# Livre blanc

## Partie 4

Stratégies et Techniques pour échapper aux Antivirus modernes et contourner les EDR


NBS System
Groupe CELESTE

# SOMMAIRE

# TECHNIQUES DE CONTOURNEMENT DE HOOKING SUR LA NTDLL

Nous pourrons contourner ces mesures de sécurité en utilisant des techniques de « **Unhooking** » afin de masquer nos activités malveillantes et éviter la détection.

L'intérêt du Unhooking pour les attaquants est de **masquer leurs activités malveillantes** en retirant les Hooks (ou crochets) mis en place par les EDRs pour les surveiller.

Les Hooks sont des points d'entrée dans les processus qui permettent aux EDR d'intercepter les appels de fonctions et de surveiller les activités suspectes. En retirant ces hooks, les attaquants peuvent masquer leur présence et éviter la détection par les EDRs.

**Les attaquants utilisent différentes techniques pour effectuer le « Unhooking ».**

# 1. UNHOOKING NTDLL DEPUIS LE DISQUE

Après avoir compris le fonctionnement de la technique de Hooking utilisée par les EDRs sur la NTDLL, nous avons décidé d'implémenter une technique de contournement en chargeant une copie fraîche de la NTDLL depuis le disque.

> Cette technique consiste à charger une copie non modifiée de la NTDLL dans l'espace d'adressage de notre processus, évitant ainsi les Hooks installés par les antivirus et EDR sur la version chargée en mémoire

Pour mettre en œuvre cette technique, nous avons commencé par écrire un code en C++ pour charger la NTDLL depuis le disque dur, en utilisant les fonctions **CreateFileMapping()** et **MapViewOfFile()**.

Nous avons ensuite modifié les appels système dans notre code pour utiliser les fonctions de la NTDLL chargée depuis le disque plutôt que celles de la version potentiellement modifiée en mémoire.

# 1. UNHOOKING NTDLL DEPUIS LE DISQUE

```
hFile = CreateFile((LPCSTR) sNtdllPath, GENERIC_READ, FILE_SHARE_READ, NULL, OPEN_EXISTING, 0, NULL);

hFileMapping = CreateFileMappingA_p(hFile, NULL, PAGE_READONLY | SEC_IMAGE, 0, 0, NULL);

pMapping = MapViewOfFile_p(hFileMapping, FILE_MAP_READ, 0, 0, 0);

result = UnhookNtdll(GetModuleHandle((LPCSTR) sNtdll), pMapping);
```

*Figure 10 - Unhooking de la NTDLL depuisle disque (CreateFileMapping)*

# 1. UNHOOKING NTDLL DEPUIS LE DISQUE

Nous avons ensuite réalisé des tests afin d'évaluer l'efficacité de cette technique. Nous avons comparé les résultats obtenus avec cette technique à ceux issus d'autres méthodes de contournement , ainsi qu'à une exécution normale du code sans aucun contournement.

Les résultats ont montré que cette technique de contournement était **efficace pour éviter les Hooks** de la NTDLL installés par les solutions antivirus et EDR.

Cependant, nous avons également rencontré des difficultés lors de l'implémentation de cette technique, notamment en raison de la complexité de l'architecture de la NTDLL et de la nécessité de prendre en compte les différentes versions du système d'exploitation ainsi que les différences entre les processeurs 32 bits et 64 bits.

Nous avons également dû faire face à des défis liés à l'intégration et à la compatibilité de notre solution avec divers environnements clients.

# 1. UNHOOKING NTDLL DEPUIS LE DISQUE

En conclusion, l'implémentation dela technique de contournement de Hooking de laNTDLL en chargeant une fraîche copie de la NTDLL depuisle disque a été une étape importante de notre projet de développement de techniques d'évasion. Bien que cette technique ait montré son efficacitédans certains cas, elle nécessite une compréhension approfondie du fonctionnement de la NTDLL et peut être bloquée par certaines solutions antivirus et EDR.

