EVL - TP3 : obfuscation de programmes

Sandrine Blazy - ISTIC

12 octobre 2018

Le but de ce TP est d'écrire des obfuscations élémentaires de programmes. Pour chaque obfuscation, vous testerez le programme obfusqué afin de vos assurer qu'il se comporte comme le programme initial dont il est issu.

Les programmes que vous aurez obfusqués devront faire partie de votre compte-rendu de TP.

1 Renommage de variables

Modifier le programme `obf1.c` en renommant toutes ses variables. Que constatez-vous dans le code assembleur généré ?

2 Découpage de variables

1. Obfusquer le programme `obf2.c` en remplaçant toute occurence de variable par deux variables. Plus précisément, une variable `x` sera remplacée par deux variables `a` et `b` telles que `a = x /5` (i.e. division entière) et `b = x % 5` (i.e. reste de la division entière).
2. En quoi les codes assembleur générés pour les deux programmes (i.e. le programme initial et le programme obfusqué) sont-ils différents ?
3. Même question en utilisant le niveau maximal d'optimisation de `gcc`.
4. Obfusquer à nouveau le programme précédemment obtenu en modifiant lorsque cela est possible l'ordre d'écriture des instructions.
5. Rajouter dans le programme un prédicat opaque introduisant du code mort dans le programme.

3 Encodage d'entiers

1. Obfusquer le programme `obf3.c` en encodant ses constantes entières de la façon suivante : chaque constante `c` devient `c*6`. Le programme obfusqué se comporte-t-il comme le programme initial ?
2. Comment faut-il modifier le programme obtenu à la question précédente pour qu'il soit équivalent au programme initial ?
3. Obfusquer le programme `obf4.c` en encodant ses constantes entières en les multipliant par 6, de façon à obtenir un programme équivalent. Cette obfuscation utilisera les fonctions `encode` et `decode` fournies dans `obf4.c`. Comment peut-on s'assurer que le programme obfusqué se comporte comme le programme initial ?
4. Reprendre la dernière question en encodant les constantes entières du programme de la façon suivante : chaque constante `c` devient `c^6`, `^` désignant l'opérateur XOR du C.

4 Expressions mixtes arithmético-booléennes

1. Le programme `obf5.c` teste l'obfuscation vue en cours par expressions mixtes arithmético-booléennes, qui obfusque `r = x ^ 92;` en la suite d'instructions suivantes.

```
a = 229*x + 247 ;
b = 237*a + 214 + ((38*a+85) & 254) ;
c = (b + ((-2*b + 255) & 254))*3 + 77 ;
d = ((86*c + 36) & 70) * 75 + 231*c + 118 ;
e = ((58*d + 175) & 244) + 99*d + 46 ;
f = e & 148 ;
g = (f-(e&255) + f)*103 + 13 ;
r = 237*(45*g + (174*g | 34)*229 + 194 - 247) & 255 ;
```

2. Compiler ce programme sans utiliser les optimisations de `gcc`. Le code assembleur généré est-il également obfusqué ?
3. Compiler ce programme en utilisant le niveau maximal d'optimisation de `gcc`. Quels sont les effets des optimisations ? Le code assembleur généré est-il également obfusqué ?