



Symboles (variables, fonctions, paramètres)



Renommer symbole



[Changer le type du symbole](#)

Navigation



Précédent



Liste des chaînes de caractères



Liste des fonctions



[Lister les références à cet endroit](#)

Manipulation de données



Affichage hexadécimal / décimal



Affichage ASCII



[Changer la taille de la donnée \(1 / 2 / 4 / 8 octets\)](#)



[Convertir la donnée en code assembleur](#)



[Définir le début d'une fonction](#)

Sites utiles

[MSDN](#): documentation des fonctions de l'API Windows

[MagnumDB](#): trouver les constantes associées à une valeur hardcodée

[Decompiler helpers](#): les macros bizarres utilisées par le décompilateur (HIWORD(), etc.)

Signification des couleurs

Rose: fonction importée depuis une DLL

```
CreateProcessA(0i64,
```

Bleu foncé: variable / fonction globale

```
find_process(0x91976854)
```

Bleu clair: variable locale (sur la pile)

```
shellcode_local[23] = _WinExec;
```

Commentaires



Commentaire sur la même ligne (décompilateur)

Commentaire au-dessus de la ligne (décompilateur)

Commentaire (vue assembleur)



Décompiler



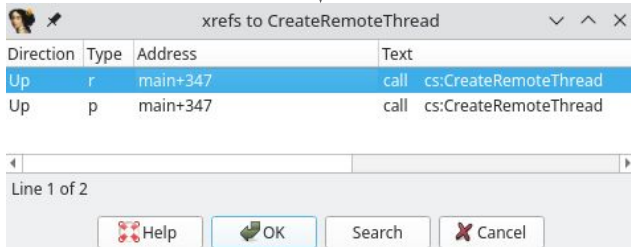
Lister les références à cet endroit

```
.idata:0000000140003040 ; DATA XREF: main+72+r  
.idata:0000000140003048 ; HANDLE (__stdcall *CreateRemoteThread)(HANDLE hProcess, LPSECURITY_ATTRIBUTES, DWORD dwStackSize, LPTSTR lpFileName, DWORD dwFlags, LPVOID lpParameter, DWORD dwCreationFlags, LPVOID lpReserved)  
.idata:0000000140003048 extrn CreateRemoteThread:qword  
.idata:0000000140003048 ; CODE XREF: main+347+p  
.idata:0000000140003048 ; DATA XREF: main+347+r  
.idata:0000000140003050 ; BOOL (__stdcall *IsDebuggerPresent)()  
.idata:0000000140003050 extrn IsDebuggerPresent:qword  
.idata:0000000140003050 ; CODE XREF: __scrt_fastfail+F8+p  
.idata:0000000140003050 ; DATA XREF: __scrt_fastfail+F8+r
```

Placer le curseur sur un symbole



Appuyer sur 



IDA vous montre tous les endroits **dans le code** où ce symbole est référencé.



Changer la taille de la donnée (1 / 2 / 4 / 8 octets)

```
.data:00405100 byte_405100    db 0BAh
.data:00405101             db 18h
.data:00405102             db 40h ; @
.data:00405103             db 0
```



```
.data:00405100 word_405100    dw 18BAh
.data:00405102             db 40h ; @
.data:00405103             db 0
```



```
.data:00405100 off_405100     dd offset __exit
```

A l'offset 0x405100, on a les 4 octets suivants:

BA 18 40 00

En appuyant sur “”, on change la manière dont IDA les interprète: comme un octet simple, comme un WORD (2 octets), comme un DWORD (4 octets)

En passant en DWORD (taille d'un pointeur 32-bits), IDA comprend automatiquement que la valeur pointe vers une fonction connue.



Convertir la donnée en code assembleur

```
.text:004010C7 unk_4010C7 db 55h ; U
.text:004010C8 db 8Bh
.text:004010C9 db 0ECh
.text:004010CA db 83h
.text:004010CB db 0ECh
.text:004010CC db 0Ch
.text:004010CD db 0C7h
.text:004010CE db 45h ; E
.text:004010CF db 0FCh
.text:004010D0 db 0
.text:004010D1 db 0
.text:004010D2 db 0
.text:004010D3 db 0
.text:004010D4 db 6Ah ; j
.text:004010D5 db 0
.text:004010D6 db 68h ; h
.text:004010D7 db 80h
.text:004010D8 db 0
.text:004010D9 db 0
.text:004010DA db 0
.text:004010DB db 6Ah ; j
..
```

Parfois, l'analyse automatique d'IDA ne parvient pas à identifier un offset comme contenant du code. Dans ce cas, il vous affiche directement les octets.

