Introduction au Reverse Engineering - 1/4

Cet article est consacré au reverse engineering (l'ingenierie inverse) c'est à dire à l'art du cracking. Qui est la forme du reverse la plus connu. Cet article est adapté plutot pour les débutants...

Introduction :

En général, on définit le reverse engineering par le fait de décompiler un programme, c'est à dire que l'on traduit
Un langage compréhensible par une machine en un langage lisible et compréhensible par un être humain.

Dans la pratique, le reverse engineering consiste à analyser le fonctionnement d'un programme ou d'un périphérique, ou
Alors en examinant pas à pas les résultats de son exécution.

Les droits du Reverse Engineering :

Le droit d'analyse :

Un nouveau droit d'analyser le logiciel est prévu par la loi de 94. Il est défini comme celui d'observer, d'étudier ou de tester le fonctionnement de ce logiciel lorsque l'utilisateur effectue toute opération de chargement, d'affichage, d'exécution, de transmission ou de stockage du logiciel.

C'est donc un droit au reverse engineering, défini comme l'analyse d'un système destinée à rechercher ses principes de conception. L'analyse n'est autorisée, tout comme la décompilation, que si elle ne cause pas de préjudice injustifié aux intérêts légitimes de l'auteur, et qu'elle ne porte pas atteinte à l'exploitation normale du programme, conformément au droit international.

Enfin, la loi prévoit que toute stipulation contractuelle contraire à ce droit est nulle et non avenue. Mais le contrat pourra fixer le cadre de l'utilisation normale du logiciel.

Le droit de décompilation :

Le droit de décompiler le logiciel, c'est à dire de tenter de retrouver le programme source à partir du programme objet, est ajouté à la condition qu'il soit effectué à des fins d'interopérabilité. La décompilation doit donc servir à interfacer le logiciel avec des logiciels de coordination, c'est à dire permettre l'articulation des logiciels entre eux. Elle doit se limiter aux parties nécessaires à l'interfaçage. En effet, cette décompilation suscite des inquiétudes notables quant à la protection du programme.

Si la loi interdit toute stipulation contraire à ce droit l'auteur pourra prévoir, contractuellement, de l'organiser, en prévoyant une procédure d'information à son égard, par l'utilisateur, de son désir de décompiler, suivi d'une communication des interfaces utiles et des points d'ancrage.

Puisqu'elle n'est autorisée que dans le cas où les informations nécessaires n'ont pas déjà été rendues facilement et rapidement accessibles, elle ne pourra être considérée comme indispensable au sens de la loi dans le cas d'une standardisation du logiciel, d'une publication ou d'un réponse à une demande individuelle.

A quoi sert le reverse engineering :

Introduction au Reverse Engineering - 2/4

L'homme est toujours très curieux, il veut toujours tous savoir. Le reverse engineering par de cette philosophie. L'être humain lorsqu'il a un logiciel entre les mains, il veut tous savoir de lui. Grâce au reverse engineering, il va pouvoir comprendre comment marche le logiciel.

Le reverse engineering fait progresser dans la programmation. Grâce au reverse engineering, on comprendra mieux comment marche notre machine, comment marche les logiciels que vous utilisez à longueur de temps. Le reverse enseignera une grande quantité d'informations sur la programmation.

Le reverse engineering a bien d'autre utilité que l'enseignement de nouvelles connaissances. On se sert beaucoup du reverse engineering pour ne pas payer les logiciels. On appel cette technique le cracking. C'est-à-dire que l'on déplombe un logiciel pour trouver un moyen de contourner les protections misent en place par les sociétés pour que l'on paye leur logiciel. Les protections peuvent aller du Sérial (code fournit à l'achat), ou Nag-Screnn (c'est une fenêtre qui s'affiche pour nous rappeler qu'il faut acheter le logiciel).

On peut se servir aussi du reverse engineering pour lire des formats de fichiers propriétaires non documentés (on peut cité le format MS-Word), tout comme pour communiquer via un protocole secret avec un serveur propriétaire (le logiciel Samba qui sert à partager des fichiers entre différentes architectures via le protocole SMB est basé entièrement sur le reverse engineering).

En utilisant des techniques de reverse engineering, on peut comprendre la structure d'un tel document et de le convertir en un format ouvert. Grâce à cela on peut garder ses documents même une fois les systèmes utilisés obsolètes.

De nombreux fabricants de matériel refusent de diffuser les spécifications de leur matériel. Les développeurs de logiciel doivent donc utiliser le reverse engineering pour utiliser ce matériel. On peut citer les cartes graphiques, les BIOS d'IBM...

