראשונה שהפונקציה נקראת
- אם זו לא הפעם הראשונה שהפונקציה נקראת, היא דורסת את אזור הקוד ברצף 0x44414544, שהינו DEAD.
- באמצעות xref אפשר לראות שפונקציית ה-TLS אכן נקראת בהמשך. אם כך יכול להיות שצריך לפצפץ' אותה? הקריאה אליה נעשית רק בסיום הריצה, לכן אפשר לוותר
- יש בדיקה האם נקודת ההתחלה של הקוד אינה 2fff השגוי, כלומר אם מישהו ניסה לפצ'פץ את ה-entry point

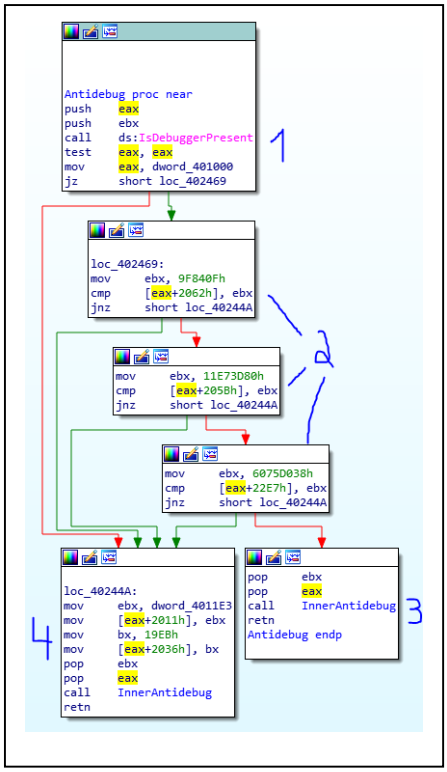


מנגנון האנטידיבאג

עם הרצת הקוד ב-x32dbg נשים BP בכתובת 0x402000.

```
00402000   E8 34040000   call bispookeygen2 - copy.402439
00402005   6A 00         push 0
00402007   68 29204000   push bispookeygen2 - copy.402029
0040200C   6A 00         push 0
0040200E   6A 25         push 25
00402010   50            push eax
00402011   FF15 E4304000 call dword ptr ds:[<&DialogBoxParamA>]
00402017   6A 00         push 0
00402019   FF15 6E304000 call dword ptr ds:[<&ExitProcess>]
0040201F   E8 43050000   call bispookeygen2 - copy.402567
00402024   E8 41060000   call bispookeygen2 - copy.40266A
```

לאחר הרצת הפונקציה בכתובת 0x402439 מסתיימת ריצת התוכנית. נחקור את מנגנון האנטידיבאג.



1. חלק 1 הוא בדיקת IsdebuggerPresent
2. בחלק 2 יש בדיקה האם קטעי קוד קריטיים השתנו. לדוגמה אם JE שונה ל-/JNE
3. בחלק 4 יש דריסה של הקוד. לדוגמה בכתובת 0x402011 נמצאת קריאה לפונקציה DialogBoxParam, שתוחלף בקריאה לפונקציה Exit.
4. המטרה היא להגיע לסוף של חלק 3, שמכיל קריאה לפונקציית אנטידיבג נוספת, שבודקת אם יש BP (קוד CC) בכל מקום בקוד, חוץ מאשר בנקודות בהן יש ממילא CC. אם כן, הפונקציה דורסת את הקוד. פונקציה זו נקראת רק מתוך האנטידיבאג הראשי (אפשר לוודא ע"י xref), לכן אם מפצפצים את הפונקציה 0x402439 אפשר לדלג גם עליה.

לאחר שוידאנו שהפונקציות לא משנות שום ערך ב- data section, שאולי נזדקק לו בהמשך, נעקוף את המנגנון באמצעות RET בתחילת הפונקציה.

