Toujours pendu !

Auteur	Catégorie	Niveau
Cécile	Pwn	Facile

Énoncé :

Intro

J'essaye de gagner ce jeu du pendu car il cache un secret, mais il semble mal fonctionner. Peux tu m'aider à trouver le flag ?

Pièce Jointe

Un exécutable

Préparation du challenge

Objectif

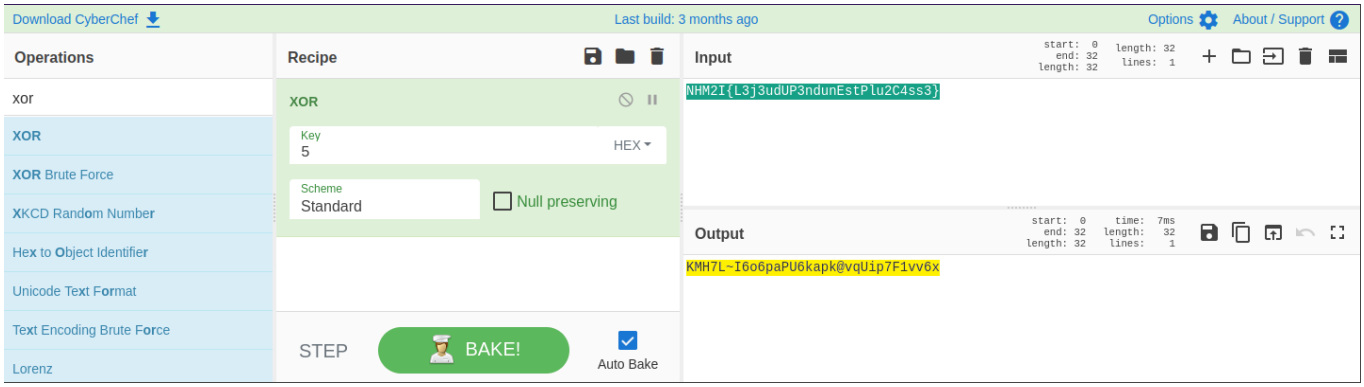
Modifier le programme pour le faire fonctionner correctement et ainsi obtenir le flag.

Flag

NHM2I{L3j3udUP3ndunEstPlu2C4ss3}

Démarche

Avec cyberchef, on obfusque le flag avec un Xor avec une clé de 5 :



On code un jeu du pendu fonctionnel avec un nom de jeu à deviner (assez facile).
On ajoute une fonction permettant de desobfusquer le flag avec un XOR comme cela nous pouvons intégrer le flag sans qu'il soit lisible de prime abord.
On teste le programme :

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

// On crée une fonction qui chiffre le flag avec un XOR ayant une clé de 5 :
char* obtenirFlag() {
    char message[] = "KMH7L~I6o6paPU6kapk@vqUip7F1vv6x";
    char xorKey = 5;
    int i, longueurMessage;

    longueurMessage = strlen(message);

    // On applique le XOR pour déchiffrer le message :
    for (i = 0; i < longueurMessage; i++) {
        message[i] ^= xorKey;
    }

    // On copie le message déchiffré dans une nouvelle chaîne de caractères allouée dynamiquement :
    char* messageDechiffre = malloc((longueurMessage + 1) * sizeof(char));
    strcpy(messageDechiffre, message);

    return messageDechiffre;
}

// On créé le programme principal
int main() {
    char motSecret[] = "leagueoflegend", motAffiche[100];
    int longueurMot, i, j;
    char lettre;
    int chances = 10;
    int lettreTrouvee = 0;

    // On initialise le mot à deviner avec des tirets :
    longueurMot = strlen(motSecret);
    for (i = 0; i < longueurMot; i++) {
        motAffiche[i] = '-';
```

```
    }
    motAffiche[i] = '\0';

    // On créé la boucle principale du jeu avec un compteur de 10 chances :
    while (chances > 0 && strcmp(motSecret, motAffiche) != 0) {
        // On affiche les lettres trouvées et les chances restantes :
        printf("Mot : %s\n", motAffiche);
        printf("Chances restantes : %d\n", chances);

        // On demande la saisie d'une lettre :
        printf("Entrez une lettre : ");
        scanf("%c", &lettre);