Les Bases du reverse engineering :

Et oui lorsque l'on décompile un programme, on n'a pas un langage comme le C, ou le Visual Basic. C'est un peu plus dur que cela. Lorsque l'on décompile un programme, le langage qui apparaît est l'ASM, c'est-à-dire l'assembleur.

Le langage assembleur est très proche du langage machine (langage binaire 0 et 1). Il dépend donc du type de processeur. Ainsi il n'existe pas un langage assembleur, mais un langage assembleur par type de processeur. Mais je ne suis pas là pour vous faire un cour sur le langage assembleur. Il y a déjà un grand nombre de site qui l'on déjà fait, alors faite une petite recherche sur internet. Pour le reverse engineering, on se sert beaucoup des sauts conditionnels.

JNE Saute si pas égal 75 ou 85
JE Saute si égal 74 ou 84
JMP Saute EB ou E9
JA Saute si au dessus 0F87
JNA Saute si pas au dessus 0F86
JAE Saute si au dessus ou égal 0F83
JNAE Saute si pas au dessus ou égal 0F82

Introduction au Reverse Engineering - 3/4

JB Saute si en dessous 0F82
JNB Saute si pas en dessous 0F83
JBE Saute si en dessous ou égal 0F86
JNBE Saute si pas en dessous ou égal 0F87
JG Saute si plus grand 0F8F
JNG Saute si pas plus grand 0F8E
JGE Saute si plus grand ou égal 0F8D
JNGE Saute si pas plus grand ou égal 0F8C
JL Saute si moins grand 0F8C
JLE Saute si moins grand ou égal 0F8D
JNLE Saute si pas moins grand ou égal 0F8E
NOP Ne fais rien 90 ou 908F

Petit tableau des sauts assembleurs.

Les Outils :

Le meilleur désassembleur est sans équivoque Windasm, qui tourne sur Windows.
Il vous faudra aussi un éditeur hexadécimal, alors là je ne peux pas citer de nom vu le nombre qu'il existe à vous de faire votre choix.
Ensuite il vous faudra un debugger, le meilleur c'est Soft Ice. Mais il existe aussi OLLYDBG.
On peut citer plein d'autre programme très utile dans le reversing comme Procdump (détecter lorsqu'un logiciel est compresser), SmartCheck...

Et surtout ce qu'il faut pour réussir dans le reverse engineering, c'est ne pas avoir peur de rester longtemps, très longtemps devant votre ordinateur, pour essayer de comprendre un logiciel.

Dans un autre article, on étudiera plus en profondeur le reverse engineering. Je pense que l'on étudiera le Java reverse engineering qui lorsque l'on a les bons outils reste assez simple.

Exemple de reverse engineering :

On va prendre un programme tout basique codé en C par TheShade. Voilà à quoi ressemble notre programme, il a l'air tout simple.

Ce programme demande simplement un password. Lorsque notre password est faux, il y a "Perdu ! " qui s'affiche. Si notre password est bon, on a : "Bravo ! " qui s'affiche.
Nous allons donc ouvrir notre programme avec W32Dasm qui est l'un des désassembleurs les plus puissants. Et nous allons faire une recherche du mot : "Perdu". Pour cela vous irez dans : Search> Find Text.

Après avoir fait la recherche, nous allons arriver ici :

Introduction au Reverse Engineering - 4/4

Ici on peut voir quelque chose de très intéressant. On voit qu'il y a un saut de l'adresse 004013DA qui arrive ici, c'est à dire que "Perdu ! " apparaîtra. Nous allons donc aller à l'adresse 004013DA pour voir à quoi ressemble ce saut.

On voit à l'adresse 004013DA que le saut est un JNE c'est à dire que si pas égale et bien au saut à l'adresse 00401413 (c'est à dire que l'on saut à l'affichage de "Perdu ! "). Maintenant ce qui nous reste à faire c'est de changer le saut JNE et JE. On va changer le 7537 en 7437 grâce à un éditeur hexadécimal. Pour cela on va regarder avec W32Dasm à quel offset se trouve notre saut à changer. Vous allez donc regarder tout en bas de W32Dasm. Voilà ce que vous devez voir :

Ce qui nous intéresse nous c'est l'offset, ici c'est : 000007DAh. On peut donc retirer les 5 zéro et le h. Notre 75 à changer se trouve donc à l'offset 7DA. On changera donc le 75 en 74. Grâce à ce changement, on pourra rentrer n'importe quel password, "Bravo ! " s'affichera toujours.

Voilà un petit exemple de reverse engineering, très basique.

Crédits :
* Introduction to Reverse Engineering Software
* The Millenium Cracking Tutorial by Acid Burn