Le TLS, mais pas celui auquel vous pensez

Maximilien CHAUX - Neverhack

Whoami

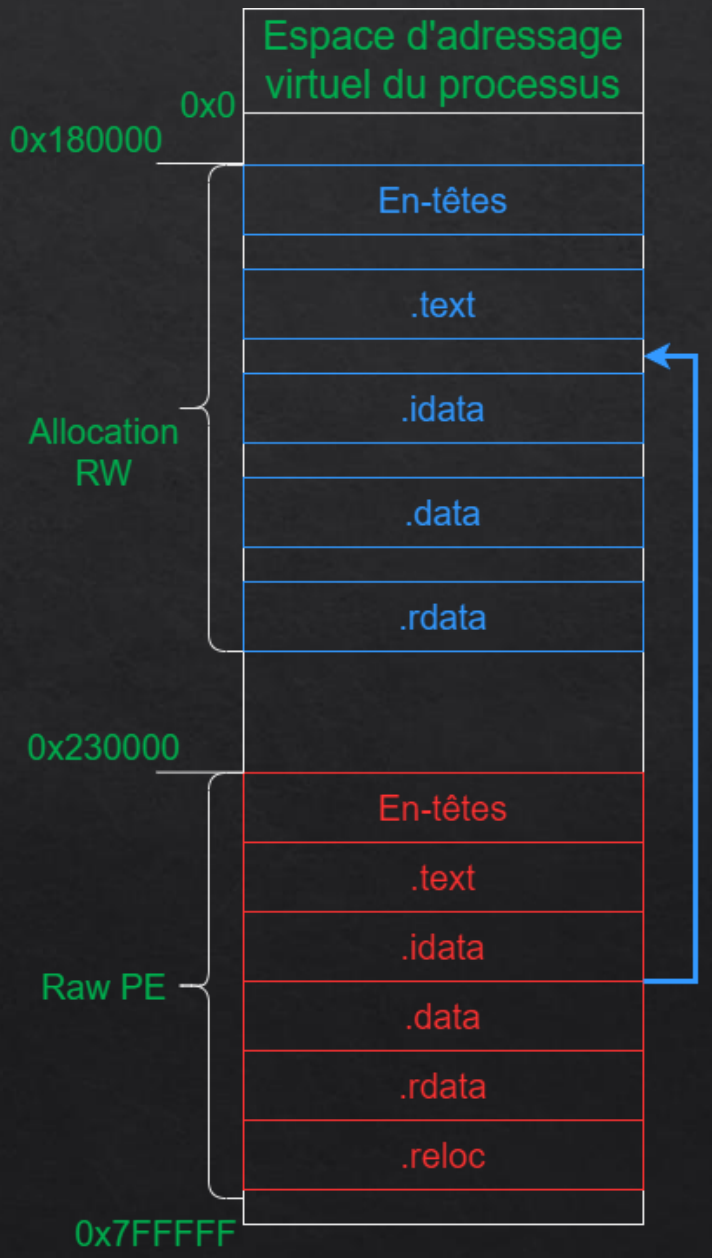
NEVERHACK

Contexte

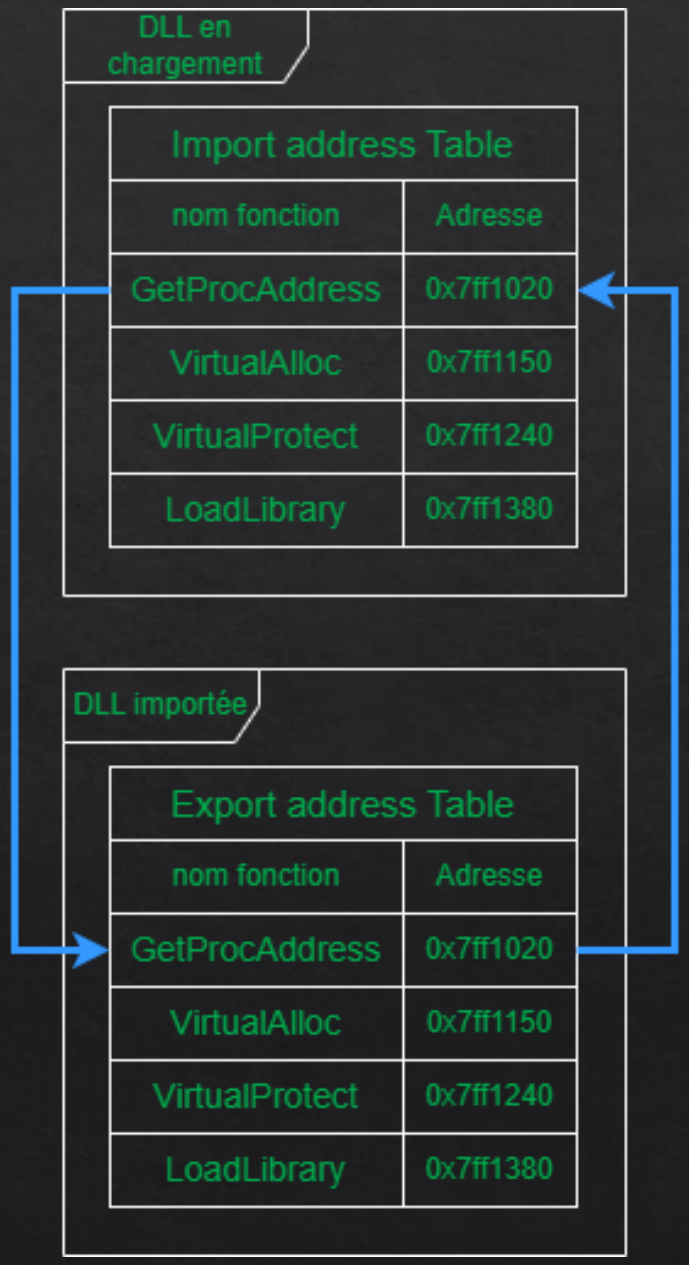
C&C automatisé

Loading de PE

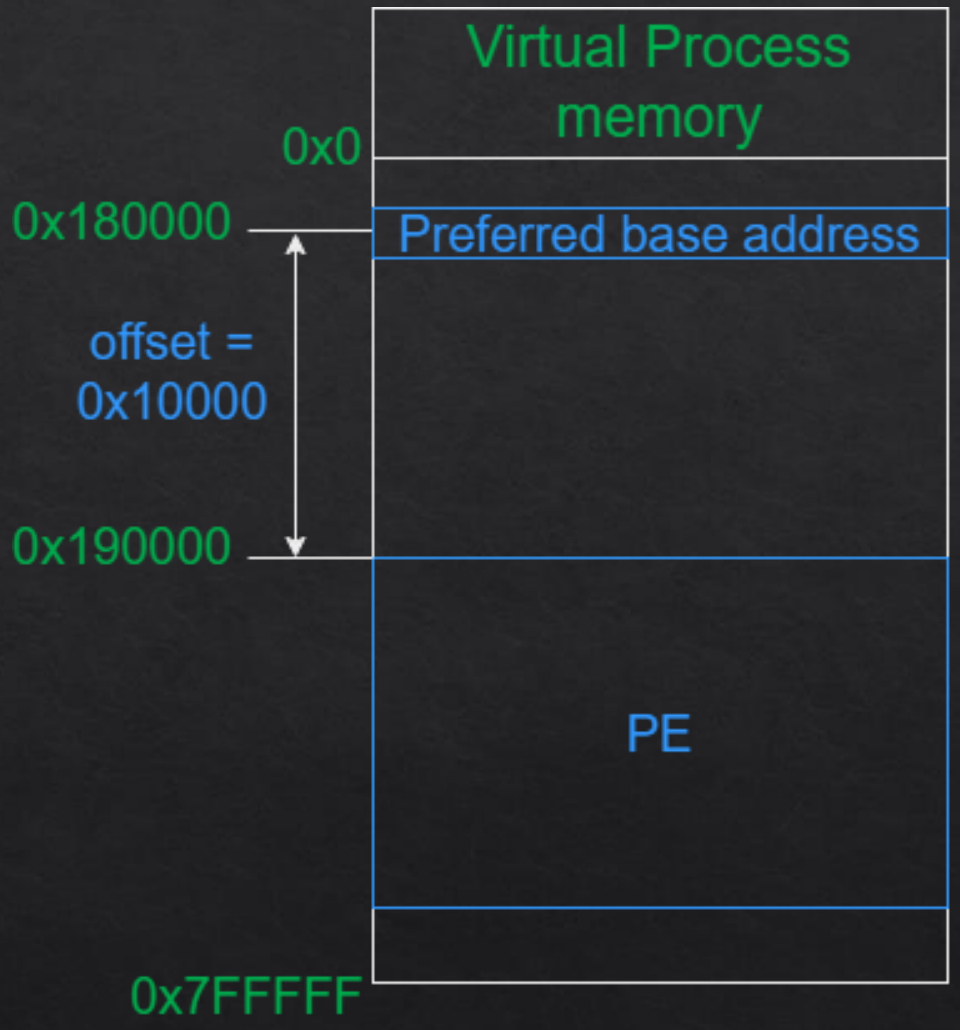
Allocation

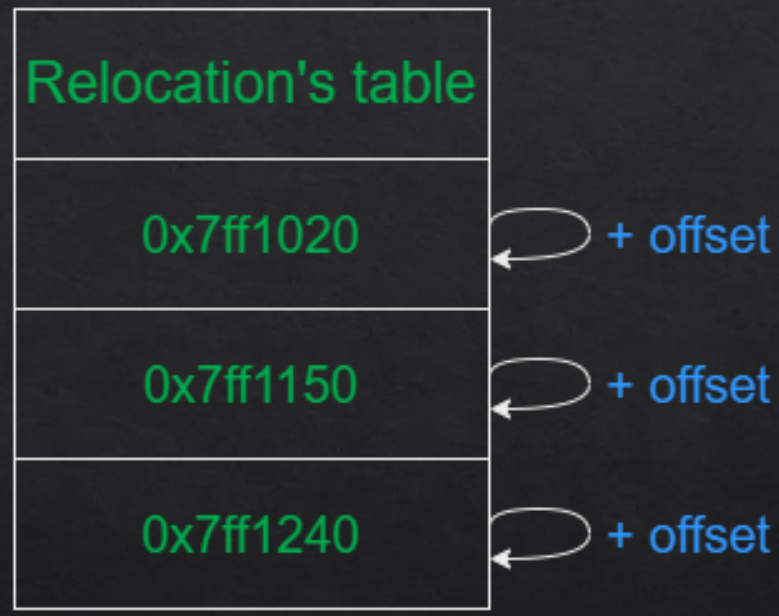

Espace d'adressage virtuel du processus
0x0
0x180000
En-têtes
.text
.idata
Allocation RW
.data
.rdata
0x230000
