תכנות בטוח

תרגיל 2

מועד אחרון להגשה: 17.5.2020

מטרות התרגיל: buffer overflow, shell code, return oriented programming

משקל התרגיל 10 נקודות

הוראות הגשה:

- לגלוש למערכת ההגשה:

http://submit.cs.biu.ac.il

לבחור בקורס תכנות בטוח, תרגיל 2.

להעלות קובץ בשם **ID.txt** שמכיל את מספר תעודת הזהות של הסטודנט.

לאחר **כמה** דקות, יתקבל במייל קובץ **tar** עם קבצי התרגיל.
(המספר **n** בשם הקובץ הוא ארבעת הספרות הימניות של תעודת הזהות, אם **n < 1000** אז **n += 1000**)

- להעביר את הקובץ שהתקבל למערכת לינוקס ולפתוח באמצעות הפקודה:

```
tar -xvf target1234.tar
```

התיקייה שנפתחה מכילה שתי תכניות **ctarget** ו- **rtarget** שאותן יש לתקוף.

שתי התכניות קוראות קלט מ- **stdin** לתוך **buf** באמצעות הפונקציה:

```
unsigned getbuf()
{
char buf[BUFFER_SIZE];
Gets(buf);
return 1;
}
```

הפונקציה **Gets** קוראת עד לתו **\n** ולכן יש אפשרות ל- **buffer overflow**.

- כדי לתקוף, צריך להכין קובץ עם קוד תקיפה ולהכניס אותו כקלט לתכנית שרוצים לתקוף.

מאחר שקוד התקיפה מורכב מכתובות ופקודות מכונה שמכילות תווים שאינם **ascii**, אפשר להשתמש בתכנית **hex2raw** כדי ליצור תווים אלו.
התכנית **hex2raw** מקבלת כקלט תווים שהערך שלהם מיוצג על ידי שני תווים הקסדצימליים.
לדוגמה, עבור הקלט **a0 a1** התכנית תיצור שני תווים שהערך שלהם **160 161**.
מאחר שהמכונה היא **little endian** צריך להכין כתובות ומספרים בסדר הפוך.

לאחר שקובץ התקיפה מוכן, אפשר להשתמש בפקודה הבאה:

```
cat solution1.txt | ./hex2raw | ./ctarget
```

- אם הפתרון נכון, תקבלו:

```
Valid solution . . .
PASSED: Sent exploit string to server to be validated.
NICE JOB!
```

נא לגלוש לשרת הבדיקות:

http://submit.org.il:2020

כדי לבדוק את הציון הנוכחי.

הערות:

1. שרת הבדיקות עדיין לא בודק את הפתרונות.
מי שיוריד את קבצי התרגיל עד סוף השבוע (1.5), יוכל בשבוע הבא להפעיל שוב את הפתרון ולקבל ציון.
מי שיוריד את קבצי התרגיל בשבוע הבא, יוכל בשבוע שאחרי כן להפעיל שוב את הפתרון ולקבל ציון.

2. אם קבלתם:

```
Valid solution . . .
. . . You caused a segmentation fault!
```

הפתרון נכון, הבעיה במערכת.
נא להפעיל את הפתרון במחשב **u2**.

- כדי לקבל את מלוא הנקודות צריך לבצע חמש משימות:

ctarget

1. לדרוס את כתובת החזרה של הפונקציה **getbuf** עם כתובת הפונקציה **touch1**.

2. לדרוס את כתובת החזרה של הפונקציה **getbuf** עם כתובת **shellcode** שקורא לפונקציה **touch2** ומעביר לה כארגומנט את ה- **cookie** (שבקובץ **cookie.txt**) **כמספר**.

3. לדרוס את כתובת החזרה של הפונקציה **getbuf** עם כתובת **shellcode** שקורא לפונקציה **touch3** ומעביר לה כארגומנט את ה- **cookie** (שבקובץ **cookie.txt**) **כמחרוזת**.

rtarget

4. כמו ב- 2, לקרוא לפונקציה **touch2** ולהעביר לה את ה- **cookie** **כמספר**.
המחסנית אינה ניתנת לביצוע וצריך לקרוא לפונקציה באמצעות **ROP**.
ה- **gadgets** נמצאים בתוך **rtarget** בין **start_farm** ל- **mid_farm**.
(המקור נמצא בקובץ **farm.c**)

5. כמו ב- 3, לקרוא לפונקציה **touch3** ולהעביר לה את ה- **cookie** **כמחרוזת**.
המחסנית אינה ניתנת לביצוע וצריך לקרוא לפונקציה באמצעות **ROP**.
ה- **gadgets** נמצאים בתוך **rtarget** בין **start_farm** ל- **end_farm**.
(המקור נמצא בקובץ **farm.c**)

- כדי לברר את גודל ה- buffer, את כתובות הפונקציות והמחסנית אפשר להפעיל את הפקודות:

```
objdump -d ctarget

gdb ctarget
b getbuf
r
n
info reg
```

- התרגיל לקוח מתוך אתר הספר CSAPP, מצורף תיעוד התרגיל באנגלית ובו פרטים נוספים.

Good luck and have fun!