

בְּשַׁבָּת וְיֶהוּדָה יְהוָה צְדָקָה

$$rSP = X \quad |(1) \quad -\rightarrow$$

rSP ∂E

push rbp ; save previous base pointer
mov rbp, rdi ; set new base pointer
push rbp ; save current base pointer
sub rbp, 8 ; move 8 bytes down
push rbp ; save current base pointer
add rbp, 8 ; move 8 bytes up
pop rbp ; restore previous base pointer
add rbp, 8 ; move 8 bytes up
pop rbp ; restore previous base pointer

```

27 .section .text
28 .global _start
29 _start:
30     mov $8, %esi
31     mov $A, %rdi
32     call func
33     movq $60, %rax
34     movq $0, %rdi
35     syscall
36
37 func:
38     pushq %rbp
39     movq %rsp, %rbp
40     cmp (%rdi), %esi
41     jne continue
42     mov $1, %eax
43     jmp finish
44
45     continue:
46         cmpq $0, 4(%rdi)
47         je next
48         pushq %rdi
49         mov 4(%rdi), %rdi
50         call func
51         pop %rdi
52         cmp $1, %eax
53         je finish
54
55     next:
56         cmpq $0, 12(%rdi)
57         je fail
58         pushq %rdi
59         mov 12(%rdi), %rdi
60         call func
61         pop %rdi
62         cmp $1, %eax
63         je finish

```

$$Esi = 8$$

$$rdi = A, B, D, C, E$$

$$3 \neq 8 \quad 4 \neq 8 \quad 2 \neq 8 \quad 5 \neq 8 \quad 6 = 8$$

$$eax = 1 \Leftarrow E - e \\ J \uparrow J \uparrow N \\ J \uparrow J \uparrow N \\ K1.1 \\ data \rightarrow 8$$

.3

```

typedef struct _Node {
    int data;
    struct _Node *left;
    struct _Node *right;
} Node;

```

```

int func ( Node *root, int x){
    If (root->data == X)
        return 1;
    if (root->left != null)
        if (func (root->left, x))
            return 1;
    if (root->right != null)
        return func (root->right, x);
    return 0;
}

```

```
27 .section .text
28 .global _start
29 _start:
30     mov $8, %esi
31     mov $A, %rdi
32     call func
33     movq $60, %rax
34     movq $0, %rdi
35     syscall
36
37 func:
38     pushq %rbp
39     movq %rsp, %rbp
40     cmp (%rdi), %esi
41     jne continue
42     mov $1, %eax
43     jmp finish
44
45 continue:
46     cmpq $0, 4(%rdi)
47     je next
48     pushq %rdi
49     mov 4(%rdi), %rdi
50     call func
51     pop %rdi
52     cmp $1, %eax
53     je finish
54
55 next:
56     cmpq $0, 12(%rdi)
57     je fail
58     pushq %rdi
59     mov 12(%rdi), %rdi
60     call func
61     pop %rdi
62     cmp $1, %eax
63     je finish
```

$$\begin{array}{r} -8 \cdot 11 \cdot 5 \cdot 7 \cdot 23 \\ \times 4 \quad 178 \\ \hline \end{array}$$

C area .(

פתרונות ג' ימי מווינט מוחלט לבצע שינויים נוספים ואלה שוגרתים בקורס מל כל שינוי שגוי מציע עליים לכתוב האם נוכנות השגורה תיגע(האם יש קלט עבורי השגורה לאחר השינוי שונה מהשגרה לפני השינוי). הסבירו בקצרה את תשובתכם! (10 נקודות)

- a. מחיקת הפקודת push pop בשורות 60 ו 57.
- b. מחיקת הפקודה pop בשורה 60
- c. מחיקת push pop push/pop בשורות 51 ו 48
- d. הוספה פקודה push %rdi אחרי continue בשורה 45
- e. הוספה הפוקודה push %rdi אחרי continue בשורה 45, שינוי פוקדת הקומן בשורה 51 לפוקודה: mov %rsp, %rdi

```
.section .data
msg1: .ascii "HOW YOOOU DOOIN?"
msg2: .ascii "JOEY DOESN'T SHARE FOOD!"
msg1_len: .quad msg2 - msg1
msg2_len: .quad msg1 - len - msg2
all_msg_len: .quad msg1_len + msg2
```

.k

```
.section .text
.global _start
_start:
    mov $msg1, %rsi
    mov $1, %rdi
    mov $1, %rdx
    mov $1, %rax
    xor %rbx, %rbx

    movq msg1_len, %r9
    call Joey_func
```

ומוצגת כאן גם הפקציה שכותב:

```
Joey_func:    .L1:    .L2:  
        cmp %rbx, %r9  
        je end  
        # addb $0x20, (%rsi)  
        test $1, %rbx  
        jnz skip  
        syscall  
skip:    inc %rsi  
        inc %rbx  
        call Joey_func  
end:    ret
```

