Hepia

Haute École du Paysage, d'Ingénierie et d'Architecture

Ingénierie et Systèmes de Communication

Année académique 2020/2021

Hacking et pentesting

Série 3 – Librairies et mémoire

Genève, 28 Mars 2021

Étudiant :

Sergio Guarino

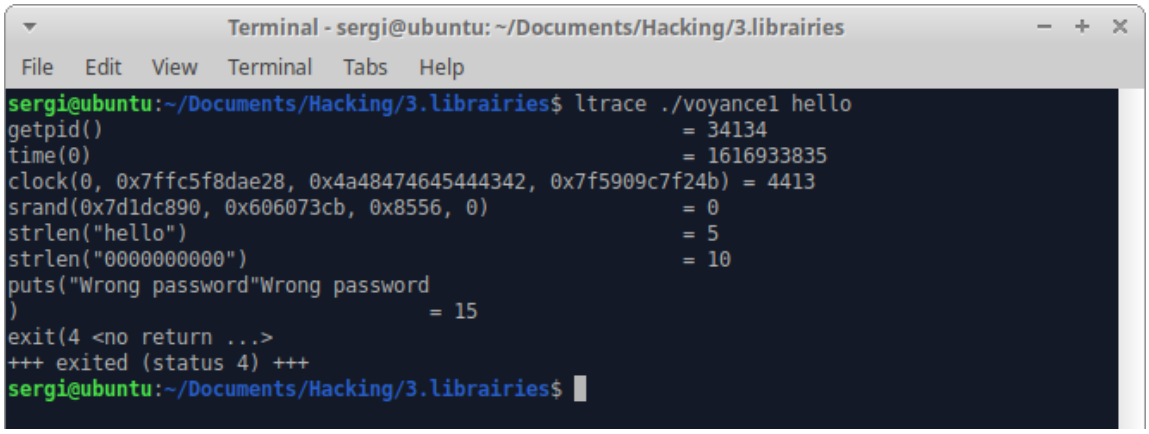
Professeur :

Stéphane Küng

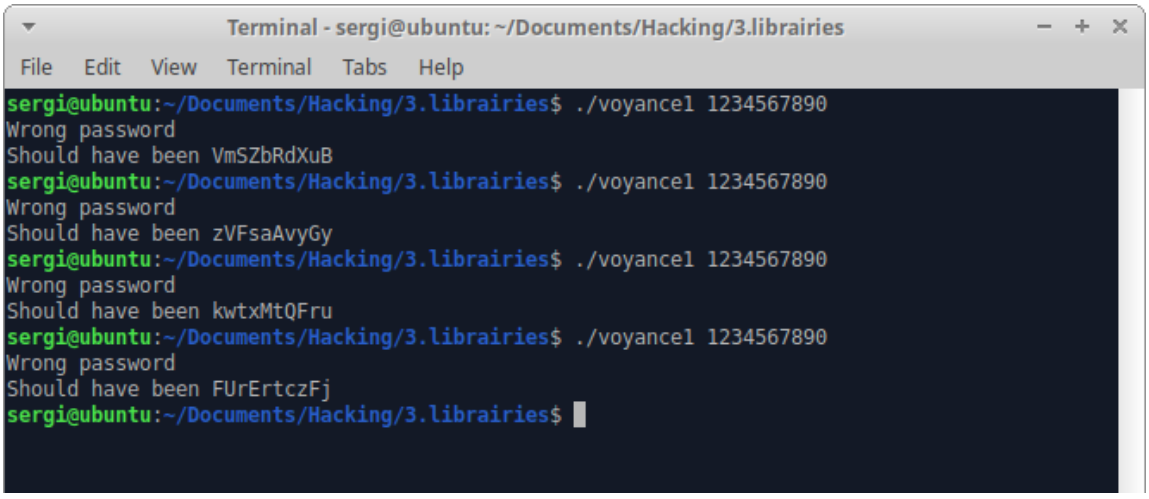
1. Voyance1

Pour le premier exercice on doit prévoir le mot de passe généré par le programme.

