

Challenge Midnight Flag CTF (Writeup)

Nom du challenge : WiiWii, en avant les amis

Difficulté : Facile

Description :

Un agent de notre organisation s'est vu transmettre un étrange fichier, qu'il doit utiliser sur son ancienne console de jeux. Aidez-le à en déchiffrer le secret.

Note : le format du flag sera **MF{xxxxxxxx}**

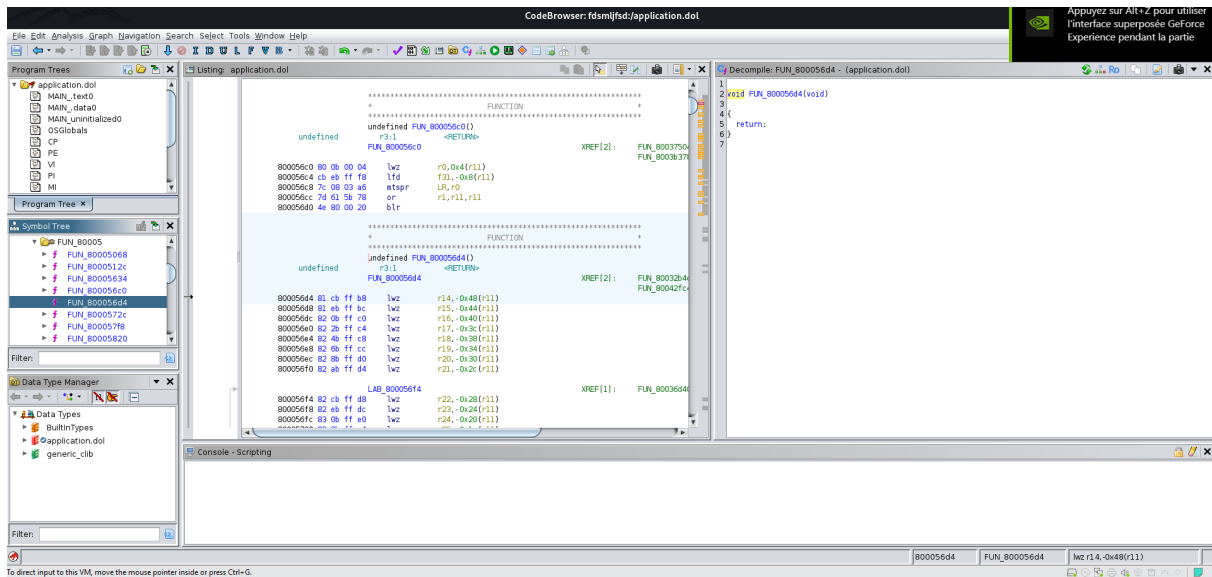
Fichier fourni : application.dol

Writeup :

On commence par examiner le fichier fourni : il s'agit d'un binaire au format "DOL", qui est apparemment le format exécutable standard pour la Wii (ainsi que la Gamecube). Des modules Ghidra existent pour désassembler facilement ce type de format (architecture PowerPc Gecko) :

<https://github.com/JoshuaMKW/GeckoLoader>

On peut alors inspecter le contenu du binaire :



En naviguant dans les fonctions de ce dernier, on finit par repérer cette fonction intéressante :

```

29 FUN_80036b30(s_request, _s_80044d30, param_1);
30 pbVar5 = &bStack_d1;
31 local_4c = DAT_8004c568;
32 uVar7 = 0;
33 local_48 = DAT_8004c56c;
34 local_42 = DAT_8004c554;
35 pbVar6 = &bStack_91;
36 local_34 = DAT_8004c562;
37 local_36 = CONCAT11((char)((ushort)DAT_8004c560 >> 8), 0x7d);
38 local_44 = DAT_8004c570;
39 local_3e = DAT_8004c558;
40 local_3a = DAT_8004c55c;
41 local_50 = CONCAT31((int3)((uint)DAT_8004c564 >> 8), 0x7b);
42 FUN_80030bf0(param_1, s_s_s_80044d40, auStack_30, abStack_90);
43 local_1c = 0;
44 local_20 = 0x494e4954;
45 while( true ) {
46     uVar1 = FUN_80031c48(abStack_90);
47     if (uVar1 < uVar7) break;
48     pbVar6 = pbVar6 + 1;
49     bVar3 = *pbVar6;
50     if ((byte)(bVar3 + 0x9f) < 0x1a) {
51         iVar4 = (uint)*(byte*)((int)&local_20 + (uVar7 & 3)) + (uint)bVar3 + -0xc2;
52         bVar3 = (char)iVar4 + (char)(iVar4 / 0x1a) * -0x1a + 0x61;
53     }
54     else if ((byte)(bVar3 + 0xbf) < 0x1a) {
55         iVar4 = (uint)*(byte*)((int)&local_20 + (uVar7 & 3)) + (uint)bVar3 + -0xa2;
56         bVar3 = (char)iVar4 + (char)(iVar4 / 0x1a) * -0x1a + 0x41;
57     }
58     pbVar5 = pbVar5 + 1;
59     *pbVar5 = bVar3;
60     uVar7 = uVar7 + 1;

```

La variable "**local_20**", une fois convertie donne le texte "INIT", et ressemble étrangement à une clé de chiffrement.