En-têtes
.text
.idata
Raw PE
.data
.rdata
.reloc
0x7FFFFF

Imports

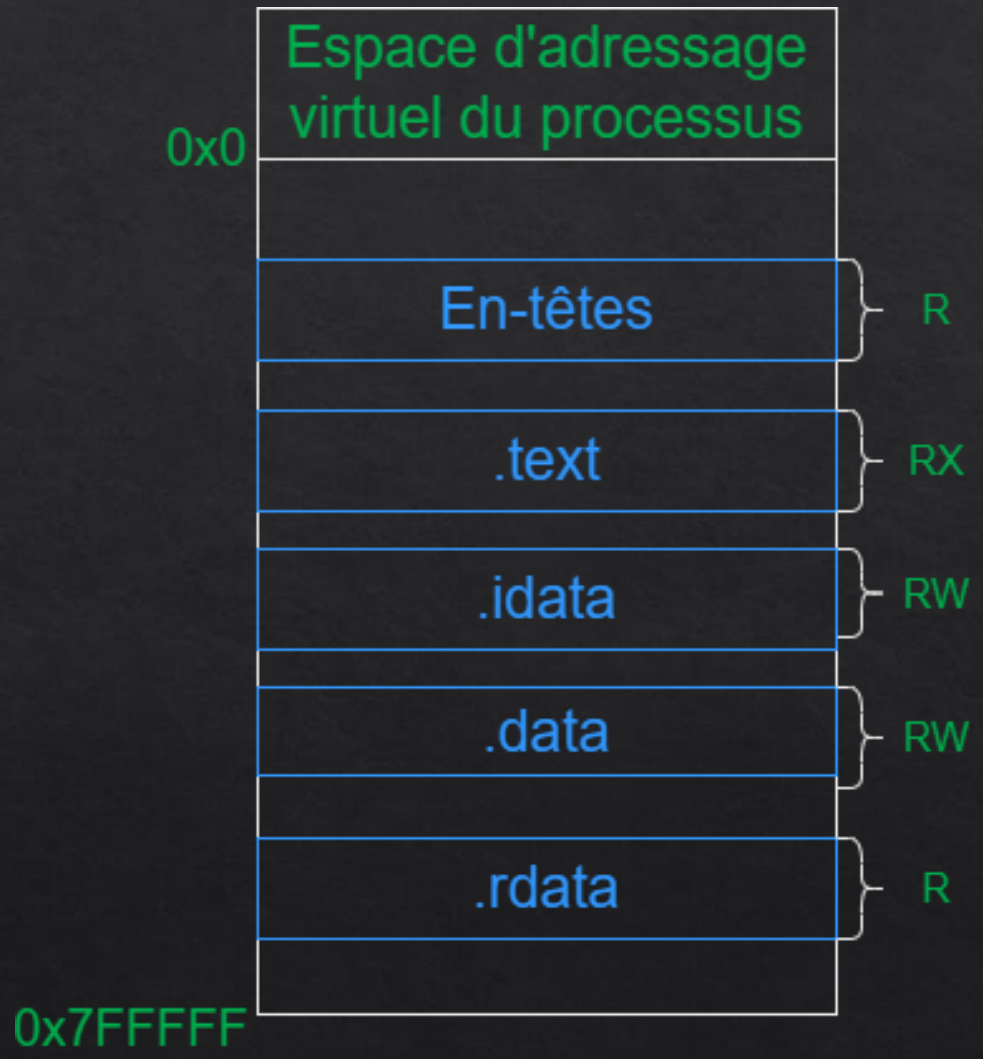

DLL en chargement
Import address Table
nom fonction
Adresse
GetProcAddress
0x7ff1020
VirtualAlloc
0x7ff1150
VirtualProtect
0x7ff1240
LoadLibrary
0x7ff1380
DLL importée
Export address Table
nom fonction
Adresse
GetProcAddress
0x7ff1020
VirtualAlloc
0x7ff1150
VirtualProtect
0x7ff1240
LoadLibrary
0x7ff1380

