FCSC 2024 - Write-up: Array d’urgence

Nom du challenge : Array durgence
Categorie : Misc / Algo
Points : 378
Auteur : Morad Halmi (Junior Pentester)
Flag obtenu : FCSC{b3a6e5e45a7efcbefe1757d689ad20c1b1ff886eced11b87b74bb25175efad05}

Objectif du challenge

On nous donne un acces distant a un service via nc chall.fcsc.fr 2050.
Le serveur genere a chaque etape un tableau de nombres entiers pseudo-aleatoires a laide dun seed, et nous der

Interaction avec le serveur

(seed, n, N) = (2529334158, 10, 4)
>>> i =
>>> j =

A ce moment-la, le serveur attend quon lui envoie deux indices i et j correspondant a la sous-liste de A qui donne l

Mais le tableau A nest pas affiche, seulement les parametres pour le reconstituer :
- seed : graine aleatoire utilisee
- n : intervalle de generation [-n, n)
- N : la taille du tableau sera 2 ** N

Analyse manuelle

En local, j'ai remarque que le tableau est genere ainsi :
random.seed(seed)
A = [random.randrange(-n, n) for _ in range(2 ** N)]

Donc a partir du triplet (seed, n, N), je peux reconstruire exactement le meme tableau en local si jutilise la meme lo

Ensuite, il faut resoudre un probleme classique dalgorithmique :
Trouver la sous-liste avec la somme maximale.

Ce probleme est bien connu, et jai utilise lalgorithme de Kadane, qui permet de le resoudre en complexite O(n).

Script local pour tester la methode

Avant d'automatiser la connexion, jai dabord teste ca en local :

```
def solve(A):
    max_sum = float('-inf')
    current_sum = 0
    start = 0
    max_start = 0
    max_end = 0

    for i, val in enumerate(A):
        if current_sum <= 0:
            current_sum = val
```

```
            start = i
        else:
            current_sum += val

        if current_sum > max_sum:
            max_sum = current_sum
            max_start = start
            max_end = i + 1

    return max_sum, max_start, max_end
```

Execution manuelle (simulation)

A chaque execution, j'ai fait les etapes suivantes a la main :

1. Me connecter au serveur avec :
   nc chall.fcsc.fr 2050
2. Copier le seed, n et N
3. Generer le tableau A localement avec Python
4. Utiliser mon algo pour trouver les bons indices i et j
5. Revenir dans le terminal, taper les valeurs

Exemple :

(seed, n, N) = (2529334158, 10, 4)

>>> i = 4

>>> j = 11

Resultat final

Jai repete ce processus pour toutes les series de tableaux generees par le challenge.

A la toute fin, jai recu ce message :

FCSC{b3a6e5e45a7efcbefe1757d689ad20c1b1ff886eced11b87b74bb25175efad05}

Ce que jai appris

- Comment reverse un algorithme aleatoire a partir dun seed
- Implementer l'algo de Kadane
- L'importance de comprendre le fonctionnement interne dun challenge meme si le contenu nest pas directement a
- Automatiser une resolution (meme si ici, jai fait les tests a la main pour mieux comprendre)

Bonus

Fichiers utilises :

- solution.py pour resoudre avec Kadane
- array-d-urgence.py script du challenge local
- Terminal avec nc pour faire les tests

Flag

FCSC{b3a6e5e45a7efcbefe1757d689ad20c1b1ff886eced11b87b74bb25175efad05}