        // On 'consomme' le caractère '\n' car si on  utilise une autre fonction de saisie de caractères sans vider le buffer,
        // cela peut entrainer un comportement imprevu du programme.
        // La fonction getchar() lit et supprime le caractère de nouvelle ligne du tampon,
        // permettant ainsi aux fonctions de saisie de caractères suivantes de fonctionner correctement.

        getchar();

        // On vérifie si la lettre est dans le nom du jeu à deviner :
        lettreTrouvee = 0;
        for (j = 0; j < longueurMot; j++) {
            if (motSecret[j] == lettre) {
                motAffiche[j] = lettre;
                lettreTrouvee = 1;
            }
        }

        // Si la lettre n'est pas dans le nom à deviner, on décrémente les chances :
        if (!lettreTrouvee) {
            chances--;
            printf("La lettre %c n'est pas dans le nom du jeu secret.\n", lettre);
        }
    }

    // On affiche le résultat final :
    if (chances > 0) {
        char* flag = obtenirFlag();
        printf("Bravo, vous avez trouvé le flag : %s\n", flag);
    } else {
        printf("Dommage, vous avez perdu. Le nom du jeu secret était : %s\n", motSecret);
    }

    return 0;
}
```

On obtient le résultat suivant :

```
Mot : --------------
Chances restantes : 10
Entrez une lettre : l
Mot : l-------l-----
Chances restantes : 10
Entrez une lettre : e
Mot : le---e--le-e--
Chances restantes : 10
Entrez une lettre : a
Mot : lea--e--le-e--
Chances restantes : 10
Entrez une lettre : g
Mot : leag-e--lege--
Chances restantes : 10
Entrez une lettre : u
Mot : league--lege--
Chances restantes : 10
Entrez une lettre : s
La lettre s n'est pas dans le nom du jeu secret.
Mot : league--lege--
Chances restantes : 9
Entrez une lettre : o
Mot : leagueo-lege--
Chances restantes : 9
Entrez une lettre : f
Mot : leagueoflege--
Chances restantes : 9
Entrez une lettre : l
Mot : leagueoflege--
Chances restantes : 9
Entrez une lettre : n
Mot : leagueoflegen-
Chances restantes : 9
Entrez une lettre : d
Bravo, vous avez trouvé le flag : NHM2I{L3j3udUP3ndunEstPlu2C4ss3}

Process returned 0 (0x0)   execution time : 24.455 s
Press ENTER to continue.
```

On rend le programme défectueux pour que le joueur perde à chaque fois, on modifie la condition chance en inversant le signe :

```
    // On affiche le résultat final :
    if (chances < 0) {
        char* flag = obtenirFlag();
        printf("Bravo, vous avez trouvé le flag : %s\n", flag);
    } else {
        printf("Dommage, vous avez perdu. Le nom du jeu secret était : %s\n", motSecret);
    }
```

```
Mot : --------------
Chances restantes : 10
Entrez une lettre : l
Mot : l-------l-----
Chances restantes : 10
Entrez une lettre : e
Mot : le---e--le-e--
Chances restantes : 10
Entrez une lettre : a
Mot : lea--e--le-e--
Chances restantes : 10
Entrez une lettre : g
Mot : leag-e--lege--
Chances restantes : 10
Entrez une lettre : e
Mot : leag-e--lege--
Chances restantes : 10
Entrez une lettre : o
Mot : leag-eo-lege--
Chances restantes : 10
Entrez une lettre : u
Mot : leagueo-lege--
Chances restantes : 10
Entrez une lettre : n
Mot : leagueo-legen-
Chances restantes : 10
Entrez une lettre : f
Mot : leagueoflegen-
Chances restantes : 10
Entrez une lettre : d
Dommage, vous avez perdu. Le nom du jeu secret était : leagueoflegend

Process returned 0 (0x0)   execution time : 17.399 s
Press ENTER to continue.
```

On supprime les commentaires et on le transforme en exécutable :

```
gcc -o lependu main.c
```

On s'aperçoit qu'il y a une erreur indiquant qu'il faut ajouter une bibliothèque, on l'ajoute et on relance le programme :