הסיבה לכך שאנחנו עוקפים עם RET ולא מבטלים את הקריאה לפונקציה הזו, היא שהפונקציה נקראת פעמים רבות (שוב, אפשר לראות עם xref)

<u>פענוח מנגנון הסיסמה</u>

מעכשיו נעבוד עם הקובץ המפוצ'פץ.

הקריאה ל-DialogBoxParam כוללת את כתובת הפונקציה הנקראת- 0x402029.

אפשר לזהות בקוד הארוך הדפסה של מחרוזות מענינות שקשורות לאורך הקלט שהמשתמש מזין. לדוגמה שהאורך לא גדול מ-15 (0xF).

נשים נקודת עצירה במקום בו מתבצעת ההשוואה ל-15, מכיוון שזה מבטיח שנעצור שם לאחר הזנת הקלט (אם אכן ביטלנו את כל מנגנוני האנטידיבאג). נזין

Username: abcdef

Password: 123456

```
bispookeygen2 - copy.0040212C
cmp eax,F
ja bispookeygen2 - copy.4021FD
```

המשך ההרצה יביא אותנו אל קטע קוד שממנו רואים שאחד הקלטים שהזנו נמצא בכתובת 0x401042. באמצעות צפיה ב-dump בכתובת הזו רואים את הרצף abcdef.

```
bispookeygen2 - copy.00402135
mov byte ptr ds:[401040],al
mov al,byte ptr ds:[401042]
cmp al,20 ; 20:' '
je bispookeygen2 - copy.4021CF
```

הקוד ממשיך לקריאה נוספת ל-GetDlgItemText, בשלב זה די הגיוני שמה שייקרא זו הסיסמה, והיא אכן מופיעה במקום 0x401004, הכתובת שנמסרת כפרמטר.

```
bispookeygen2 - copy.00402147
 call bispookeygen2 - copy.402439
 push 15
 push bispookeygen2 - copy.401004
 push 12E
 push dword ptr ss:[ebp+8]
 call dword ptr ds:[<&GetDlgItemTextA>]
■cmp eax,10
 ja bispookeygen2 - copy.40234B
```

ההשוואה של אורך הסיסמה עם 0x10 מביאה אותנו לפיצול, החלק הימני דוחף מחרוזת מוצפנת שנמצאת בכתובת 0x4011c4, מחרוזת שנראית מוצפנת.

```
■cmp eax,10
 ja bispookeygen2 - copy.40234B

bispookeygen2 - copy.0040216A
 jmp bispookeygen2 - copy.402269

bispookeygen2 - copy.0040234B
 push bispookeygen2 - copy.4011C4
 call bispookeygen2 - copy.4023B9
 push 20
 push bispookeygen2 - copy.4011C4
 push 12E
 push dword ptr ss:[ebp+8]
 call dword ptr ds:[<&SetDlgItemTextA>]
 push bispookeygen2 - copy.4011C4
 call bispookeygen2 - copy.4023B9
 leave
 ret 10
```

רואים שהפונקציה שנקראת מבצעת xor למחרוזת עם BE (הסיומת של DEADBABE) ומפענחת אותה.

תוצאת הפענוח תודפס בהמשך קטע הקוד שבכתובת 0x40234B.

מכך אפשר להבין שזה קטע הקוד שמדפיס "כשלון". לקטע הקוד זה נכנס חץ נוסף, מלמטה, מהמקום שבו נבדק אם הסיסמה תואמת לשם המשתמש.

```
; Attributes: bp-based frame

XorBy0xBE proc near

arg_0= dword ptr  8

push    ebp
mov     ebp, esp
xor     ecx, ecx
xor     edi, edi
mov     edi, [ebp+ecx+arg_0]

loc_4023C4:
mov     al, [edi+ecx]
cmp     al, 0
jz      short locret_4023D6

xor     eax, 0DEADBABEh
mov     [edi+ecx], al
inc     ecx
jmp     short loc_4023C4

locret_4023D6:
leave
retn    4
XorBy0xBE endp
```

