# סדנת מבוא ל REVERSING

# היכרות ראשונית

- הורידו מאתר הקורס את הקובץ sampleNI.exe.
  - Crackme פשוט שמדפיס OK עבור סיסמה נכונה או Wrong אחרת.
  - מצאו את הסיסמה הנכונה.

- שנו את הקוד כך שידפיס OK עבור כל קלט.
  - בהתחלה יש לשנות את הקוד כפי שהוסבר בכיתה.
  - לאחר מכן חפשו מקום נוסף שניתן לשנות על מנת להשיג תוצאה דומה.

# מציאת קטע קוד רלוונטי

- נתון הקוד (הפשוט הבא) :

```
int main() {
        printf("Hello World\n");
}
```

- מצאו לפחות 3 דרכים שונות למצוא את קטע הקוד ב-EXE.
  - חשבו אם הדרכים שלכם היו עובדות גם עבור תוכניות גדולות עם הרבה קוד.
- הורידו מאתר הקורס את הקובץ printf.exe.

# פתרון

- מעבר על כל הקוד מההתחלה.
  - לא מומלץ ולא יעבוד בתוכניות גדולות (וכל תוכנית נחשבת גדולה).
- חיפוש מחרוזות בתוכנית ומציאת פקודות שמשתמשות במחרוזת.
  - במקרה שלנו נחפש Hello World.
- עכשיו ננסה למצוא את הקטע קוד בעזרת IDA (בדרכים דומות).

# מציאת קטע קוד רלוונטי
## חלק 2

- נתונה תוכנית שמדפיסה את הפלט הבא:

```
Hello World
```

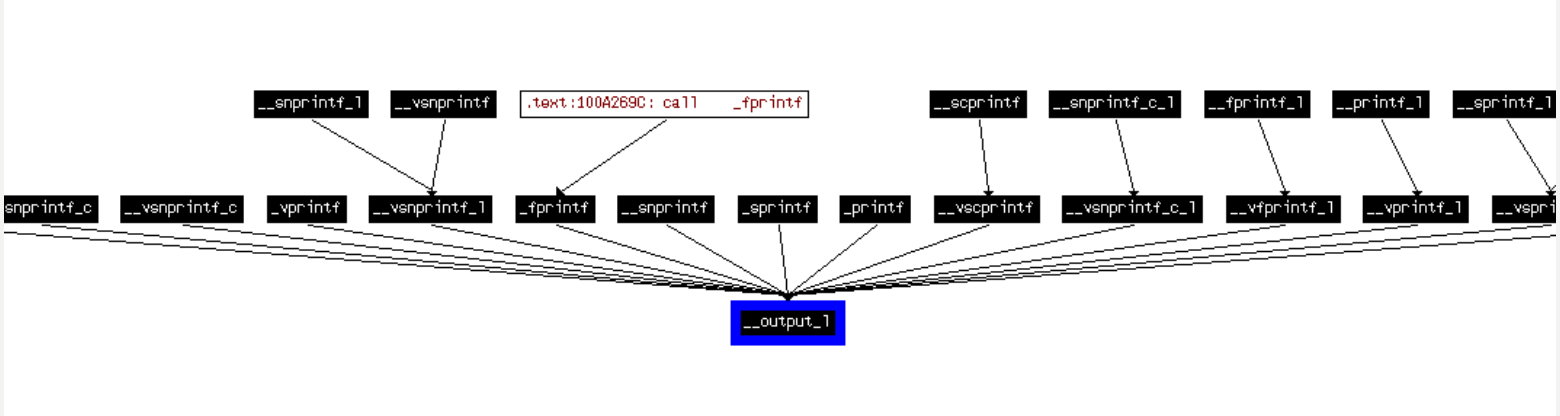
- המתכנת מצא דרך להדפיס מבלי להשתמש ב-printf.
  - ידוע שהוא השתמש בפונקציה שתומכת ב-format string כמו printf.
- כיצד ניתן למצוא את קטע הקוד הרלוונטי?
  - לבדוק את ה-import table על מנת לגלות דרך איזו פונקציה הדפיס.
  - מצאו דרך נוספת - יישמו את הפתרון שלכם על הקובץ printf.exe.

- רמז: איפה printf ממומש? מצאו מכנה משותף לסוג הפונקציות האלו.

# פתרון

- נרצה למצוא מכנה משותף לכל פונקציות ה-printf.
  - כלומר פונקציה משותפת שכל הפונקציות משתמשות בה.

- בהינתן פונקציה (כתובת), איך נדע אילו פונקציות קוראות לה?



# תרגיל

- נתון קובץ struct.exe.
  - הבינו את האסמבלי והוסיפו הערות מתאימות עם IDA.

- סוגי הערות:
  - Insert - הערה מעל השורה.
  - Shift + Insert – הערה מתחת לשורה.
  - ; הערה בצד.