```
./lependu
Mot : ──────────────
Chances restantes : 10
Entrez une lettre :
```

On teste l'executable avec strings, on s'aperçoit qu'on voit le flag (meme obfusqué) donc on rajoute de faux flag et des caractères aléatoires pour noyer un peu le bon flag.

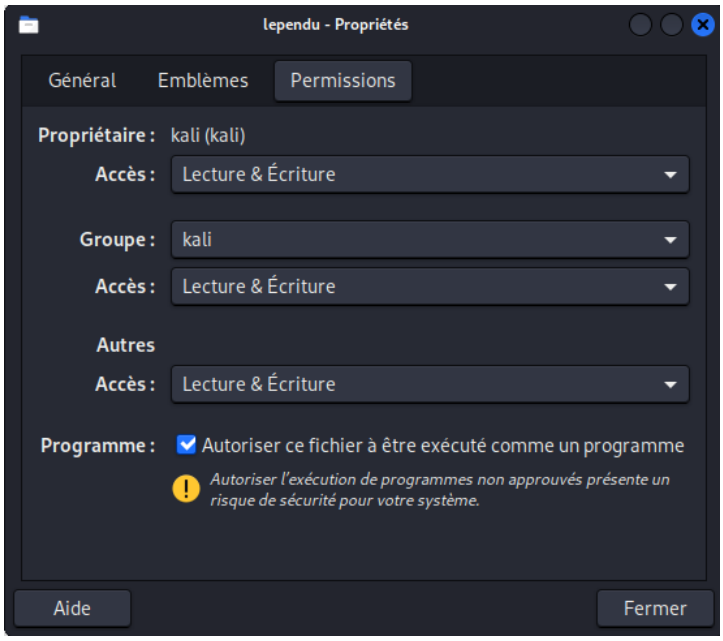
```
    char message14[] = "NOPE~N0nNOnC32Tp42IC1";
    char message15[] = "c0nFu5E~uC8tXzZt";
    char message16[] = "tR1qu3M1c0n7rE~l2u";
    char message17[] = "jK7fzDnC1pB6sLx8Wt3q";
    char message18[] = "pUv9XrZnT1bM6dA7gKfQ2c0";
    char message5[] = "NOPE~C3stP4s1c1N0nPlu2L3F74g";
    char message4[] = "KJU@~F`vqUdvIdK5kUipvIdW`ujkv`x.";
    char message3[] = "KJU@~F`vqUdvLflIdW`ujkv`x.";
    char message0[] = "NOPE~C3stP4s1c1L3F74g";