`fsi = msg1, msg1+1, msg1+2, ...`

10

$$1 \text{ kg} = 1$$

四百三

$$rbx=0, \quad rbx=1, \quad rbx=2$$

$$\cdot \text{b} = 17$$

HwYroudon

י' ג' צ' ס' ס' ס'

```
.section .data  
msg1: .ascii "HOW YOOOU DOOIN?"  
msg2: .ascii "JOEY DOESN'T SHARE FOOD!"
```

```
.section .text
.global _start
_start:
    mov $msg1, %rsi
    mov $1, %rdi
    mov $1, %rdx
    mov $1, %rax
    xor %rbx, %rbx
    movq msg1_len, %r11
    call Joey_func
```

ומוצגת כאן גם הפקציה שכותב:

Joey func:

```
    cmp %rbx, %r9
    je end
    addb $0x20, (%rsi)
    test $1, %rbx
    jnz skip
    syscall
skip: inc %rsi
      inc %rbx
      call Joey_func
end:  ret
```

```
.section .text
.global _start
_start:
    mov $msg1, %rsi
    mov $1, %rdi
    mov $1, %rdx
    mov $1, %rax
    xor %rbx, %rbx
        ah
    movq msg_len, %r9
    call Joey_func
```

ומוצגת כאן גם הפונקציה שכתב:

```
Joey_func:  
    cmp %rbx, %r9  
    je end  
    addb $0x20, (%rsi)  
    test $1, %rbx  
    jnz skip  
    syscall  
skip: inc %rsi  
    inc %rbx  
    call Joey_func  
end: ret
```

inc r9y ←

%rdi %01272 en-ten

פ' ב' מציעה להחליף את השימוש ב σ בשימוש ב dir . **alSyscall**

 מוניקה מציעה להחליף את השימוש ב9z בשימוש ב11z.

ררוש מציע להחליף את השימוש בgo בשימוש בget.
בgo נאומן go לעיר, לארון לארון לארון.

 ניתן למצוא להחליף את השימוש ב9 בשימוש באcxz.

 מוניקה מציעה להחליף את השימוש ב9z בשימוש ב11z.

✓ ריצ'ל מציעה להחליף את השימוש בgo בשימוש ב12r.

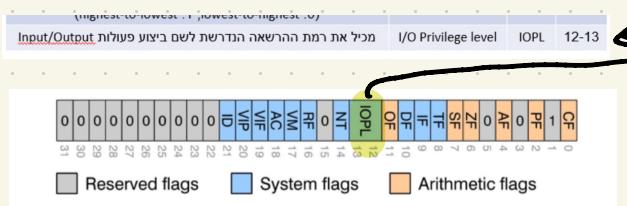
1. 1960-61, 1970-71, 1977-78, 1981-82, 1984-85, 1987-88, 1990-91, 1993-94.

סבב 3

א. רצף pushfq register SKI יוג'ג ה-> RSP. רצף \$ 2049, %RSP or \$ 2049, 2049 הינה סכום שלם ככינאי.

היא $2^{32} + \dots + 2^0 = 2^{32} - 1$. מכאן ניתן לראות כי סכום זה נזקק ל-12bits. אולם מוקדם מכך, בפונקציית ADD, מוחלט סכום סכום סכום.

ב. כדי לתקן את זה, נזקק שולחן IOPL גודל סך אמצעי.



ג. ה-IOPL מוגדר על ידי הערך 0xa0 (השווה ל-0). מוגדר גורם המאפשר גישה למשתנים. מוגדר גורם המאפשר גישה למשתנים.

ד. וויל החבר המוביל של הולி מطالب כיצד ניתן לחסום פסיקות תוכנה لكن הוא שואל את הולוי. אילו מבין התשובות הבאות על הולוי לענות לו? יש לשמן את האפשרות הנכונה. (5 נקודות)

1. כיבוי דגל IF באוגר הדגלים
2. הדלקת דגל IF באוגר הדגלים
3. שינוי CPL ל00
4. לא ניתן לחסום פסיקות תוכנה.

ה. כתע נתון שולי הצליח להציג מצב שבו CPL שווה ל-0. וויל מעוניין לחסום פסיקות חומרה שאין מועברות דרך כניסה NMI. כיצד הוא יכול לעשות זאת? (5 נקודות)

1. כיבוי דגל IF באוגר הדגלים
2. הדלקת דגל IF באוגר הדגלים
3. עלוי לחבר את הפסיקות לכיניסט NMI ואז לכבות את דגל IF
4. לא ניתן לחסום פסיקות חומרה ولكن לא יצילח.