Rapport Exercices Sécurité des Applications : Série 3

Thomas Dagier

October, 10, 2020

1 Voyance 1 (LDPR1)

Le but de cet exercice est de deviner le mot de passe qui est choisit aléatoirement. Pour cela, on doit utiliser la notion de LD_PRELOAD qui permet de faire des appels vers des librairies distantes au moment de l'éxécution du binaire.

La première chose que j'ai fait est d'ouvrir le binaire avec ghidra pour tenter de comprendre le cheminement jusqu'au moment où le mot de passe s'affiche. J'ai alors remarqué deux fonctions appelées par des libaires externes : `strlen()` et `rand()`. En regardant un peu l'énoncé des deux exercices, j'ai remarqué que `rand()` ne devait pas être la bonne option puisque un morceau de code à été rajouté dans le second binaire pour contrer la fonction `rand()` si elle est modifiée par notre librairie externe. J'ai donc fait le choix de modifier la fonction `strlen()`.

```
srand(__seed);
sVar5 = strlen((char *)puParm2[1]);
sVar6 = strlen((char *)&local_63);
if (sVar5 != sVar6) {
  puts("Wrong password");
                    /* WARNING: Subroutine does not return */
  exit(4);
}
local_78 = 0;
while( true ) {
  sVar5 = strlen((char *)&local_63);
  if (sVar5 <= (ulong)(long)local_78) break;
  iVar2 = rand();
  sVar5 = strlen((char *)&local_58);
  *(undefined *)((long)&local_63 + (long)local_78) =
       *(undefined *)
        ((long)&local_58 +
        SUB168((ZEXT816(0) << 0x40 | ZEXT816((ulong)(long)iVar2)) % ZEXT816(sVar5),0));
  local_78 = local_78 + 1;
}
local_74 = 0;
while( true ) {
  sVar5 = strlen((char *)&local_63);
  if (sVar5 <= (ulong)(long)local_74) {
    puts("Congratulations");
                    /* WARNING: Subroutine does not return */
    exit(0);
  }
```

Figure 1: binaire de voyance 1 avec ghidra

On voit bien que la fonction `strlen()` est très souvent appelée donc cela parait interessant de modifier cette fonction.
La première étape est donc de créer un fichier que l'on ferra passer pour une librairie externe. Ce fichier se nomme `strlen-hijack.c`

```c
int strlen(const char *s1) {
    return 0;
    }
```

Figure 2: fonction `strlen()` modifiée dans le fichier `strlen-hijack.c`

Une fois le fichier créé et la fonction `strlen()` modifiée, on va compiler la nouvelle "fausse librairie" en tant que librairie partagée avec les commandes :
`gcc -fPIC -c strlen-hijack.c -o strlen-hijack.o`
puis :
`gcc -shared -o strlen-hijack.so strlen-hijack.o`.

Dans un second temps, on peut essayer d'éxécuter notre binaire avec LD_PRELOAD en utilisant la commande : `LD_PRELOAD="./strlen-hijack.so" ./voyance1 password`. Avec `strlen-hijack.so` notre librairie partagée à utiliser avec le binaire que l'on execute ainsi : `./voyance1 password`

```
thomas@thomas:~/Documents/GIT/securite_applications/Serie3$ LD_PRELOAD="./strlen-hijack.so" ./voyance1 password
Congratulations
```

Figure 3: éxécution du binaire avec LD_PRELOAD pour voyance 1

Le fait de modifier la fonction `strlen()` est très pratique dans notre cas. En effet, comme les valeur de retour de cette fonction sont souvent comparées, toutes les conditions sont vérifiées. En effet on compart 0 et 0 donc cela marche toujours.

Remarque : Comme la fonction `strlen()` retourne toujours 0, le binaire nous annonce que le mot de passe est le bon peut importe celui que l'on rentre ce qui n'aurais pas été le cas si on avait modifié la fonction `rand()`. Si `rand()` retournait toujours la même valeur une fois modifié, le mot de passe aurait été une chaine de caractère de longueur à determiner contenant toujours le même caractère (exemple : bbbbbbbbb).

2 voyance 2 (LDPR2)

Dans le second exercice, le code est sensiblement le même à la différence qu'une condition vérifie que la fonction `rand()` ne retourne pas toujours la même valeur. Dans le cas contraire, le programme se coupe avec un message d'erreur. Le fait d'avoir modifié la fonction `strlen()` et non pas la fonction `rand()` permet de réutiliser exactement la même librairie pour deviner le mot de passe du second binaire. Ceci n'aurait pas été possible dans le cas où le binaire de voyance 2 aurait changé au point de ne plus dépendre de la fonction `strlen()`.

On refait donc exactement la même manipulation que pour le premier exercice :
`gcc -fPIC -c strlen-hijack.c -o strlen-hijack.o`
`gcc -shared -o strlen-hijack.so strlen-hijack.o`.
`LD_PRELOAD="./strlen-hijack.so" ./voyance1 password`.

```
thomas@thomas:~/Documents/GIT/securite_applications/Serie3$ LD_PRELOAD="./strlen-hijack.so" ./voyance2 password
Congratulations
```

Figure 4: éxécution du binaire avec LD_PRELOAD pour voyance 2

On observe que le LD_PRELOAD fonctionne tout aussi bien pour voyance 2 avec la fausse librairie partagée.