    char message9[] = "KJU@~F6m6wfm6@kf5w6x.";
    char message[] = "KMH7L~I6o6paPU6kapk@vqUip7F1vv6x";
    char message2[] = "NOPE~N0nNOnC32Tp42IC1";
    char message6[] = "NOPE~C3h3rch3Enc0r3";
    char message7[] = "NOPE~TuV45F1n1rP4rTr0uv3r";
    char message8[] = "KJU@~QpS10C4k4wU1wQw5ps6wx.";
    char message10[] = "KJU@~F`vqUdvI`GjkCidbKjkUipvx.";
    char message11[] = "NOPE~CestPasLeBonFlagNonPlus";
    char message12[] = "NOPE~Ilf4utLePatcher";
    char message13[] = "KJU@~Lic1pqI`Udqfm`wx.";
    char message19[] = "9p4zJ4C0vR1h7W8eN0gM";
    char message20[] = "pL5dG9gA0oU6fE2cB7hK";
    char message21[] = "L5jK3×8qM4pB9f6zD7v";
    char message22[] = "a1H5e6fL8pN4s0jK9uX2mI7";
```

Résolution :

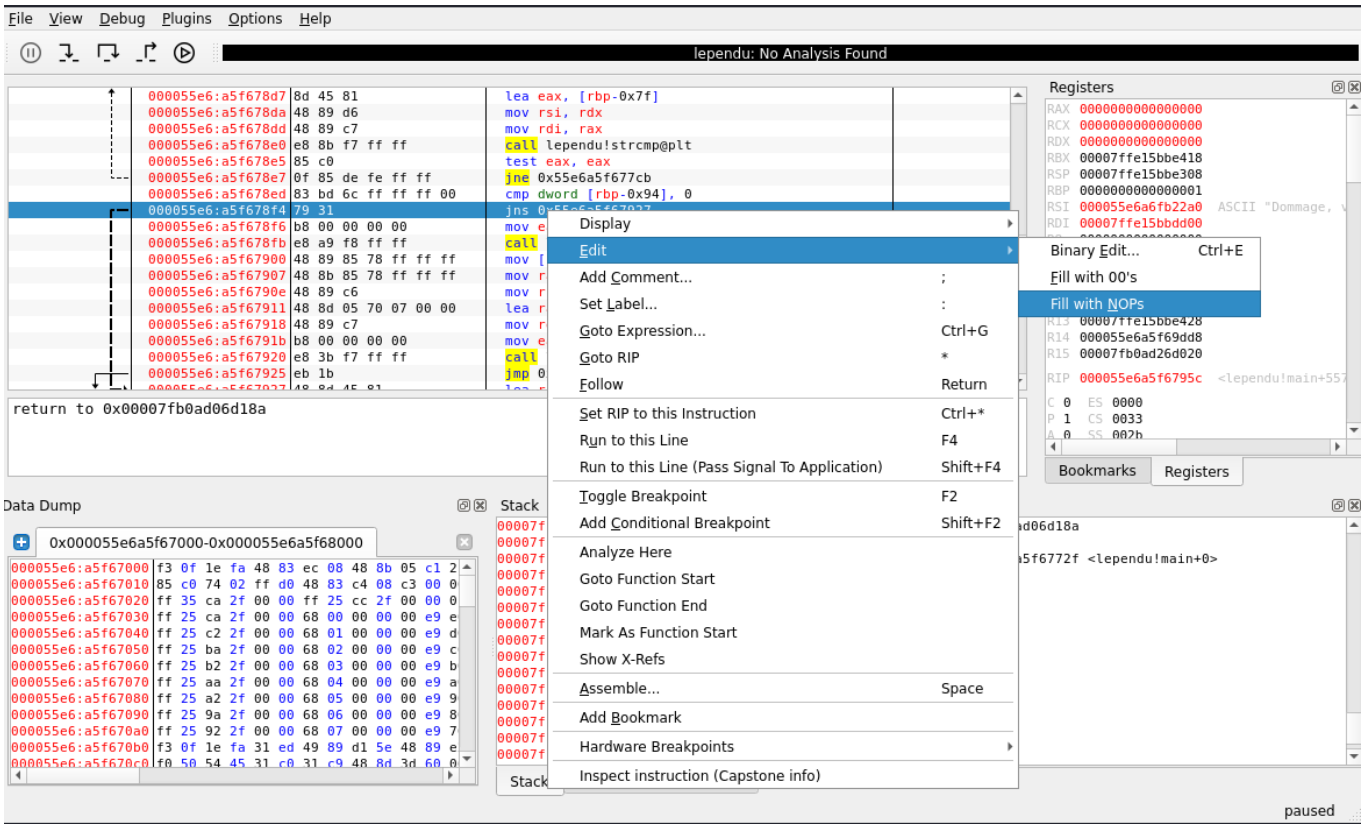
Solution débutant :