Relocalisations

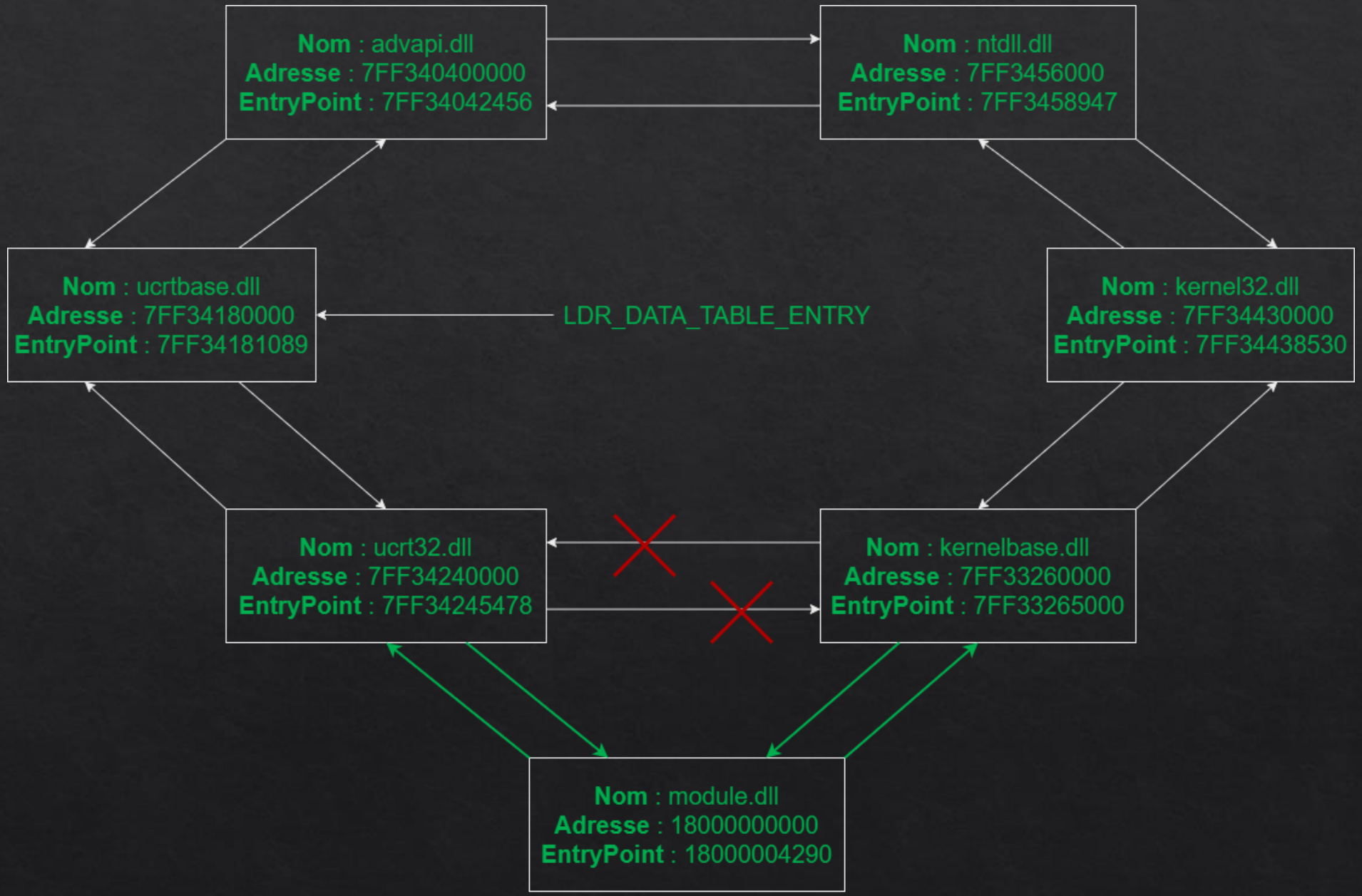




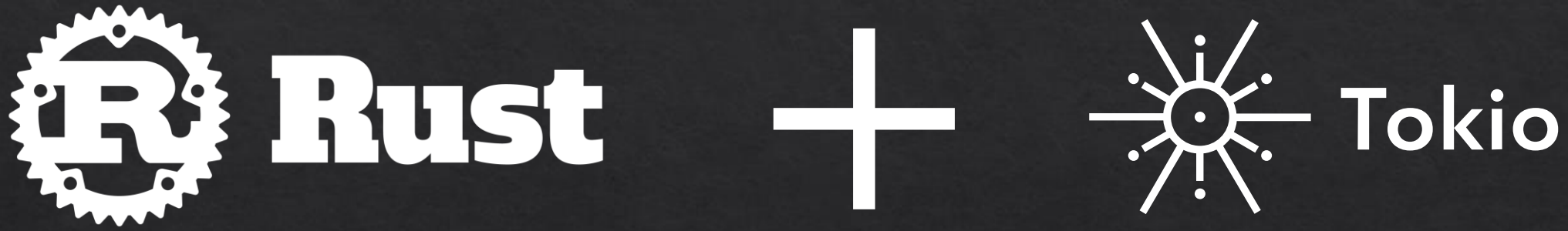
Protections



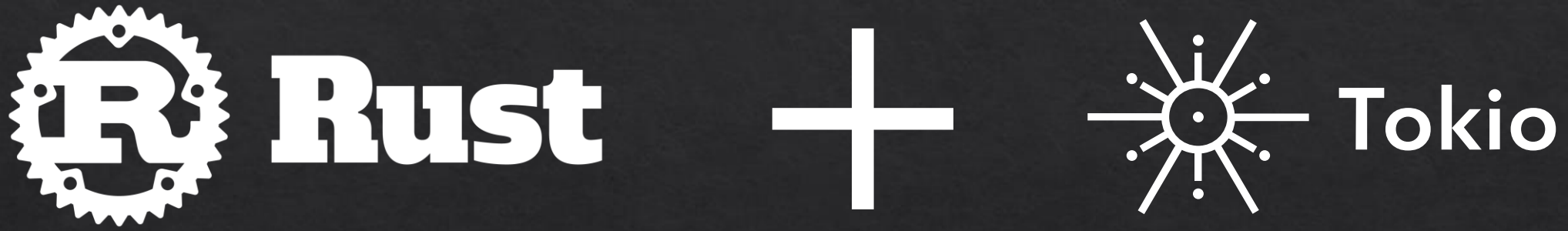
Insertion dans le PEB



Un loader qui marche du feu de Dieu, mais ...



