ארגון ותכנות המחשב – תרגיל בית 3

מתיאס בונה

הסתכלו בקובץ prog המצורף לתרגיל וענו על השאלות הבאות. מומלץ להיעזר בכלים עליהם למדנו בקורס (readelf, objdump וכו').

שימו לב: שני חלקי התרגיל מבוססים על אותו קובץ prog המצורף לתרגיל.

חלק א': Reverse Engineering

בחלק זה נסתכל על תוכנית מקומפלת וננסה להבין מה היא עושה.

1. כמה program headers מוגדרים בקובץ? 9
2. עבור כל program header מסוג LOAD כתבו את הנתונים הבאים: מיקום בקובץ (offset בבייטים), כתובת בזיכרון, גודל בקובץ, גודל בזיכרון והרשאות.
3. מהו ערך הבייט שנמצא בכתובת 0x40108d בתחילת ריצת התוכנית?
4. להלן הגדרה של משתנה שנמצא בכתובת 0x603050. השלימו את ערך האתחול החסר.

```
unsigned long foo = 0x_________________;
```

5. להלן הגדרת הפונקציה check_password. השלימו את החלקים החסרים. מותר להשתמש בהגדרת המשתנה foo מהסעיף הקודם.

```
int check_password(unsigned char *s)
{
        unsigned long x, y;

        if (*s == 'a')
                return 0;

        x = 0;
        while (y != 0) {
                y = *s - 'a';
                if (y > 25) 25.. its not an alphabetical character
                        return 0;

                if (x > *s) x>rax
                        return 0;
                x = 26*x+(*s);

                s++;
        }

        return (x==foo);
}
```

6. מהי הסיסמה הנכונה?

חלק ב': Binary Exploitation

בחלק זה ננצל באג בתוכנית כדי לגרום לה להריץ קוד לבחירתנו על המחשב של המשתמש. נשתמש בטכניקה נפוצה לניצול באגים מסוג זה שנקראת ROP.

להלן הגדרת הפונקציה main:

```
int main ()
{
        char user_password [16];

        printf ("Enter the password: ");
        scanf ("%s", user_password);

        if (check_password (user_password))
                printf ("Good! That's the password.\n");
        else
                printf ("Sorry, that's not the password.\n");

        return 0;
```

}

1. הסבירו בקצרה מה הבעיה בקריאה ל־scanf שמבצעת התוכנית.
2. משתמש הכניס את הקלט הבא:

   aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZ

   לאיזו כתובת תקפוץ פקודת ret שמבצעת main? (לפתרון הסעיף מומלץ להסתכל בקוד אסמבלי של main או להשתמש ב־gdb)
3. בכל שורה בטבלה הבאה מופיע קטע קוד קצר. עבור כל קטע קוד כיתבו:

   (1) את קידוד הפקודות לפי סדר הופעתן, ב־hex משמאל לימין.

   (2) כתובת בזיכרון שבו נמצא קידוד הפקודות, באזור בעל הרשאת execute.

   ראו דוגמה בשורה הראשונה. בסעיף זה מומלץ שלא להיעזר ב־objdump.

פקודות	קידוד	כתובת
`pop %rdi` `ret`	`5f c3`	0x401d13
`pop %rax` `ret`		
`syscall`		

`pop %rsi`
`pop %r15`
`ret`		

`push %rbp`
`mov $0x602e20, %edi`
`mov %rsp, %rbp`
`call *%rax`		

`add %r15, %rdi` `ret`		

4. תנו דוגמה לקלט שיגרום לתוכנית לצאת עם קוד יציאה 17 (דצימלי). לכתיבת ערכים בינאריים השתמשו בפורמט \xHH. לדוגמה, אם הקלט הוא האות a ואחריה בייט עם ערך 0x80 ואחריו בייט עם ערך 0x90, כיתבו: "a\x80\x90". בסעיף זה הדרישה היחידה היא שקוד היציאה של התוכנית יהיה 17; אין חשיבות לפלט שהתוכנית מדפיסה למסך לגבי נכונות הסיסמה.

5. תנו דוגמה לקלט שיגרום לתוכנית ליצור תיקייה בשם `my_first_exploit` עם הרשאות 0755 (אוקטלי) תחת התיקייה הנוכחית. הניחו שלא קיים קובץ או תיקייה בשם זה תחת התיקייה הנוכחית, ושיש הרשאות ליצור תיקייה כזו. בסעיף זה הדרישה היחידה היא שתיווצר תיקייה כפי שהוגדר; אין צורך לצאת מהתוכנית בצורה מסודרת לאחר יצירת התיקייה, ואין חשיבות לפלט שמודפס לגבי נכונות הסיסמה. בפרט, זה בסדר שלאחר יצירת התיקייה התוכנית תסתיים כתוצאה מ־segfault או מסיגנל אחר.