Il y a plusieurs solutions. On peut commencer par analyser le programme avec *ltrace* :



On remarque que la longueur du mot de passe doit être de 10 caractères. On essaye :



Le résultat est différent à chaque fois. Ceci était prévisible parce que avec *ltrace* on remarque l'appel à *srand*, ce qui implique un appel à *rand*.

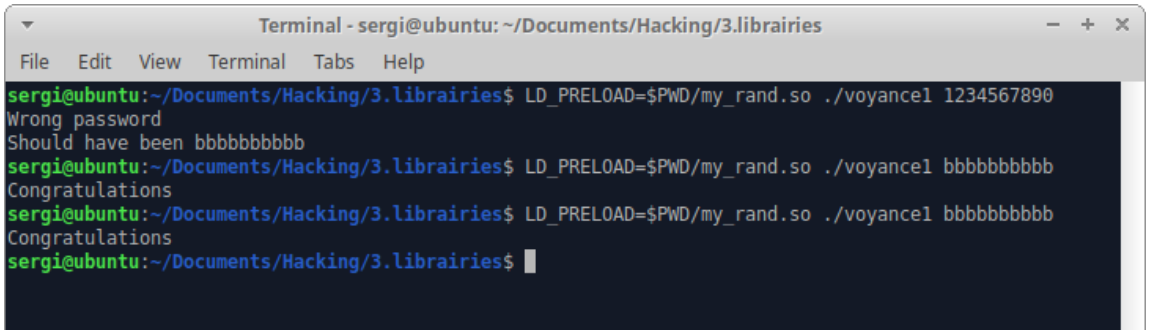
On peut donc modifier ce que la fonction *rand* retourne. On écrit un programme en C avec la ligne suivante :

int rand () { return 1 ; }

Ce programme est ensuite compilé en tant que librairie avec *gcc* :

gcc -shared -fPIC mon_programme.c -o mon_random.so

Enfin, on utilise *LD_PRELOAD* pour charger la librairie que l'on vient de créer à l'appel du programme voyance1 :



On a donc réussi à hacker le programme voyance1.

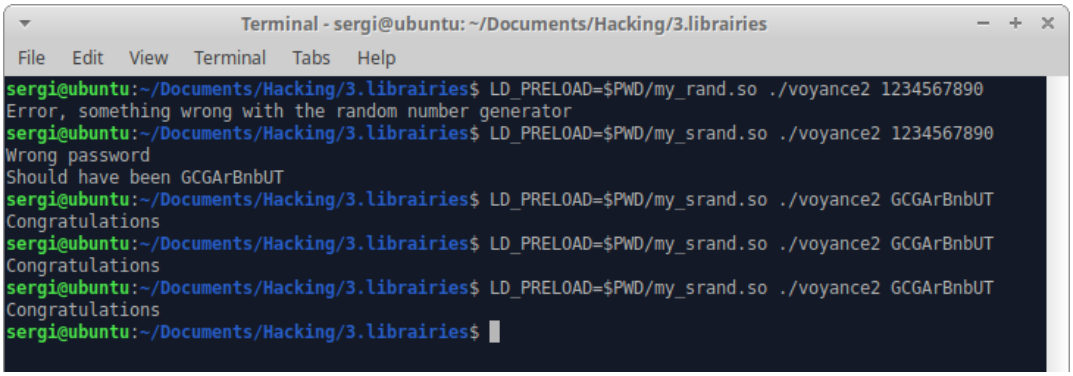
2. Voyance2

Le deuxième exercice est légèrement plus complexe, car il y a des contrôles supplémentaires qui ont été introduits dans le programme.

