Cahier des charges

Code obfuscator

Semestre d’hiver 2010/11

Professeurs : Schuler Jean-Roland, Kilchoer François  
Etudiants : Franzi Robin, Derbel Amine

# Généralités

## Contexte

Dans le cadre d'un projet de recherche, le groupe de compétences IT Security veut rendre difficile le reverse-engineering d'un programme executable.   
En effet des utilitaires comme objdump ou IDAPro permettent facilement de trouver le code assembleur et de là retrouver des algorithmes industriels qui sont peut-être des secrets de fabrications d'une entreprise.

## Objectifs du projet

Pour faire ceci, il faudra:

* Interpréter le code machine d'un programme
* Créer un obfuscateur de code qui permet de changer la structure du programme initial pour le rendre inintelligible sans changer sa sémantique

# Développement du projet

## Avancement du projet dans le temps :

1. Recherche d’information

* structures des executables et codage elf
* techniques d'obfuscation en assembleur
* langage assembleur ARM

1. Développement d'un obfuscator simplifié

* proposition de solutions
* essais sur des boucles : if, while
* tests

1. Développement du programme final

* ajout de fonctionnalités
* évaluation des performances
* tests

1. Rendu final

## Planning du projet et répartition des taches



Travail en commun

Travail effectué par Amine

Travail effectué par Robin

# Jalons

Les jalons représentent une étape importante dans le développement du projet, nous avons choisi d’en fixer deux comme montré sur le planning, ils nous aideront à nous repérer dans le temps et de bien atteindre les buts que nous avons fixés.

Jalon 1 : fixé à la semaine A11, nous présenteront un prototype fonctionnel d’un obfuscateur simple que nous pouvons opérer sur des instructions if et while.

Jalon 2 : fixé à la semaine A14, nous présenteront le programme final deux semaines avant le rendu final pour vérifier les dernières modifications à rajouter.

# Cahier des charges

* Comprendre le langage assembleur des processeurs ARM
* Comprendre la structure elf
* Essayer le reverse-engineering : obtenir le code assembleur à partir d’un exécutable
* Maitriser quelques techniques d’obfuscation
* Ecrire un programme qui permet d’obfusquer l’exécutable d’un programme simple
* Ecrire un programme qui permet d’obfusquer le code assembleur d’un programme simple
* Comparaison des deux méthodes