On trouve également deux chaînes de caractères suspectes :

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

On peut alors lancer le programme avec l'émulateur Dolphin :

```
Starting web server from thread ...
Thread created (65538), press B to quit.
initialized
socket created
Listening on : 192.168.1.30
Calling accept()
```

Ce dernier semble lancer un serveur HTTP sur le port 80, et reçoit des requêtes en entrée qui aboutissent actuellement à une erreur 404 :

```
(matthieu@kali) - [~/Desktop]
$ nc 192.168.1.30 80
GET /test HTTP
HTTP/1.1 404 Not Found
Content-Type: text/html

<html><body><h1>Endpoint not found</h1></body></html>
```

L'algorithme utilisé par cette boucle

```
while( true ) {
    uVar1 = FUN_80031c48(abStack_90);
    if (uVar1 < uVar7) break;
    pbVar6 = pbVar6 + 1;
    bVar3 = *pbVar6;
    if ((byte)(bVar3 + 0x9f) < 0x1a) {
        iVar4 = (uint)*(byte *)((int)&local_20 + (uVar7 & 3)) + (uint)bVar3 + -0xc2;
        bVar3 = (char)iVar4 + (char)(iVar4 / 0x1a) * -0x1a + 0x61;
    }
    else if ((byte)(bVar3 + 0xbf) < 0x1a) {
        iVar4 = (uint)*(byte *)((int)&local_20 + (uVar7 & 3)) + (uint)bVar3 + -0xa2;
        bVar3 = (char)iVar4 + (char)(iVar4 / 0x1a) * -0x1a + 0x41;
    }
}
```

semble en réalité être un algorithme de Vigenère légèrement modifié. On peut alors utiliser la clé "INIT" et réimplémenter ce dernier pour déchiffrer les deux chaînes de caractères rassemblées.

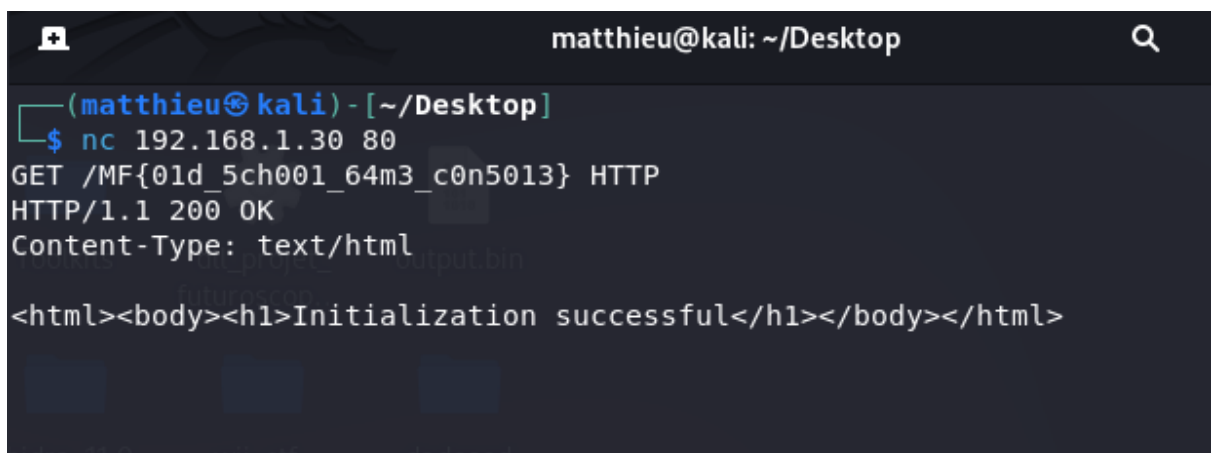
Par ailleurs, les lignes suivantes nous indiquent que les caractères "{" et "}" ont été remplacés à la volée :

```
local_36 = CONCAT11((char)((ushort)DAT_8004c560 >> 8),0x7d);  
  
local_50 = CONCAT31((int3)((uint)DAT_8004c564 >> 8),0x7b);
```

Il faut donc rassembler les deux chaînes de caractères, remplacer les deux caractères modifiés par des accolades, puis utiliser l'algorithme tel que décrit par la boucle while pour obtenir le flag :

MF{01d_5ch001_64m3_c0n5013}

On peut obtenir confirmation en le passant au programme lors de l'exécution, qui nous renvoie cette fois-ci une réponse 200 avec le message "Initialization successful" :



```
matthieu@kali: ~/Desktop  
  
(matthieu@kali) - [~/Desktop]  
$ nc 192.168.1.30 80  
GET /MF{01d_5ch001_64m3_c0n5013} HTTP  
HTTP/1.1 200 OK  
Content-Type: text/html  
  
<html><body><h1>Initialization successful</h1></body></html>
```