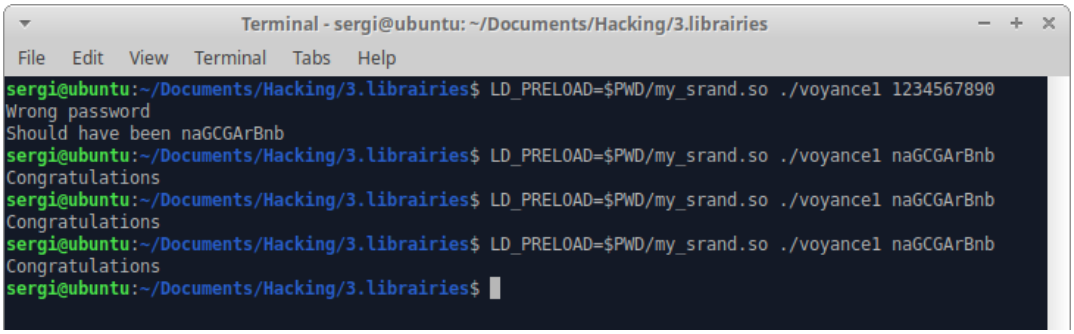
Si on lance le programme avec *LD_PRELOAD* et la librairie avec *rand* modifié, le programme retourne un message d'erreur. On peut alors modifier la fonction *srand*, avec un programme similaire à celui d'avant :

int srand () { return 100 ; }

Si on compile ce nouveau programme avec *gcc* comme avant et on utilise *LD_PRELOAD*, on arrive toujours à deviner le mot de passé :



C'est intéressant remarquer que cette librairie fonctionne également avec voyance1 :



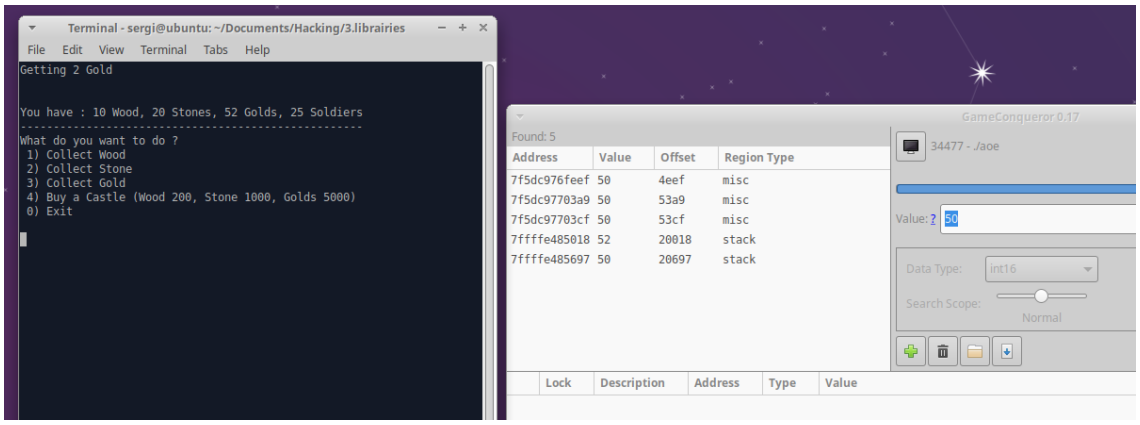
3. Age of Empire

Le dernier exercice demande de modifier la mémoire du programme pendant son exécution. Pour cet exercice on pourrait utiliser quelque chose comme *gdb*, mais on va se servir d'un éditer de mémoire. L'éditeur choisi pour cet exercice est *GameConqueror*.

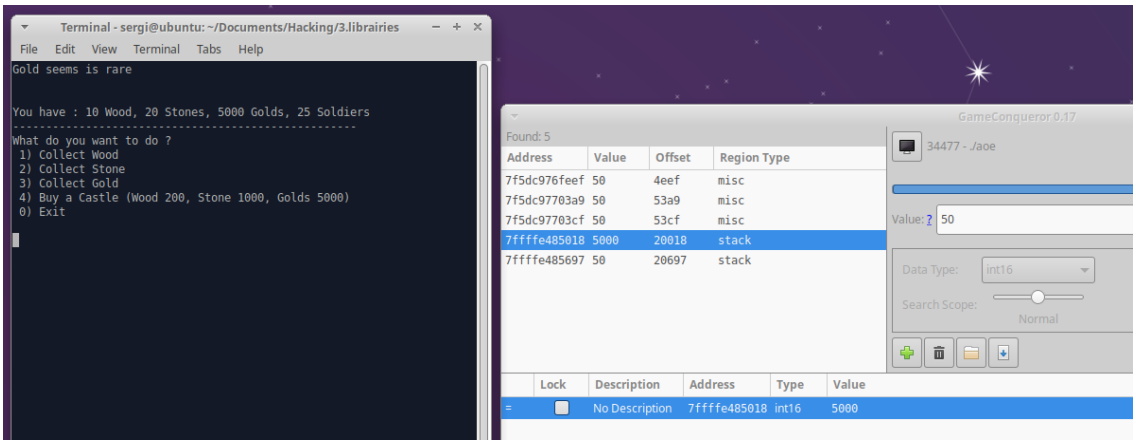
On ouvre un terminal et on lance le jeu Age of Empire. Au même temps on ouvre GameConqueror et on choisit le programme pour lequel on souhaite éditer la mémoire. Pour pouvoir réussir le jeu, il faut acheter un château, mais en n'ayant pas assez de ressources, c'est leur valeur que l'on va essayer de modifier. C'est certainement plus simple que d'essayer d'obtenir le château avec moins de ressources que celles demandées.

