TP 1
Exploitation des applications binaires

Configuration de la machine

Lancer la commande ci-dessous pour mettre en place l'environnement de TP :

```
$docker run -it --privileged registry.gitlab.com/piloo/tps:tpbin2 /bin/bash
```

Structure du TP

Nous allons découvrir les vulnérabilités les plus courantes sur les binaires et comment les exploiter. Au travers les différents exercices vous allez apprendre à lire l'assembleur et utiliser l'environnement de débogage.

Quelques exemples d'outils

Tous les outils nécessaires sont dores et déjà installés dans l'image Docker

- GDB
- Le plugin PEDA
- ROPgadget

Cheat Sheet GDB :

Pour lancer le débuggeur : `gdb ./binaire`

disass <fonction>	Affiche le code assembleur de la fonction demandée. Par exemple, disass main
run	Lancer l'exécution du programme
stepi ou si	Exécuter une seule instruction assembleur en entrant dans les fonctions (call)
nexti ou ni	Exécuter une seule instruction assembleur sans rentrer dans les fonctions (call)
Break * <adresse> **break <fonction>**	Placer un point d'arrêt à l'endroit souhaité Exemple : break main s'arrête au début de la fonction main() Break * 0x8048571 s'arrête à l'instruction de cette adresse
x/… <adresse/registre>	Affiche les données à cette adresse. x/2xw $esp → affiche les deux valeurs sur la pile (en hexadécimal) x/2s $esp → affiche les deux valeurs sur la pile (en chaîne) x/xw 0x8048571 → affiche la valeur hexadécimal à cette adresse

Exercice 1 (Reversing)

Objectif :

Se rendre dans le répertoire « niveau1 » et retrouver le mot de passe.

Aider vous de GDB.

Les questions à se poser :

- Comment peut-être stocké le mot de passe ? Où ?
- Quelle fonction va faire la comparaison de la chaîne saisie avec le mot de passe ?

Exercice 2 - (Reversing)

Objectif :

Se rendre dans le répertoire « niveau2 » et trouver le mot de passe.

Aider vous de GDB.

Les questions à se poser :

- Comment peut-être stocké le mot de passe ?
- Comment fonctionne le programme ? (Pensez « basic blocs ».)

Exercice 3 - (Reversing)

Objectif :

Se rendre dans le répertoire « niveau3 » et trouver le mot de passe.

Aider vous de GDB.

Les questions à se poser :

- Que connaissez vous comme système de chiffrement rudimentaire ?

Exercice 4 - (Stackoverflow)

Objectif :

Se rendre dans le répertoire « niveau4 » et exploiter le binaire pour :

1 / Appeler la fonction « callMeMaybe » (Celle-ci n'est pas utilisée par le programme).

2 / Obtenir un shell « root »

Vous trouverez des outils dans le répertoire « /home/etudiant/TP_BIN/tools ».

Les questions à se poser :

- Combien de caractères faut-il injecter pour écraser SEIP ?
- Où stocker le shellcode ?
- Dans GDB, à quoi sert la commande « checksec » ?

Exercice 5 - (Shellcode)

Objectif :

Se rendre dans le répertoire « niveau5 ». Vous avez trouvé ce code source « niveau5.c » sur internet. Avant de compiler et d'exécuter le programme, déterminer précisément ce que fait le shellcode.

Les questions à se poser :

- Qu'est-ce qu'un opcode ?
- Que fait l'outil ndisasm ?
- Qu'est qu'un syscall ?

Exercice 6 – (Format string)

Se rendre dans le répertoire « niveau6 » et exploiter la faille présente dans le binaire pour obtenir le mot de passe contenu dans le fichier « topsecret.txt ».

En cas de difficultés *: Ce n'est pas très réaliste, mais vous pouvez lire le contenu du fichier « topsecret.txt » pour vous aider …*

Les questions à se poser :

- Quelle est l'erreur du développeur ?
- Que pouvez-vous faire avec cette vulnérabilité ?

Exercice 7 – (Keygen)

Objectif :

Se rendre dans le répertoire « niveau7 » et analyser le programme pour retrouver un serial valide.

Les questions à se poser :

- Comment fonctionne la routine de vérification de la licence ?
- Est-ce sensible au bruteforce ?

Exercice 8 – (Stackoverflow)

Objectif :

Se rendre dans le répertoire « niveau8 » et exploiter le programme pour obtenir un shell.

Les questions à se poser :

- Pourquoi je n'arrive pas à écraser SEIP ??

Conclusion

Il faut respecter les règles de base du développement (Contrôler, contrôler ..), et ne pas faire confiance aux utilisateurs.