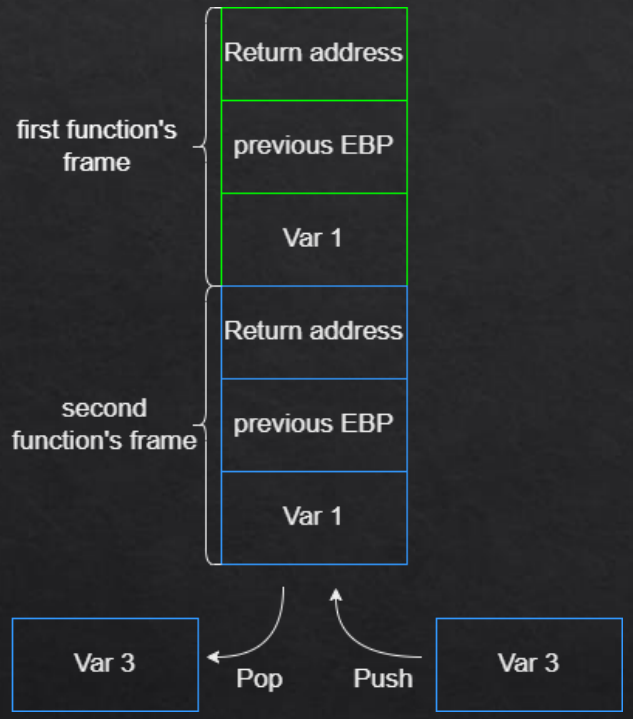
Un loader qui marche du feu de Dieu, mais ...




```
thread 'tokio-runtime-worker' has overflowed its stack
```

Qu'est-ce que le TLS

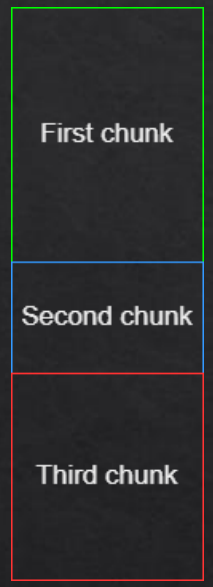
Stack




```
int a = 3;
```

- Locale à une fonction
- 1 par thread
- Allocation dynamique par l'OS

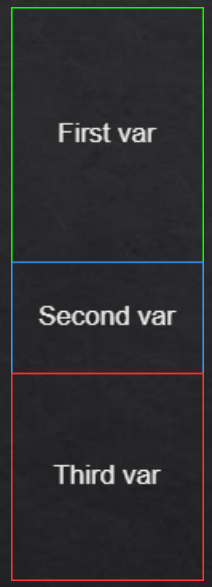
Heap




```
void *heap = malloc(10);
```

- Global au processus
- 1 par processus
- Allocation dynamique utilisateur

.data




```
int a = 3;
```

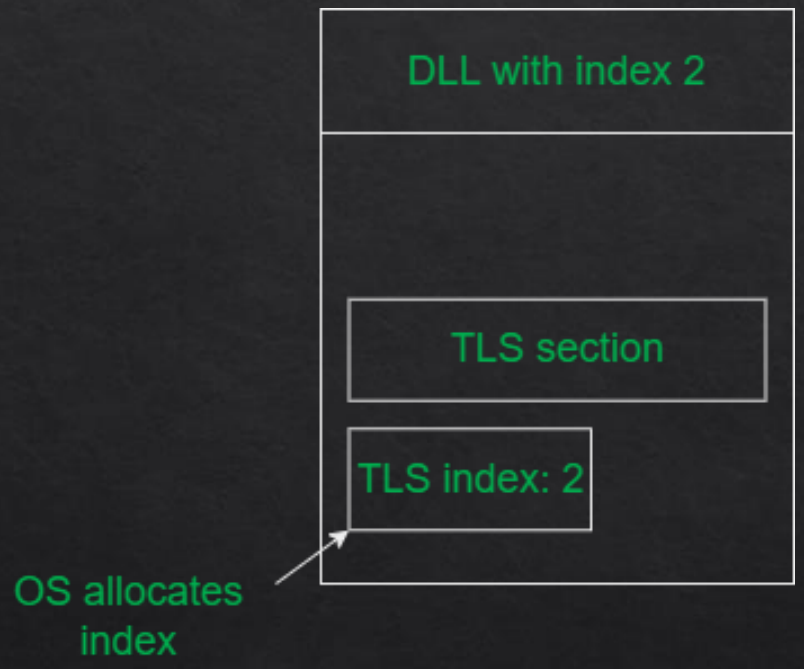
- Global au processus
- 1 par processus
- Pré alloué au sein de l'exe

TLS


```
__thread int a = 3;
__declspec( thread ) int a = 3;
```

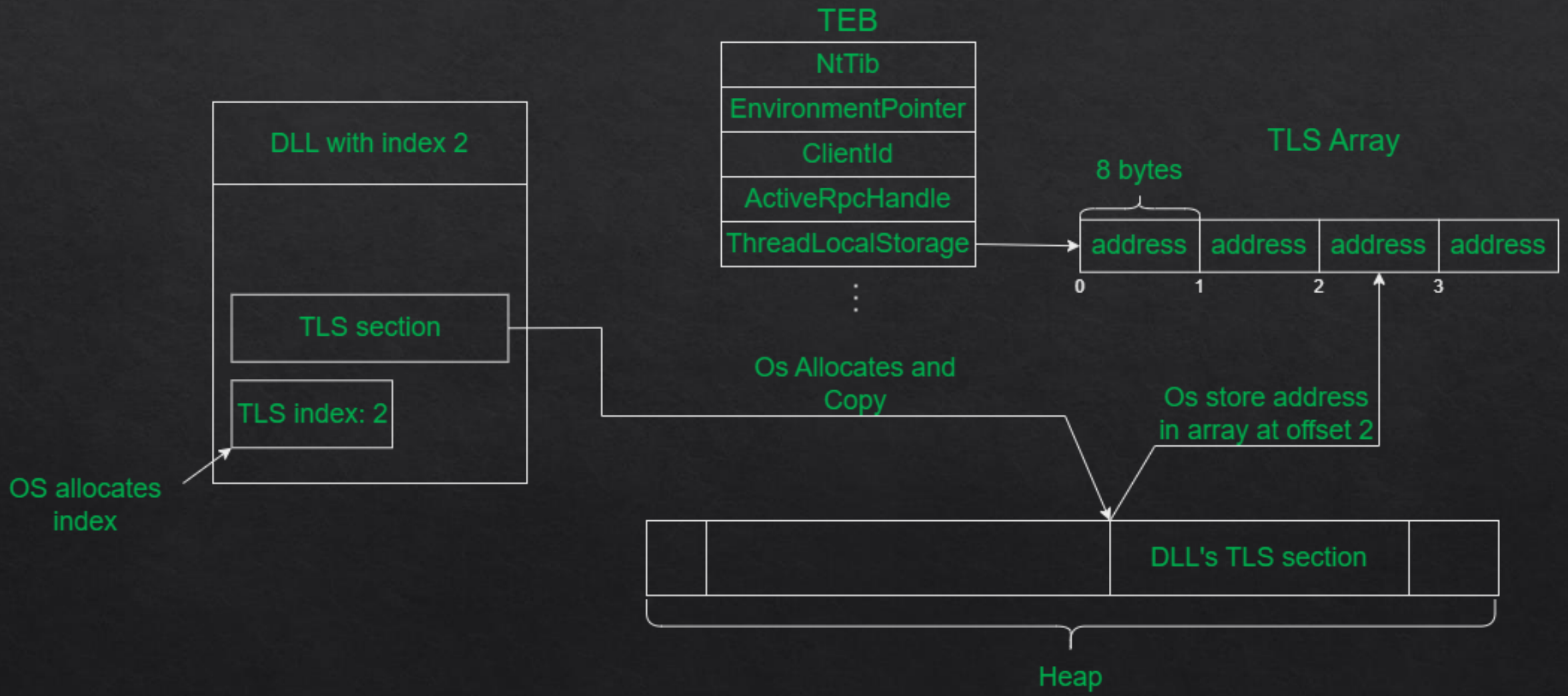
- Global à un thread
- 1 par thread
- Pré alloué au sein de l'exe + Allocation dynamique par l'OS