En appuyant sur , on peut forcer IDA à interpréter les octets comme du code assembleur.



```
.text:004010C7 loc_4010C7:
.text:004010C7
.text:004010C8
.text:004010CA
.text:004010CD
.text:004010D4
.text:004010D6
.text:004010DB
.text:004010DD
.text:004010DF
.text:004010E1
.text:004010E6
.text:004010EB
.text:004010F1
.text:004010F4
.text:004010F8
.text:004010FA
.text:004010FF
.text:00401100

push ebp
mov ebp, esp
sub esp, 0Ch
mov dword ptr [ebp-4], 0
push 0
push 80h
push 4
push 0
push 2
push 40000000h
push offset FileName ; "pract
call ds:CreateFileA
mov [ebp-8], eax
cmp dword ptr [ebp-8], 0FFFFh
jnz short loc_4010FF
jmp loc_40143D ; CODE X
```



Définir le début d'une fonction

```
.text:004010C7 loc_4010C7:
.text:004010C7
.text:004010C8      push    ebp
.text:004010CA      mov     ebp, esp
.text:004010CD      sub     esp, 0Ch
.text:004010D4      mov     dword ptr [ebp-4], 0
.text:004010D6      push    0
.text:004010D8      push    80h
.text:004010DA      push    4
.text:004010DD      push    0
.text:004010DF      push    2
.text:004010E1      push    40000000h
.text:004010E6      push    offset FileName ; "pract
.text:004010EB      call    ds:CreateFileA
.text:004010F1      mov     [ebp-8], eax
.text:004010F4      cmp     dword ptr [ebp-8], 0FFFF
.text:004010F8      jnz     short loc_4010FF
.text:004010FA      jmp     loc_40143D
```



Le décompilateur ne fonctionne que sur des fonctions, pas sur du code qui se balade n'importe où. En appuyant sur , on demande à IDA de considérer le code commençant à l'offset donné comme étant une fonction.

Le début et la fin de la fonction peuvent être édités en appuyant sur   (clic droit, "Edit function")

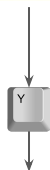
```
.text:004010C7 ; int __cdecl sub_4010C7(int Buffer)
.text:004010C7 sub_4010C7      proc near      ; CODE XREF: fn+21+p
.text:004010C7
.text:004010C7      var_C      = dword ptr -0Ch
.text:004010C7      hFile      = dword ptr -8
.text:004010C7      NumberOfBytesWritten= dword ptr -4
.text:004010C7      Buffer      = dword ptr 8
.text:004010C7
.text:004010C7      push     ebp
.text:004010C8      mov     ebp, esp
.text:004010CA      sub     esp, 0Ch
.text:004010CD      mov     [ebp+NumberOfBytesWritten], 0
.text:004010D4      push    0 ; hTemplateFile
.text:004010D6      push    80h ; dwFlagsAndAttributes
.text:004010D8      push    4 ; dwCreationDisposition
.text:004010DA      push    0 ; lpSecurityAttributes
.text:004010DD      push    2 ; dwShareMode
.text:004010DF      push    40000000h ; dwDesiredAccess
.text:004010E1      push    offset FileName ; "practicalmalwareanalysis.log"
.text:004010E6      call    ds:CreateFileA
.text:004010EB      mov     [ebp+hFile], eax
.text:004010F1      cmp     [ebp+hFile], 0FFFFFFFFh
.text:004010F4      jnz     short loc_4010FF
.text:004010FA      jmp     loc_40143D
```



Changer le type du symbole

Changer le type d'une variable ou d'un paramètre permet à IDA de résoudre (par exemple) les offsets dans les structures, et d'afficher le résultat de manière plus parlante que juste avec de l'arithmétique de pointeurs.

```
CloseHandle(ProcessInformation[1]);  
WaitForSingleObject(ProcessInformation[0], 0xFFFFFFFF);  
CloseHandle(ProcessInformation[0]);
```



Please enter a string

Please enter the type declaration

OK Cancel

```
CloseHandle(ProcessInformation.hThread);  
WaitForSingleObject(ProcessInformation.hProcess, 0xFFFFFFFF);  
CloseHandle(ProcessInformation.hProcess);
```