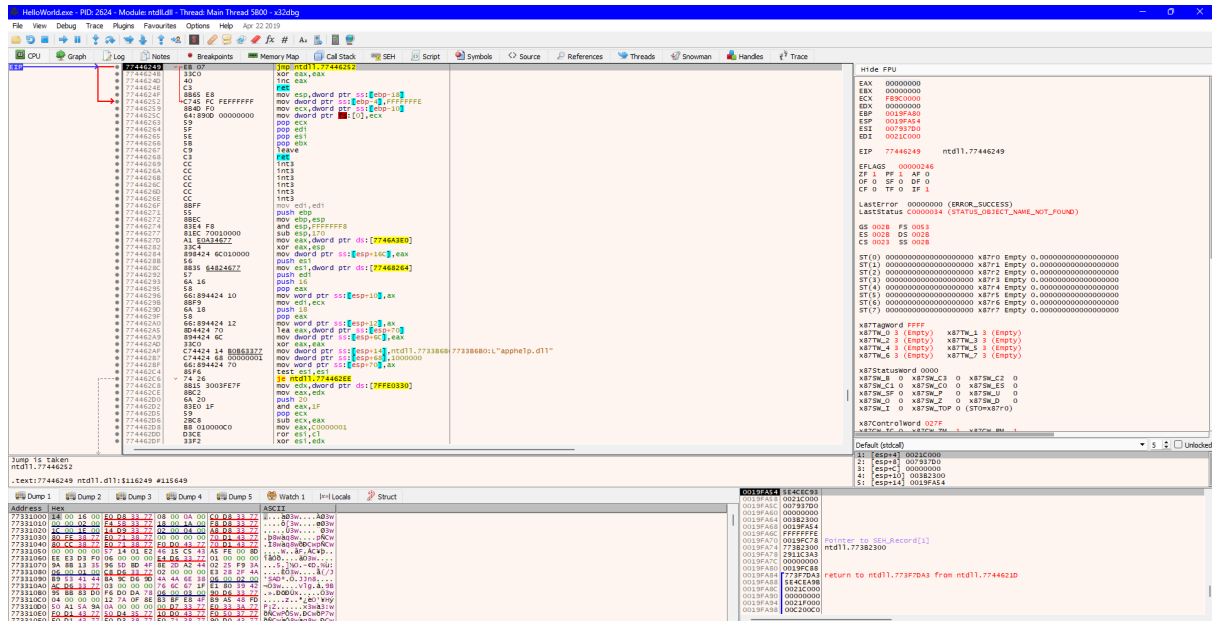


A. Appel de fonctions : a-b

Après avoir compilé le programme HelloWorld, je l'ai exécuté à l'aide d'un débogueur (ici x32dbg) pour observer comment s'effectue l'appel à la fonction printf.



Dans un premier temps, on peut voir que le programme effectue plusieurs instructions push, notamment pour placer la valeur 2A (en hexadécimal, soit 42 en décimal) ainsi qu'une adresse pointant vers une chaîne de caractères dans la pile. Cela correspond aux arguments que la fonction printf va utiliser. Ces arguments sont empilés dans l'ordre inverse de leur utilisation dans la fonction.

00401000	6A 2A	push 2A	EntryPoint
00401002	68 00304000	push helloworld.403000	403000:"Hello world : %d\n"
00401007	FF15 0C204000	call dword ptr ds:[<printf>]	
0040100D	68 12304000	push helloworld.403012	403012:"Pause\r\n"
00401012	FF15 08204000	call dword ptr ds:[<system>]	
00401018	83C4 04	add esp,4	
0040101B	B8 00000000	mov eax,0	
00401020	50	push eax	
00401021	E8 00000000	call <JMP.&ExitProcess>	call \$0
00401026	FF25 00204000	jmp dword ptr ds:[<ExitProcess>]	JMP.&ExitProcess
0040102C	0000	add byte ptr ds:[eax],al	

Ensuite, la fonction printf est appelée via l'instruction CALL. Cette instruction pousse l'adresse de retour sur la pile, puis effectue un saut vers le code de la fonction printf. Une fois dans printf, celle-ci doit récupérer les arguments précédemment empilés. Pour cela, elle s'appuie sur le contenu de la pile à partir de la base de pile (EBP), qui ne change pas durant l'exécution de la fonction.

Masquer FPU		
EAX	0019FF58	
EBX	00317000	
ECX	00401000	"j*h"
EDX	00401000	"j*h"
EBP	0019FF68	
ESP	0019FF3C	
ESI	00401000	"j*h"
EDI	00401000	"j*h"

Dans notre cas, à un instant donné de l'exécution dans printf, le registre EBP contient la valeur 0019FF68.

En observant la pile à cette adresse, on retrouve bien la structure attendue :

EBP + 8 → adresse de la chaîne "Hello World : %d\n"

EBP + C (soit +12 en décimal) → valeur entière 42

0019FF60	465E14BC	
0019FF64	FFFFFFFE	
0019FF68	0019FF84	
0019FF6C	0040100D	return to helloworld.0040100D from ???
0019FF70	00403000	"Hello world : %d\n"
0019FF74	0000002A	
0019FF78	77795D49	return to kernel32.77795D49 from ???

Cela montre que les arguments sont bien empilés dans l'ordre attendu et accessibles via des décalages fixes par rapport à EBP.