Comment ça fonctionne le TLS ?

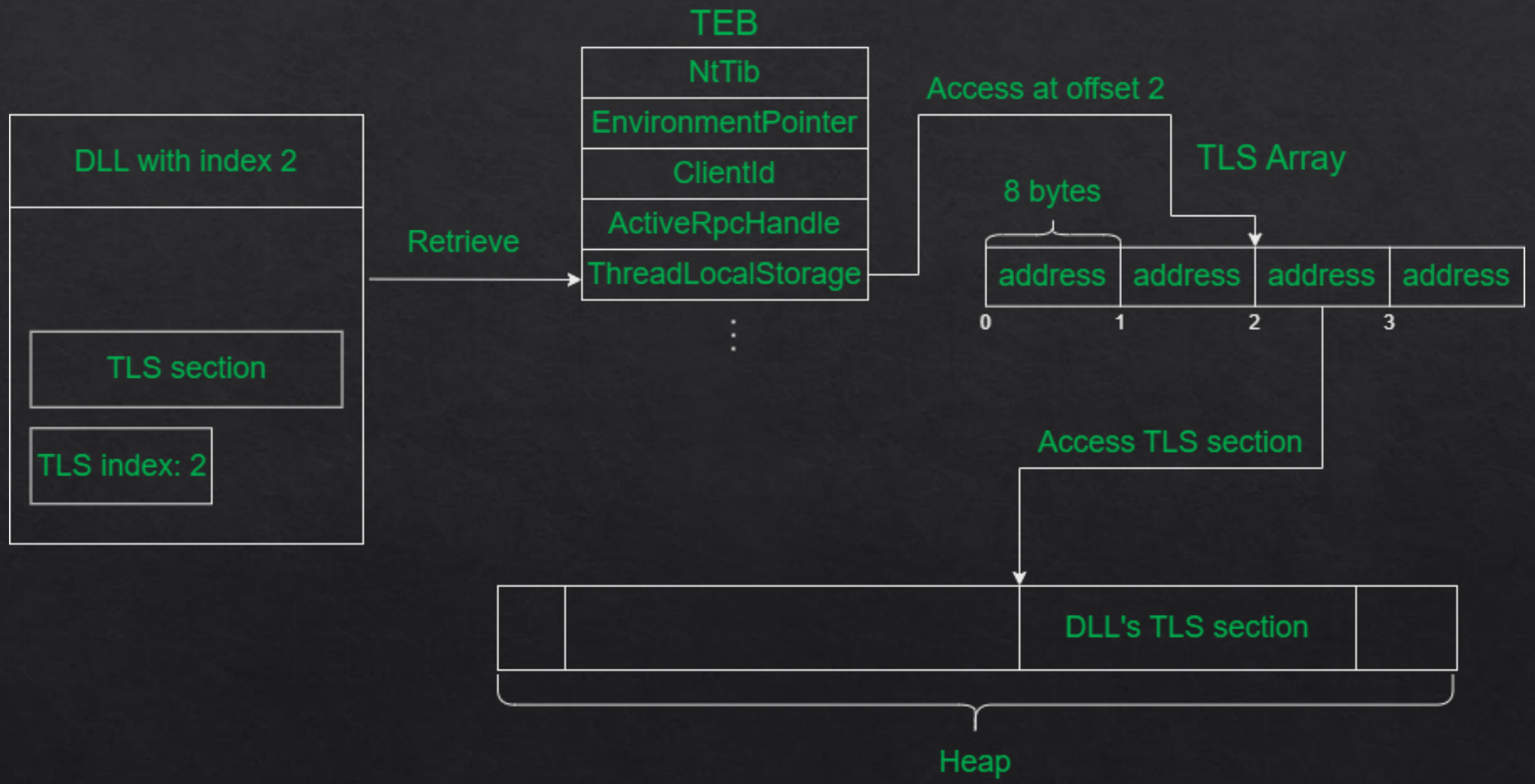


Comment ça fonctionne le TLS ?

Comment ça fonctionne le TLS ?



Cheminement complet d'un accès au TLS



Comment gérer le TLS en custom loading

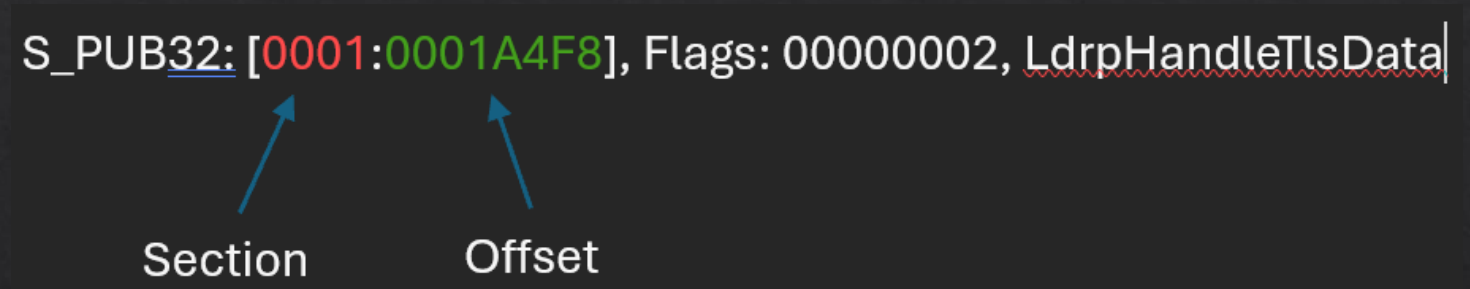
Première solution: call LdrpHandleTlsData

```
NTSTATUS LdrpHandleTlsData(LDR_DATA_TABLE_ENTRY ModuleEntry);

int main(int argc, char* argv[]) {
    long module_address = 233000;
    LDR_DATA_TABLE_ENTRY module_entry;
    module_entry.BaseAddress = module_address;

    NTSTATUS return_value = LdrpHandleTlsData(module_entry);
}
```