# מערכים ומבנים – "פתרון"

```
                push    ebp
                mov     ebp, esp
                push    14h             ; size_t
                call    ds:malloc
                mov     ecx, [ebp+argc]
                mov     [eax], ecx
;
; var = malloc(0x14);
; *(int *)var = argc;
;
                xor     ecx, 7
                add     esp, 4
                mov     [eax+4], ecx
;
; *((int *)var)+1) = argc ^ 7;
;
                xor     ecx, ecx
                lea     edx, [eax+8]
;
; edx = ((int*)var) + 2;
;
                mov     edi, edi

loc_401020:                             ; CODE XREF: sub_401000+2A↓j
                lea     eax, [ecx+30h]
; ecx = 0;
; up:
; al = 0x30 + ecx;
; *(char*)(edx+ecx) = al;
; ecx++;
; if (ecx <= 0xa) goto up
;
                mov     [edx+ecx], al
                inc     ecx
                cmp     ecx, 0Ah
                jle     short loc_401020
                push    edx
                push    offset aS       ; "%s"
```

# מערכים ומבנים - פתרון

```
                push    ebp
                mov     ebp, esp
                push    14h             ; size_t
                call    ds:malloc
                mov     ecx, [ebp+argc]
                mov     [eax+struc_1.int1], ecx
;
; struct = malloc(0x14);
; struct->int1 = argc;
;
                xor     ecx, 7
                add     esp, 4
                mov     [eax+struc_1.int2], ecx
;
; struct->int2 = argc ^ 7;
;
                xor     ecx, ecx
                lea     edx, [eax+struc_1.string]
;
; edx = struct->string;
;
                mov     edi, edi

loc_401020:                             ; CODE XREF: sub_401000+2A↓j
                lea     eax, [ecx+30h]
; for (i = 0 ; i <= 10 ; i++)  {
;  struct->string[i] = 0x30 + i;
; }
;
                mov     [edx+ecx], al
                inc     ecx
                cmp     ecx, 0Ah
                jle     short loc_401020
                push    edx
                push    offset aS       ; "%s"
                call    ds:printf
                add     esp, 8
                xor     eax, eax
                pop     ebp
```

```
struc_1         struc ; (sizeof=0x9)
int1            dd ?
int2            dd ?
string          db ?
struc_1         ends
```

# מערכים ומבנים

```
struct exampleStruct {
int x;
int y;
char array[10] ;
};
int main(int argc, char * argv[] ) {
        struct exampleStruct * e = malloc(sizeof(struct exampleStruct));
        int i = 0;
        e->x = argc;
        e->y = argc^7;
        for (i = 0 ; i <= 10 ; i++)  {
                e->array[i] = 0x30 + i;
        {
        printf("%s" , e->array);
        return 0;
{
```

מה הבעיה?

# מערכים ומבנים

```
struct exampleStruct {
int x;
int y;
char array[10] ;
};
int main(int argc, char * argv[] ) {
        struct exampleStruct * e = malloc(sizeof(struct exampleStruct));
        int i = 0;
        e->x = argc;
        e->y = argc^7;
        for (i = 0 ; i <= 10 ; i++)  {
                e->array[i] = 0x30 + i;
        {
        printf("%s" , e->array);
        return 0;
{
```

מה הבעיה?
מה דרסנו? מדוע התוכנית לא קורסת?

# מערכים ומבנים

- גודל ה-Struct צריך להיות 18 בתים:

```
struct exampleStruct {
        int x;
        int y;
        char array[10] ;
};
```

- בפועל הקצינו 20 בתים:

```
push    14h             ; size_t
call    ds:malloc
```

- יש פה עקרון של Alignment. ה-Struct מיושר לכפולות של 4 בתים.

# מחרוזות

**SCAS**
SCAS compares EAX with doubleword at [edi], and increments EDI;
*scas DWORD PTR [edi]; // or just scas*

**STOS**
STOS (store to string) moves a dword from eax to [edi], and increments EDI.
*stos DWORD PTR ES:[edi]*

**CMPS**
compares dword at address [esi] with dword at address [edi];
IF the direction flag is 0 - increments EDI and ESI.
If the direction flag is 1 (STD was executed), the registers decrement.
*cmps DWORD PTR DS:[esi], DWORD PTR ES:[edi]*

⋮

# מחרוזות

**REP/REPE/REPNE**

The string instructions may be prefixed by REP/REPE/REPNE which will repeat the instructions according to the following conditions:

| | |
|---|---|
| rep | decrement ecx ; repeat if ecx ≠ 0 |
| repe | decrement ecx ; repeat if ecx ≠ 0 AND zf = 1 |
| repz | decrement ecx ; repeat if ecx ≠ 0 AND zf = 1 |
| repne | decrement ecx ; repeat if ecx ≠ 0 AND zf = 0 |
| repnz | decrement ecx ; repeat if ecx ≠ 0 AND zf = 0 |

# מחרוזות

- מה המטרה של הקוד?

```
; EDI point to "hello world"
xor ecx,ecx
xor eax,eax
not ecx
repne scasb
not ecx
dec ecx

ECX = ?
```

```
ECX = strlen(EDI);
```

# סיכום

- ראינו היום
  - כיצד להשתמש ב IDA.
  - כיצד מבנים מסוימים נראים באסמבלי (ולהפך).
  - פקודות נוספות.
  - סביר להניח שיש שוני בין הקומפיילרים השונים אך העקרונות זהים.

- יש עוד המון מה ללמוד מהבחינה הזאת, קוד שהקומפיילר יוצר לא תמיד פשוט להבנה.

# CRACKMES

- תוכנית קטנה שנועדה לבחון יכולות הנדסה לאחור.
- בדרך כלל נבנית ע"י חוקרים אחרים - לכן חוקי לחקור אותה.
- מכילים מנגנונים זהים למנגנונים שקיימים בתוכנות מסחריות.

- דוגמא לאתרים:
  - Crackmes.de – מכיל מאגר שלם של תוכניות.
  - Tdhack.com – כנ"ל.
  - וכמובן http://webcourse.cs.technion.ac.il/236653/
    - תחת Assigment 2

- נפתור עכשיו crackme פשוט בתור חימום...