TP CYBER 2 - 14/10/2025

Les outils qui seront vu pendant le TP sont :

- objdump
- jwasm / nasm sous linux (x86)
- gcc et versions croisées
- gdb
- hexdump

L'objectif est de se familiariser avec l'assembleur et les notions d'ABI/architectures/format.

Exercice 1

En utilisant les outils gcc et objdump, trouver comment refaire l'exemple du cours pour l'architecture x86-64 linux.

Comment faire pour les autres architectures ? Qu'est-ce que c'est que la compilation croisée ?

Expliquer le concept de code natif et code managé.

Exercice 2

Télécharger les binaires et observer leurs structures avec les outils objdump / file :

Exécuter la commande :

File <binaire> puis interprétez le résultat pour tous les binaires.

Exécuter la commande :

Objdump –d –M intel <binaire> en remplaçant <binaire> par les différents exécutables. Que peut-on observer sur les registres et l'ABI utilisée ?

Exécuter la commande :

Objdump –h <binaire> en remplaçant binaire par les différents exécutables. Utiliser le manuel pour savoir ce que fait la commande. Interprétez.

Exercice 3

Télécharger le squelette ex3.asm à partir de moodle. Nous allons utiliser les outils jwasm / gcc / objdump dans cet exercice.

Compiler le fichier avec les commandes suivantes :

- Jwasm –elf64 ex3.asm && gcc –nostdlib ex3.o
- Objdump –d –M intel a.out

Expliquer ce que font les instructions push / pop / mov rax, [rsp+10] / cmp et noter leurs opcodes.

Expliquer les sauts conditionnels ja / jne / je / jg

Quel est l'opcode généré pour un jg ? et pour un ja ?

Expliquer la ligne :

104d: eb 00 jmp 104f <_start+0x4f>

Que se passe t'il si on remplace le eb 00 par eb fe ? Expliquer.

Exercice 4

Faire l'exemple du cours slide 42 en utilisant gcc. Expliquer

Ecrire un petit programme pour mettre en évidence la taille des types int / long / pointeur sous linux et windows.

Exercice 5: crackme du cours avec objdump, première approche vers RE statique

Observer avec objdump et bvi (pour modifier le binaire) le fonctionnement du programme et répondez aux questions suivantes :

- Quelles sont les pistes qu'on peut suivre pour comprendre le fonctionnement de ce programme ?
- Observer le fonctionnement des sauts conditionnels
- Observer les appels aux sous-fonctions et expliquer le passage de paramètre
- Investiguez pour trouver quels sont les paramètres de la fonction strncmp juste en regardant l'assembleur.
- Modifier les sauts conditionnels pour valider le crackme
- Retrouver le mot de passe recherché sans exécuter le binaire.

Exercice 6 : Première approche de RE dynamique

Dans cet exercice, on va utiliser un outil appelé gdb. Il s'agit d'un programme très connu qui permet de debugger des exécutables. Pour une première approche, on va reprendre le squelette de l'exercice 3 et exécuter les commandes suivantes :

Gdb ./a.out

(gdb) set disassembly-flavor intel

(gdb) run

(gdb) disp /i $rip

(gdb) x/32i $rip

(gdb) si

(gdb) i r

Après avoir vu l'effet des commandes, expliquer ce qu'elles font. En vous aidant du tableau sur les sauts conditionnels du cours, expliquer les sauts à l'aide des flags du registres eflags.

Enfin, reprendre le crackme de l'exercice 5 et lancer le avec gdb : ./gdb crackme. Naviguer dans le programme afin de visualiser les paramètres de la fonction strncmp. Résoudre.

Discuter des méthodes dynamique et statique.