Problème: fonction non exportée par NTDLL

Lire les fichiers de symboles

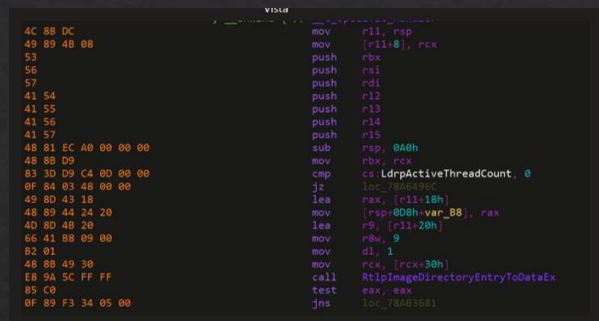
URL: https://msdl.microsoft.com/download/symbols/ntdll.pdb/4A71B509A35941A796F5B86F1AD38F531/ntdll.pdb



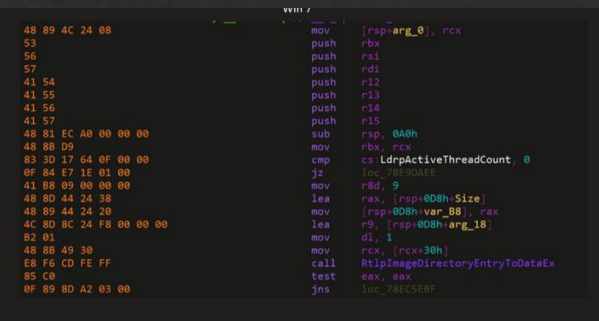
Inconvénient:

- Requête vers les serveurs de microsoft
- Téléchargement de fichier sur la machine
- Ou hardcodé chaque offset en fonction de chaque version de NTDLL (y en a beaucoup trop)

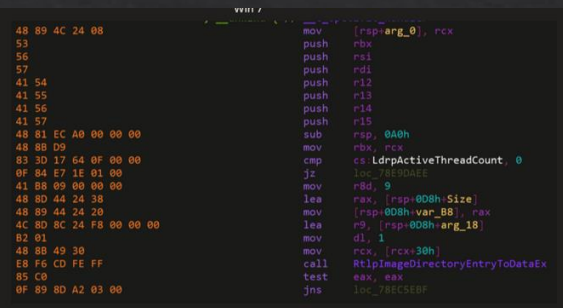
Pattern Matching



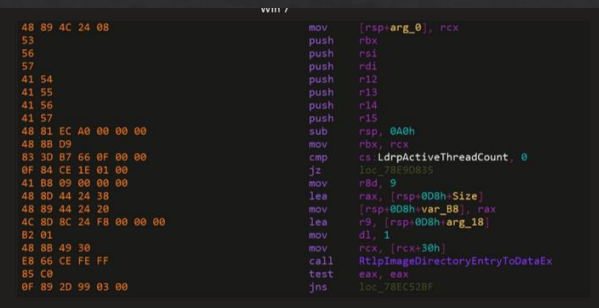
```
4C 8B DC                    mov     r11, rsp
49 89 4B 08                 mov     [r11+8], rcx
53                          push    rbx
56                          push    rsi
57                          push    rdi
41 54                       push    r12
41 55                       push    r13
41 56                       push    r14
41 57                       push    r15
48 81 EC A0 00 00 00        sub     rsp, 0A0h
48 8B D9                    mov     rbx, rcx
83 3D D9 C4 0D 00 00        cmp     cs:LdrpActiveThreadCount, 0
0F 84 03 48 00 00           jz      loc_78A6496C
49 8D 43 18                 lea     rax, [r11+18h]
48 89 44 24 20              mov     [rsp+0D8h+var_B8], rax
4D 8D 48 20                 lea     r9, [r11+20h]
66 41 B8 09 00              mov     r8w, 9
B2 01                       mov     dl, 1
48 8B 49 30                 mov     rcx, [rcx+30h]
E8 9A 5C FF FF              call    RtlpImageDirectoryEntryToDataEx
85 C0                       test    eax, eax
0F 89 F3 34 05 00           jns     loc_78AB3681
```



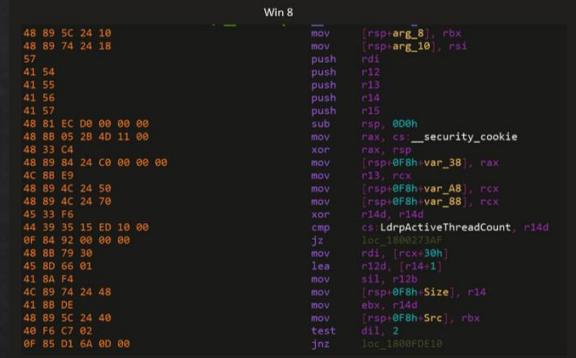
```
48 89 4C 24 08              mov     [rsp+arg_0], rcx
53                          push    rbx
56                          push    rsi
57                          push    rdi
41 54                       push    r12
41 55                       push    r13
41 56                       push    r14
41 57                       push    r15
48 81 EC A0 00 00 00        sub     rsp, 0A0h
48 8B D9                    mov     rbx, rcx
83 3D 17 64 0F 00 00        cmp     cs:LdrpActiveThreadCount, 0
0F 84 E7 1E 01 00           jz      loc_78E9DAEE
41 B8 09 00 00 00           mov     r8d, 9
48 8D 44 24 38              lea     rax, [rsp+0D8h+Size]
48 89 44 24 20              mov     [rsp+0D8h+var_B8], rax
4C 8D 8C 24 F8 00 00 00     lea     r9, [rsp+0D8h+arg_18]
B2 01                       mov     dl, 1
48 8B 49 30                 mov     rcx, [rcx+30h]
E8 F6 CD FE FF              call    RtlpImageDirectoryEntryToDataEx
85 C0                       test    eax, eax
0F 89 8D A2 03 00           jns     loc_78EC5EBF
```



