WRITE-UP

HACKY' NOV

Overflowing - C

Mehdi Mammar

HACKY'NOV

Hacky'Nov est une association créée dans le cadre des YDAYS organisés par l'école YNOV qui organise chaque année un CTF afin d'initier le grand public aux différentes problématiques de cybersécurité.

L'événement est organisé par les étudiants du campus YNOV d'Aix-en-Provence et se décompose en trois parties.

La première partie est l'organisation d'un Capture The Flag (CTF). Chaque étudiant, de bachelor 1 à master 2 propose des challenges de cybersécurité, afin que les participants puissent en résoudre le maximum et gagner la compétition ! Les challenges sont axés de sorte que même les débutants puissent en résoudre un maximum tout en sachant faire plaisir aux plus expérimentés

La deuxième partie est dédiée à l'organisation de conférences autour de problématiques et sujets de cybersécurité. Elles sont proposées soit par des étudiants volontaires, soit par des intervenants externes afin de former et de sensibiliser les participants sur des sujets ciblés.

La troisième et dernière partie permet d'organiser la rencontre des étudiants avec des entreprises travaillant autour de la cybersécurité. Les entreprises partenaires de l'événement qui sont en majorité de grands acteurs du domaine, auront un espace unique et dédié à la mise en relation avec les participants, qui sont pour la plupart, des étudiants en cybersécurité.

https://hackynov.fr/

Table des matières

Partie 1 : Présentation du challenge

Nom du challenge : Overflowing

Domaine : Programmation C

Difficulté : Moyenne ★★★★★

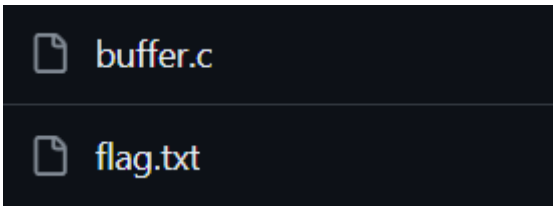
Auteur : Mehdi Mammar

Description : Un programme en C qui nous donne le Flag lorsqu'on a de la chance, ou pas ?

Impossible de gagner !

Partie 2 : Sources

Le challenge comporte les fichiers suivants :



Partie 3 : Résolution

Le but de ce challenge est d'exploiter la fonction gets() qui lis l'entrée de l'utilisateur sans se soucier de la taille de la chaine de caractère, ce qui peut être contourner pour faire en sorte que la fonction vulnerable _fonction() execute bien la fonction win().

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>

#define BUFSIZE 32
#define FLAGSIZE 64

// Fonction pour afficher le contenu du fichier "flag.txt"
void win() {
    char buf[FLAGSIZE];
    FILE *f = fopen("flag.txt", "r");
    if (f == NULL) {
        printf("Erreur: 'flag.txt' n'est pas dans votre directory.\n");
        exit(1);
    }

    fgets(buf, FLAGSIZE, f);
    printf("%s", buf);
    fclose(f);
}

// Fonction vulnérable qui lit l'entrée de l'utilisateur et compare avec "win"
void vulnerable_function() {
    char buf[BUFSIZE];
    printf("Tentez votre chance ?  : ");
    gets(buf);

    printf("La Chance vous sourit ?\n");
    if (strcmp(buf, "win") == 0) {
        printf("Bien jouer !\n");
        win();
    } else {
        printf("Mince, retentez votre chance.\n");
    }
}

int main(int argc, char **argv) {
    setvbuf(stdout, NULL, _IONBF, 0);

    // Désactive la mise en mémoire tampon de la sortie standard
    // pour afficher immédiatement les messages à l'écran
    vulnerable_function();

    return 0;
}
```

Pour cela il faut faire en sorte que la fonction vulnerable_fonction renvois l'adresse de la fonction win() dans la mémoire.

Pour obtenir l'adresse de win() on peut par exemple utiliser gdb :

```
0000000140002ab0 T VirtualQuery
00000001400014d6 T vulnerable_function
0000000140007090 b was_init.0
0000000140001450 T win
00000001400013d0 T WinMainCRTStartup
```

On obtient ici la mémoire de win .

Il suffirant donc de bufferiser en augmentant progressivement la taille du buffer, d'ecraser la mémoire et de fournir l'adresse de la fonction win.

```
La Chance vous sourit ?
Bien jouer !
flag{bien0ouej0tu0as0le0flag}
C:\Users\letak\Documents\GitHub\Hackynov\Source>
```