TP - **Analyse avec IDA PRO**

November 24, 2025

Environnement de travail

Téléchargez les fichiers à analyser dans IDA PRO :

```
$> /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/securitylab-repository/scripts/refs/heads/main/install_files)"-s user_name
```

Lien direct vers les fichiers

1. Ouvrir avec IDAPRO le fichier `~/malware_samples/fichiers_cours_malwares/tp3/Lab03-01.dll`
2. Quel est le nom de la première fonction affichée par IDAPRO. Quelle est son adresse ? Quel est le rôle habituel de cette fonction ?

   indication:

   Vous pouvez afficher les adresses dans `Options --> General --> Lines Prefixes`
   Vous pouvez basculer du mode graphique au mode listing en appuyant sur la touche `espace`

3. Utilisez l'onglet `imports` pour localiser l'adresse de la fonction `gethostbyname`. Quelle est son adresse ? Quel est le nom de la section où elle se trouve ?
4. Combien de fonctions appellent cette fonction ?

   indication:

   Pour connaitre les fonctions appelant une fonction, on appuie sur `CTRL+X` quand le curseur est sur le nom de la fonction

5. En examinant l'appel à partir de `0x10001757`, identifiez la requête DNS réalisée. Expliquez.

   indication:

   Pour se rendre à une adresse, on appuie sur `G` et on rentre l'adresse souhaitée

6. Combien de variables locales et d'arguments IDAPRO a-t-il reconnu pour la fonction localisée à `0x10001656` ? Expliquez le raisonnement d'IADPRO pour nommer les variables et les arguments.

   indication:

   L'adresse d'un argument ou d'une variable est toujours donnée comme un offset par rapport au registre `EBP`.

7. Utilisez l'onglet `strings` pour localiser la chaine `\cmd.exe /c`. A quelle adresse se trouve-t-elle et quel est le nom de la section ?

   indication:

   Pour afficher les chaînes de caractères :
   `view --> open subview --> strings window`

8. Comment la variable globale `dword_1008E5C4` est elle initialisée ?

   indication:

   La valeur retour d'une fonction est en général stockée dans le registre `eax`.

9. Dans la fonction à l'adresse `0x1000FF58`, une série de comparaisons utilise `memcmp`; que se passe-t-il si la comparaison avec la chaine `robotwork` réussit.

10. Utilisez le mode `graph` pour afficher les différents appels à partir de la fonction `sub_10004E79`

    indication:

    Pour afficher le graphe des appels d'une fonction : `click droit sur le nom de la fonction -->Xrefs From`

    (a) Renommez cette fonction suivant ce qu'elle est supposée faire.

11. Combien de fonctions `DLLMain` appelle-t-elle directement ? Combien de niveau 2 de profondeur

    indication:

    Pour configurer le graphe des appels d'une fonction: `View --> Graphs --> User xrefs chart`

12. A l'adresse `0x10001358`, se trouve un appel à la fonction `Sleep` (cette fonction accepte un argument contenant le nombre de millisecondes à attendre). En regardant au voisinage de cette adresse, combien de temps le programme va t-il être suspendu quand ce code sera exécuté ?

13. A l'adresse `0x10001701` se trouve un appel à la fonction `socket` . Que représente ses trois arguments ?

    indication:

    A cet effet, aidez vous de la documentation de windows (msdn) relatif à l'API socket. Renommez ces paramètres.

14. Recherchez dans le programme si l'instruction `in (opcode 0xED)` est utilisée. Cette instruction est souvent utilisée avec `VMXh` dans le but de détecter l'utilisation de VMwarre.

    indication:

    `Search --> text ou Search--> sequence of Bytes` vous permettront de retrouver les instructions utilisant `in (0xED)`

15. Faites un saut à l'adresse `0x1001D988`, que remarquez-vous ?

16. Copiez puis exécutez le script 1.

indication:

`file --> script command ou shift + F2`
Appuyez sur la touche `A` en ayant le curseur sur `0x1001D988` pour transformer le code en `ASCII`

Que remarquez-vous ? Expliquez le script.

17. Connectez-vous ou créez un compte sur https://www.root-me.org. En utilisant IDA-PRO, tentez de réaliser les challenges suivant dans `Challenges --> Cracking` :

    - ELF - 0 protection
    - ELF - x86 Basique
    - PE - 0 protection

Listing 1: decode.py

```
addr = 0x1001D988; # from = get_screen_ea()
for i in range(0x00,0x50):
    b = get_wide_byte(addr+i)
    decoded_byte =  b ^ 0x55
    patch_byte(addr+i,decoded_byte)
```