Rapport Exercices Sécurité des Applications : Série 2

Thomas Dagier

October, 4, 2020

1 Code Generator (RND1)

Nous avons vu en classe que la fonction `rand()` dépend de la fonction `srand()` qui elle-même dépend de la date précise à laquelle le programme a été lancé. Si on connait la date avec précision, on est capable de changer la graine génératrice du `srand()` et de ce fait, la fonction `rand()` n'est plus du tout aléatoire.

On sait que le programme générant les codes a été lancé le Vendredi 31 Juillet 2020 à 12:08:05. Il existe, en C, une structure `tm` (ligne 13) qui créée une base de temps que l'on peut utiliser dans le `srand()` (ligne 16). En appellant dans la fonction `srand()` une base de temps fictive, on peut reproduire exactement la même succession de `rand()` (ligne 21). On réutilise alors le code fournit que l'on modifie avec la bonne date :

```
1  // Nuclear Warhead code generator
2
3  #define NB_NUM 10
4  #define MIN_NUM 1
5  #define MAX_NUM 9
6
7  #include <stdio.h>
8  #include <stdlib.h>
9  #include <time.h>
10
11 int main(void){
12     puts("Nuclear Warhead code generator");
13     struct tm plop = {5,8,11,31,6,120};
14     time_t kek = mktime(&plop);
15     printf("%s", ctime(&kek));
16     srand(kek);
17     while(1) {
18         puts("Press Any Key generate new code");
19         getchar();
20         for(int i = 0; i < NB_NUM; i++){
21             printf("%d ",rand() % (MAX_NUM - MIN_NUM + 1) + MIN_NUM);
22         }
23         printf("\n");
24     }
25     return 0;
26 }
```

Figure 1: programme à executer pour trouver les codes

En éxécutant ce petit programme, on obtient la suite de code déjà générée plus les nouveaux :

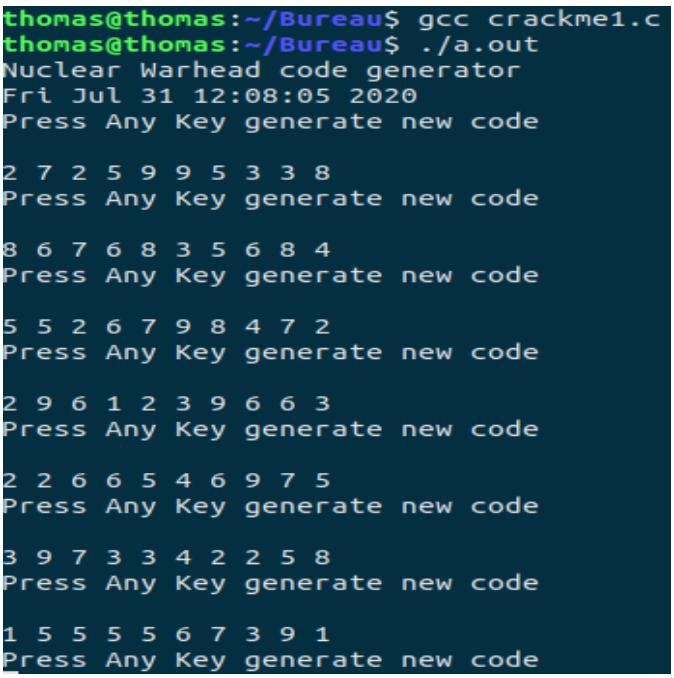

Figure 2: éxécution du programme pour trouver les codes suivants

2 Card Game (RND2)

Pour le second programme, la logique est exactement la même. Nous devons changer la date à laquelle le programme a été lancé pour reproduire exactement la même succession de `rand()`. La complexité, ici, réside dans le fait que nous ne connaissons pas la date mis à part que le programme a été lancé en 2020.

Il faut donc tester la fonction `srand()` avec toutes les dates possibles du 1er Janvier 2020 à 00:00:00 jusqu'à la date actuelle. Il faut changer le timestamp (soit la date qui est prise en compte par `srand()`) en l'incrémentant de 1 seconde à chaque fois (boucle `for()` lignes 17 et 18).

Nous connaissons la première suite de carte ce qui signifie que nous savons quel est le premier `rand()` de chaque `srand()`. Pour chaque seconde qui s'écoule (soit chaque `srand()`), on génère une suite de carte que l'on va comparer à la suite de carte que l'on sait être la bonne (boucle `for()` lignes 19 à 24 et condition ligne 25).

Si la suite est la même, on conserve le `srand()` et on génère les suites de cartes suivantes jusqu'a obtenir les combinaisons gagnantes du concours de cette année (ligne 26 à 42). Dans le cas contraire, on recommence avec un nouveau `srand()` et on continu jusqu'à trouver la bonne combinaison ou jusqu'à atteindre la date actuelle (ce qui annoncerait un problème dans le code). En éxécutant ce petit programme, on obtient la date à laquelle il a été officiellement lancé ainsi que les combinaisons suivantes :

```c
int main(){
    struct tm start = {0,0,0,1,0,120};
    time_t debut = mktime(&start);
    time_t end = time(NULL);

    const char* CardColors[] = {"♠","♥","♦","♣"};
    const char* CardValues[] = {"2","3","4","5","6","7","8","9","J","Q","K","A"};
    char value[100] = "A♦ 4♣ J♠ 7♦ A♠ 3♠ ";
    char tmp1[100];
    char tmp2[100];

    for (time_t i = debut; i < end; i++) {
        srand(i);
        for (int j = 0; j < 6; j++) {
            strcpy(tmp1, CardColors[rand()% 4]);
            strcat(tmp2, CardValues[rand()% 12]);
            strcat(tmp2, tmp1);
            strcat(tmp2, " ");
        }
        if(strcmp(tmp2,value)==0){
            printf("%s",ctime(&i));
            printf("%s\n", tmp2);
            strcpy(tmp1, "");
            strcpy(tmp2, "");
            for(int k = 0; k < 4; k++){
              for (int l = 0; l < 6; l++) {
                  strcpy(tmp1, CardColors[rand()% 4]);
                  strcat(tmp2, CardValues[rand()% 12]);
                  strcat(tmp2, tmp1);
                  strcat(tmp2, " ");
              }
              printf("%s\n", tmp2);
              strcpy(tmp1, "");
              strcpy(tmp2, "");
            }
            break;
        }
        strcpy(tmp1, "");
        strcpy(tmp2, "");
    }
    return 0;
}

```

Figure 3: éxécution du programme pour trouver les codes suivants

```
thomas@thomas:~/Bureau$ gcc crackme2.c
thomas@thomas:~/Bureau$ ./a.out
Fri Feb  7 03:00:00 2020
A♦ 4♣ J♠ 7♦ A♠ 3♠
K♥ 3♦ 9♦ 4♦ A♠ 4♣
2♥ Q♥ 2♦ 2♥ J♠ 6♣
Q♠ 2♦ K♠ 8♥ J♠ K♣
Q♣ J♠ 9♠ Q♦ 9♠ K♣
```

Figure 4: éxécution du programme pour trouver les combinaisons suivantes