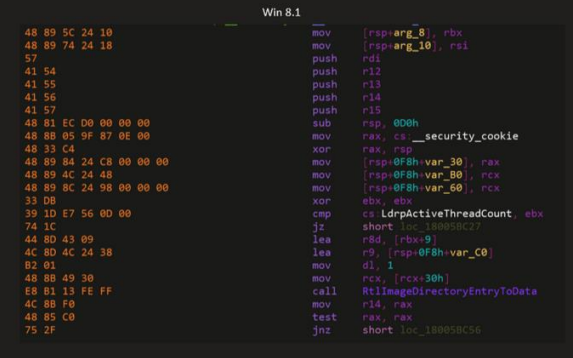
```
48 89 4C 24 08              mov     [rsp+arg_0], rcx
53                          push    rbx
56                          push    rsi
57                          push    rdi
41 54                       push    r12
41 55                       push    r13
41 56                       push    r14
41 57                       push    r15
48 81 EC A0 00 00 00        sub     rsp, 0A0h
48 8B D9                    mov     rbx, rcx
83 3D 17 64 0F 00 00        cmp     cs:LdrpActiveThreadCount, 0
0F 84 E7 1E 01 00           jz      loc_78E9DAEE
41 B8 09 00 00 00           mov     r8d, 9
48 8D 44 24 38              lea     rax, [rsp+0D8h+Size]
48 89 44 24 20              mov     [rsp+0D8h+var_B8], rax
4C 8D 8C 24 F8 00 00 00     lea     r9, [rsp+0D8h+arg_18]
B2 01                       mov     dl, 1
48 8B 49 30                 mov     rcx, [rcx+30h]
E8 F6 CD FE FF              call    RtlpImageDirectoryEntryToDataEx
85 C0                       test    eax, eax
0F 89 8D A2 03 00           jns     loc_78EC5EBF
```



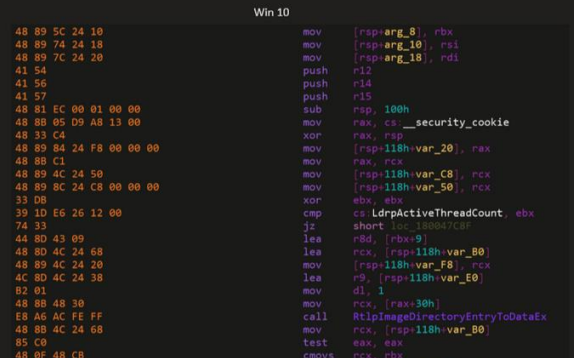
```
48 89 4C 24 08              mov     [rsp+arg_0], rcx
53                          push    rbx
56                          push    rsi
57                          push    rdi
41 54                       push    r12
41 55                       push    r13
41 56                       push    r14
41 57                       push    r15
48 81 EC A0 00 00 00        sub     rsp, 0A0h
48 8B D9                    mov     rbx, rcx
83 3D B7 66 0F 00 00        cmp     cs:LdrpActiveThreadCount, 0
0F 84 CE 1E 01 00           jz      loc_78E9D835
41 B8 09 00 00 00           mov     r8d, 9
48 8D 44 24 38              lea     rax, [rsp+0D8h+Size]
48 89 44 24 20              mov     [rsp+0D8h+var_B8], rax
4C 8D 8C 24 F8 00 00 00     lea     r9, [rsp+0D8h+arg_18]
B2 01                       mov     dl, 1
48 8B 49 30                 mov     rcx, [rcx+30h]
E8 66 CE FE FF              call    RtlpImageDirectoryEntryToDataEx
85 C0                       test    eax, eax
0F 89 2D 99 03 00           jns     loc_78EC52BF
```



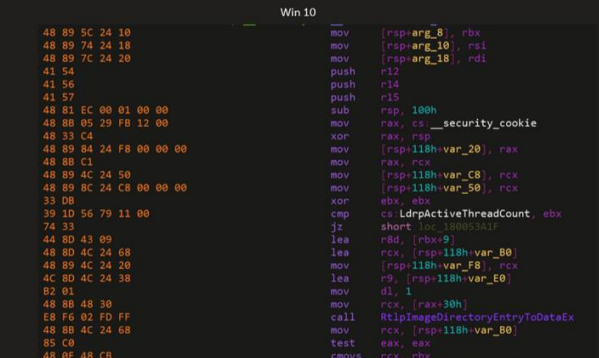
```
48 89 5C 24 10              mov     [rsp+arg_8], rbx
48 89 74 24 18              mov     [rsp+arg_10], rsi
57                          push    rdi
41 54                       push    r12
41 55                       push    r13
41 56                       push    r14
41 57                       push    r15
48 81 EC D0 00 00 00        sub     rsp, 0D0h
48 8B 05 2B 4D 11 00        mov     rax, cs:__security_cookie
48 33 C4                    xor     rax, rsp
48 89 84 24 C0 00 00 00     mov     [rsp+0F8h+var_38], rax
4C 8B E9                    mov     r13, rcx
48 89 4C 24 50              mov     [rsp+0F8h+var_A8], rcx
48 89 4C 24 70              mov     [rsp+0F8h+var_88], rcx
45 33 F6                    xor     r14d, r14d
44 39 35 15 ED 10 00        cmp     cs:LdrpActiveThreadCount, r14d
0F 84 92 00 00 00           jz      loc_1800273AF
48 8B 79 30                 mov     rdi, [rcx+30h]
45 8D 66 01                 lea     r12d, [r14+1]
41 8A F4                    mov     sil, r12b
4C 89 74 24 48              mov     [rsp+0F8h+Size], r14
41 8B DE                    mov     ebx, r14d
48 89 5C 24 40              mov     [rsp+0F8h+Src], rbx
40 F6 C7 02                 test    dil, 2
0F 85 D1 6A 0D 00           jnz     loc_1800FDE10
```



```
48 89 5C 24 10              mov     [rsp+arg_8], rbx
48 89 74 24 18              mov     [rsp+arg_10], rsi
57                          push    rdi
41 54                       push    r12
41 55                       push    r13
41 56                       push    r14
41 57                       push    r15
48 81 EC D0 00 00 00        sub     rsp, 0D0h
48 8B 05 9F 87 0E 00        mov     rax, cs:__security_cookie
48 33 C4                    xor     rax, rsp
48 89 84 24 C8 00 00 00     mov     [rsp+0F8h+var_30], rax
48 89 4C 24 48              mov     [rsp+0F8h+var_B0], rcx
48 89 8C 24 98 00 00 00     mov     [rsp+0F8h+var_60], rcx
33 DB                       xor     ebx, ebx
39 1D E7 56 0D 00           cmp     cs:LdrpActiveThreadCount, ebx
74 1C                       jz      short loc_18005BC27
44 8D 43 09                 lea     r8d, [rbx+9]
4C 8D 4C 24 38              lea     r9, [rsp+0F8h+var_C0]
B2 01                       mov     dl, 1
48 8B 49 30                 mov     rcx, [rcx+30h]
E8 B1 13 FE FF              call    RtlImageDirectoryEntryToData
4C 8B F0                    mov     r14, rax
48 85 C0                    test    rax, rax
75 2F                       jnz     short loc_18005BC56
```



```
48 89 5C 24 10              mov     [rsp+arg_8], rbx
48 89 74 24 18              mov     [rsp+arg_10], rsi
48 89 7C 24 20              mov     [rsp+arg_18], rdi
41 54                       push    r12
41 56                       push    r14
41 57                       push    r15
48 81 EC 00 01 00 00        sub     rsp, 100h
48 8B 05 D9 A8 13 00        mov     rax, cs:__security_cookie
48 33 C4                    xor     rax, rsp
48 89 84 24 F8 00 00 00     mov     [rsp+118h+var_20], rax
48 8B C1                    mov     rax, rcx
48 89 4C 24 50              mov     [rsp+118h+var_C8], rcx
48 89 8C 24 C8 00 00 00     mov     [rsp+118h+var_50], rcx
33 DB                       xor     ebx, ebx
39 1D E6 26 12 00           cmp     cs:LdrpActiveThreadCount, ebx
74 33                       jz      short loc_180047C8F
44 8D 43 09                 lea     r8d, [rbx+9]
48 8D 4C 24 68              lea     rcx, [rsp+118h+var_B0]
48 89 4C 24 20              mov     [rsp+118h+var_F8], rcx
4C 8D 4C 24 38              lea     r9, [rsp+118h+var_E0]
B2 01                       mov     dl, 1
48 8B 48 30                 mov     rcx, [rax+30h]
E8 A6 AC FE FF              call    RtlpImageDirectoryEntryToDataEx
48 8B 4C 24 68              mov     rcx, [rsp+118h+var_B0]
85 C0                       test    eax, eax
48 0F 48 CB                 cmovs   rcx, rbx
48 89 4C 24 68              mov     [rsp+118h+var_B0], rcx
48 85 C9                    test    rcx, rcx
75 31                       jnz     short loc_180047CC0
```

