Reverse - CheckMyStrings - 25 points

Waïl TAHMAOUI, Kévin DUVERGER

Table des matières

1 Résolution CheckMyStrings : 2
1.1 En utilisant des outils très légers : 2
1.2 En utilisant Ghidra : 3

1 Résolution CheckMyStrings :

1.1 En utilisant des outils très légers :

C'est le premier challenge de la catégorie **reverse** et habituellement la première chose que l'on tente est de lister les chaînes de caractères du fichier (si le mot de passe est écrit en clair, il sera aussi en clair dans l'exécutable). De plus, le nom du challenge, **CheckMyStrings** nous oriente assez fortement vers cette possibilité.

Une première façon de le résoudre est d'utiliser la commande **strings** qui permet de faire exactement ce que l'on veut : afficher les chaînes de caractères dans un fichier exécutable. Sous Linux elle est présente par défaut, sous Windows ce n'est pas le cas mais elle est assez souvent livrée avec **gcc** qui est donc une possibilité pour l'obtenir.
On peut donc lancer `strings checkMyStrings.exe` et en parcourant les chaînes vous pouvez voir :

```
__register_frame_info
__deregister_frame_info
this_is_not_the_password !
Please enter the password :
drowssap_eht_ton !
r3v3rs3_1s_3asy
Yay ! You got the good password !
You don't have the right password !
not_password_too
Unknown error
_matherr(): %s in %s(%g, %g)  (retval=%g)
Argument domain error (DOMAIN)
Argument singularity (SIGN)
Overflow range error (OVERFLOW)
The result is too small to be represented (UNDERFLOW)
Total loss of significance (TLOSS)
Partial loss of significance (PLOSS)
Mingw-w64 runtime failure:
Address %p has no image-section
  VirtualQuery failed for %d bytes at address %p
  VirtualProtect failed with code 0x%x
  Unknown pseudo relocation protocol version %d.
  Unknown pseudo relocation bit size %d.
GCC: (i686-posix-dwarf-rev0, Built by MinGW-W64 project) 8.1.0
```

Vous pouvez voir que nous avons trouvé le flag qui est `cryptisCTF{r3v3rs3_1s_3asy}` ! Il aurait été possible d'aller plus vite en n'affichant que les chaînes qui ont 10 caractères ou plus (par défaut `strings` considère qu'une suite de 4 caractères affichables est une chaîne), et la commande serait : `strings checkMyStrings.exe -n 10`

Une autre façon de faire est de chercher un outil en ligne qui vous permet de résoudre ce problème. J'ai pu le faire sur un site se nommant `https://suip.biz/` et on obtient sensiblement la même chose que pour `strings` :

The following lines were found in the file:

```
libgcc_s_dw2-1.dll
__register_frame_info
__deregister_frame_info
this_is_not_the_password !
Please enter the password :
drowssap_eht_ton !
r3v3rs3_1s_3asy
Yay ! You got the good password !
You don't have the right password !
not_password_too
Unknown error
_matherr(): %s in %s(%g, %g)  (retval=%g)\n
Argument domain error (DOMAIN)
Argument singularity (SIGN)
Overflow range error (OVERFLOW)
The result is too small to be represented (UNDERFLOW)
Total loss of significance (TLOSS)
Partial loss of significance (PLOSS)
Mingw-w64 runtime failure:\n
```

Pour finir, vous pouvez aussi recoder un programme qui vous permet d'afficher les chaînes de caractères d'un fichier. Voici ce que cela donnerait si on le codait en C (on peut facilement modifier la longueur minimale des chaînes) :

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define LONGUEUR_MIN_CHAINE 4
#define LONGUEUR_MAX_CHAINE 1000

int main() {
    // Initialisation
    char buffer[LONGUEUR_MAX_CHAINE];
    const char* nomFichier = "checkMyStrings.exe";
    FILE* fichier = fopen(nomFichier, "rb");
    // Affichage
    unsigned int indexInBuffer = 0;
    while (!feof(fichier)) {
        unsigned char c = fgetc(fichier);
        if (c >= 32 && c < 128) {
            buffer[indexInBuffer++] = c;
        } else {
            if (indexInBuffer >= LONGUEUR_MIN_CHAINE) printf("%s\n", buffer);
            memset(buffer, 0, indexInBuffer);
            indexInBuffer = 0;
        }
    }
    // Fin et nettoyage
    fclose(fichier);
}
```

Et en python :

```python
fichier = open("checkMyStrings.exe", "rb")
data = fichier.read()
fichier.close()

buffer = ""
for i in range(0, len(data)) :
    if data[i] >= 32 and data[i] < 128 :
        buffer += chr(data[i])
    else :
        if len(buffer) >= 4 : print(buffer)
        buffer = ""
```

1.2 En utilisant Ghidra :

Un autre outil que l'on peut utiliser ici est Ghidra qui va décompiler un exécutable pour retrouver un code source qui aurait pu le générer. Après une simple recherche du mot-clé "password" en sélectionnant tous les champs, on trouve :

```c
int __cdecl _main(int _Argc,char **_Argv,char **_Env) {
    FILE *pFVar1;
    int iVar2;
    char acStack_401 [1005];
    char *pcStack_14;
    ___main();
    printf("Please enter the password : ");
    pFVar1 = (FILE *)(*(code *)__imp____acrt_iob_func)(0);
    fgets(acStack_401 + 1,999,pFVar1);
    iVar2 = strlen(acStack_401 + 1);
    if (acStack_401[iVar2] == '\n') {
        iVar2 = strlen(acStack_401 + 1);
        acStack_401[iVar2] = '\0';
    }
    pcStack_14 = "drowssap_eht_ton !";
    iVar2 = strcmp(acStack_401 + 1,"r3v3rs3_1s_3asy");
    if (iVar2 == 0) {
        puts("Yay ! You got the good password !");
    } else {
        puts("You don\'t have the right password !");
    }
    return 0;
}
```

On peut directement voir le mot de passe codé en dur : `r3v3rs3_1s_3asy`.