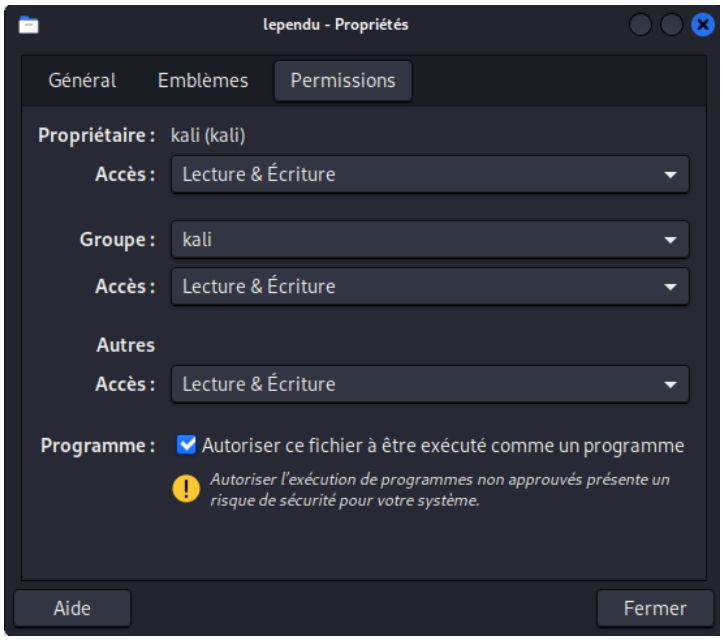
On va utiliser edb qui est un debugger avec une interface graphique.

On change les permissions du fichier pour qu'on puisse l'executer :

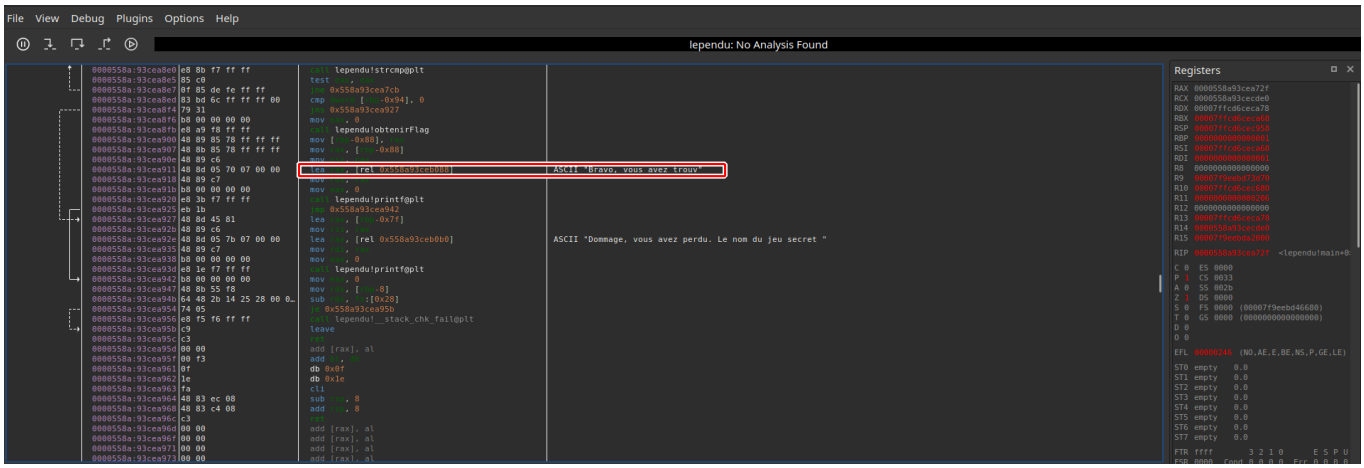


On lance edb en sudo et on ouvre le ficher lependu avec edb :

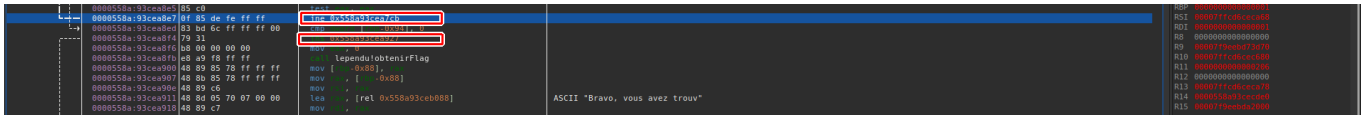




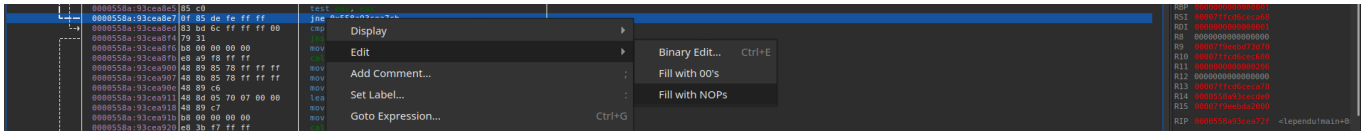
En parcourant le débugger, on voit "Bravo, vous avez trouvé", on cherche donc comment il est possible d'arriver à ce message, sauf que rien n'arrive à cette condition.



On regarde un peu plus haut, on voit la fonction "obtenir flag" qui dépend des conditions avec les instructions **jne** et **jns**:



On remplace ces conditions par des "NOP" :



Et on relance le programme, on obtient le flag :