Grâce à GameConqueror, on peut effectuer une recherche dans les registres du jeu pour trouver les adresses des variables des nos ressources. Pour ce faire, on rentre la valeur actuelle d'une des ressources, puis on joue pour essayer d'incrémenter cette valeur : on verra la valeur s'incrémenter également sur GameConqueror.

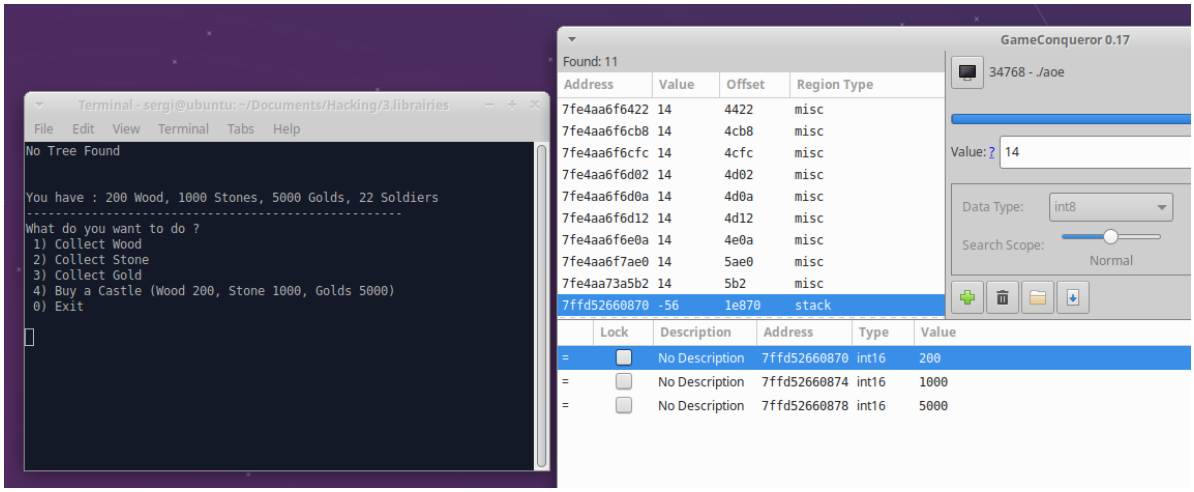
Dans la capture ci-dessous on peut voir que l'on a fait une recherche d'un **int16** avec valeur **50** et que en jouant on a incrémenté cette valeur à **52**. La modification s'est reflétée sur GameConqueror et on a pu trouver l'adresse de la variable contenante l'or.



On peut éditer cette variable et entrer la valeur que l'on souhaite. Dans ce cas on a rentré **5000** (qui est la quantité nécessaire pour acheter le château) et on peut voir que dans le jeu on a également 5000.



Pour les autres variables c'est la même procédure. Il y a une seule différence : elles sont des **int8** et non pas des **int16** et il a fallu changer leur type pour pouvoir dépasser la valeur maximale des entiers à 8 bits (127). Toutes les variables ont été paramétrées en tant que **int16**.



Une fois avoir récolté toutes les ressources nécessaires, on peut acheter notre château :