כעת יש שתי אפשרויות להשוואה ששם המשתמש תואם לסיסמה:

1. Hash(Username) == Password
2. Hash(Password) == Username

קטע הקוד הבא אכן מבצע Hash לשם המשתמש, רואים שמוזן לתוכו כתובת 0x401040 בה נשמר אורך המחרוזת, ולאחר מכן פעולות לוגיות על התווים במחרוזת. התוצאה נשמרת ב-0x401055 והלאה.

```
bispookeygen2 - copy.00402269
 xor ecx,ecx
 xor eax,eax
 xor ebx,ebx
 xor edi,edi
 xor esi,esi
 mov edx,eax
 mov edi,bispookeygen2 - copy.401055
 mov esi,1
 mov bl,byte ptr ds:[401040]

bispookeygen2 - copy.00402285
■mov al,byte ptr ds:[ecx+401042]
 imul eax,esi
 add eax,esi
 shl eax,DE
 shr eax,AD
 xor eax,esi
 add al,35
 mov byte ptr ds:[edi],al
 add edx,eax
 inc edi
 inc ecx
 inc esi
 cmp cl,bl
 jne bispookeygen2 - copy.402285
```

תוצאת ה-Hash מועתקת למקום אחר בזיכרון, 0x40121D, ומתבצע עליה Hash נוסף, בפונקציה 0x4023DA.

```
bispookeygen2 - copy.004022A5
imul edx,dword ptr ds:[401055]
not edx
push edx
push bispookeygen2 - copy.401064
push bispookeygen2 - copy.40121D
call dword ptr ds:[<&wsprintfA>]
add esp,C
push bispookeygen2 - copy.40121D
call bispookeygen2 - copy.4023B9
push bispookeygen2 - copy.40121D
call bispookeygen2 - copy.4023DA
xor ecx,ecx
```

```
bispookeygen2 - copy.004022D8
mov al,byte ptr ds:[ecx+401004]
mov dl,byte ptr ds:[ecx+40121D]
xor dl,BE
cmp al,dl
jne bispookeygen2 - copy.40234B
```

הנה הנקודה בה הסיסמה מושווית תו תו ל-Hash של שם המשתמש, לא לפני שהוא עובר XOR נוסף עם BE. בשלב זה אפשר כבר לחלץ את הסיסמה המתאימה לכל שם משתמש. מהמקום 0x40121D נעתיק לתוך פייתון את התווים, לרשימה בשם x.

```
pass = ""

for i in x:

  pass += chr(i^0xbe)
```

ועבור המשתמש abcdef הסיסמה היא

3953228419-LEET

```
bispookeygen2 - copy.00402303
push esp
push bispookeygen2 - copy.401064
push bispookeygen2 - copy.40121D
call dword ptr ds:[<&wsprintfA>]
add esp,C
push bispookeygen2 - copy.40119D
call bispookeygen2 - copy.4023B9
call bispookeygen2 - copy.402439
push 1040
push bispookeygen2 - copy.401067
push bispookeygen2 - copy.40119D
push 0
call dword ptr ds:[<&MessageBoxA>]
push bispookeygen2 - copy.40119D
call bispookeygen2 - copy.4023B9
leave
ret 10
```

אם באיזשהו שלב בבדיקת הסיסמה יש טעות, הקוד קופץ לאזור שאותו הגדרנו לפני כן כ"כשלון".

אם הבדיקות עוברות בהצלחה, נגיע אל הקטע האחרון, אזור ה"הצלחה", בו מתפענחת מחרוזת ההצלחה בכתובת 0x40119D.

<u>פצפוץ' הקובץ</u>

יש דרכים רבות לפצפץ'. אחת מהן היא שבכתובת 0x40212C, המקום שבו יש את ההשוואה בין eax ו-0xF, אורך המחרוזת, נבצע קפיצה אל אזור "ההצלחה".

```
.code:0040212C E9 D2 01 00 00          jmp     good
```