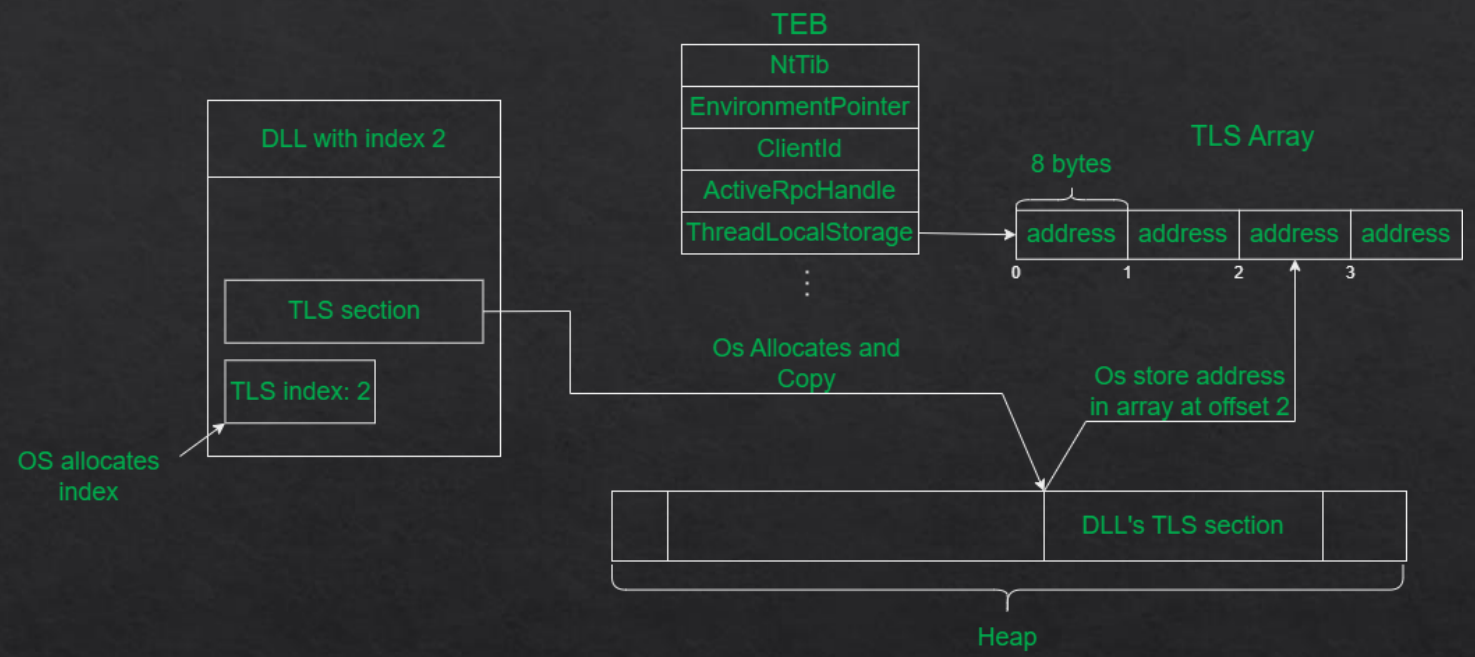


```
48 89 5C 24 10              mov     [rsp+arg_8], rbx
48 89 74 24 18              mov     [rsp+arg_10], rsi
48 89 7C 24 20              mov     [rsp+arg_18], rdi
41 54                       push    r12
41 56                       push    r14
41 57                       push    r15
48 81 EC 00 01 00 00        sub     rsp, 100h
48 8B 05 29 FB 12 00        mov     rax, cs:__security_cookie
48 33 C4                    xor     rax, rsp
48 89 84 24 F8 00 00 00     mov     [rsp+118h+var_20], rax
48 8B C1                    mov     rax, rcx
48 89 4C 24 50              mov     [rsp+118h+var_C8], rcx
48 89 8C 24 C8 00 00 00     mov     [rsp+118h+var_50], rcx
33 DB                       xor     ebx, ebx
39 1D 56 79 11 00           cmp     cs:LdrpActiveThreadCount, ebx
74 33                       jz      short loc_180053A1F
44 8D 43 09                 lea     r8d, [rbx+9]
48 8D 4C 24 68              lea     rcx, [rsp+118h+var_B0]
48 89 4C 24 20              mov     [rsp+118h+var_F8], rcx
4C 8D 4C 24 38              lea     r9, [rsp+118h+var_E0]
B2 01                       mov     dl, 1
48 8B 48 30                 mov     rcx, [rax+30h]
E8 F6 02 FD FF              call    RtlpImageDirectoryEntryToDataEx
48 8B 4C 24 68              mov     rcx, [rsp+118h+var_B0]
85 C0                       test    eax, eax
48 0F 48 CB                 cmovs   rcx, rbx
48 89 4C 24 68              mov     [rsp+118h+var_B0], rcx
48 85 C9                    test    rcx, rcx
75 31                       jnz     short loc_180053A50
```

Inconvénient:

- La fonction change souvent donc Pattern à maintenir dans le temps
- Beaucoup de pattern à hardcodé dans le loader

Deuxième solution: réimplémenté à la main



Inconvénient:

- Avoir un moyen d'exécuter du code à chaque exécution de thread
- Se battre avec LdrpHandleTlsData (état interne de la fonction)

Callback sur les threads: DllMain

4 raisons:

- DLL_PROCESS_ATTACH: Chargement dans le processus
- DLL_THREAD_ATTACH: Nouveau thread
- DLL_THREAD_DETACH: Thread terminé
- DLL_PROCESS_DETACH: Déchargement du processus

Comparaison avec les autres solutions

Critère \ Technique	LdrpHandleTlsData Symbol	LdrpHandleTlsData Pattern	Manuel
Appatition dans le PEB	Non	Non	Oui
Compatibilité Windows version	Simple	Compliqué	Compliqué
Online / Téléchargement	Oui	Non	Non
Stabilité	Très stable	Très stable	Pas stable

Questions ?