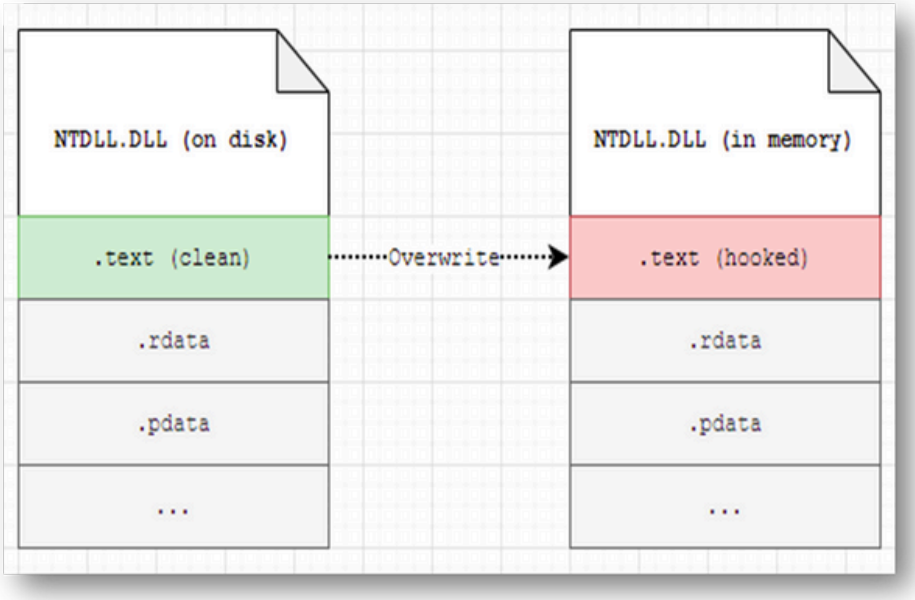


*Figure 11 - Unhooking de la NTDLL depuisle disque*

# 2. UNHOOKING NTDLL DEPUIS UN PROCESSUS SUSPENDU

La technique précédente qui consiste à charger en mémoire une NTDLL depuis le disque présente une limite majeure, car chaque processus a déjà une NTDLL dans son espace d'adressage, ce qui peut donc susciter la suspicion des antivirus et EDRs.

> Pour contourner cette limite, nous avons exploré d'autres méthodes pour charger une copie fraîchede la NTDLL en mémoire sans toucher l'originale sur le disque. Après des recherches sur le chargement des processus en mémoire, nous avons découvert qu'il était possible d'obtenir une NTDLL en mémoire sans Hooks en lançant un processus avec l'état suspendu.

Cette approche permet de court-circuiter le processus de Hooking des EDRs sur la NTDLL. Nous pouvons alors lancer un processus quelconque en mode suspendu, copier sa NTDLL en mémoire et remplacer la version hookée de la NTDLL dans le processus cible pour contourner les Hooks de l'antivirus ou de l'EDR.

# 2. UNHOOKING NTDLL DEPUIS UN PROCESSUS SUSPENDU

Pour mettre en œuvre cette technique, nous avons utilisé la fonction **CreateProcess** de Windows **en spécifiant le flag CREATE_SUSPENDED** dans les options de création du processus. Cette fonction renvoie un Handle du processus créé.

```
BOOL result = CreateProcessA(
    NULL,
    (LPSTR)"cmd.exe",
    NULL,
    NULL,
    FALSE,
    CREATE_SUSPENDED | CREATE_NEW_CONSOLE,
    NULL,
    "C:\\Windows\\System32\\",
    &si,
    &pi);
```

*Figure 12 - Création de processus suspendu (Unhooking)*

# 2. UNHOOKING NTDLL DEPUIS UN PROCESSUS SUSPENDU

Une fois que le processus est suspendu, nous pouvons accéder à sa mémoire et extraire la NTDLL en utilisant la fonction **GetModuleHandle** avec le nom de module **"ntdll.dll"**. Cette fonction renvoie un Handle du module en mémoire.

Ensuite, nous copions la NTDLL en mémoire dans un buffer en utilisant la fonction **memcpy**. Comme le processus est suspendu, il n'y a pas suffisamment de temps pour que les EDRs puissent injecter leurs Hooks dans la NTDLL. Nous obtenons ainsi une copie fraîche de la NTDLL en mémoire, sans Hooks. Nous pouvons alors utiliser cette copie pour remplacer la version hookée de la NTDLL dans notre processus malveillant.

Enfin, en utilisant cette copie non-hookée de la NTDLL, nous sommes alors en mesure **d'exécuter des actions malveillantes sans être détecté par les EDRs ou les antivirus**.

Il est important de noter que cette technique peut être détectée par les EDRs si le processus suspendu reste en état suspendu pendant une période prolongée, ce qui peut indiquer une activité suspecte. Il est donc recommandé de relancer rapidement le processus cible après avoir effectué les modifications nécessaires

# 2. UNHOOKING NTDLL DEPUIS UN PROCESSUS SUSPENDU

En résumé, l'implémentation de cette technique implique le lancement d'un processus en mode suspendu, l'extraction de sa NTDLL en mémoire, la copie et l'utilisation de cette NTDLL non hookéedans notre processus malveillant. Cette technique peut contourner les hooks de l'antivirus ou de l'EDR,mais elle peut être détectéesi elle est utilisée de manière prolongée.

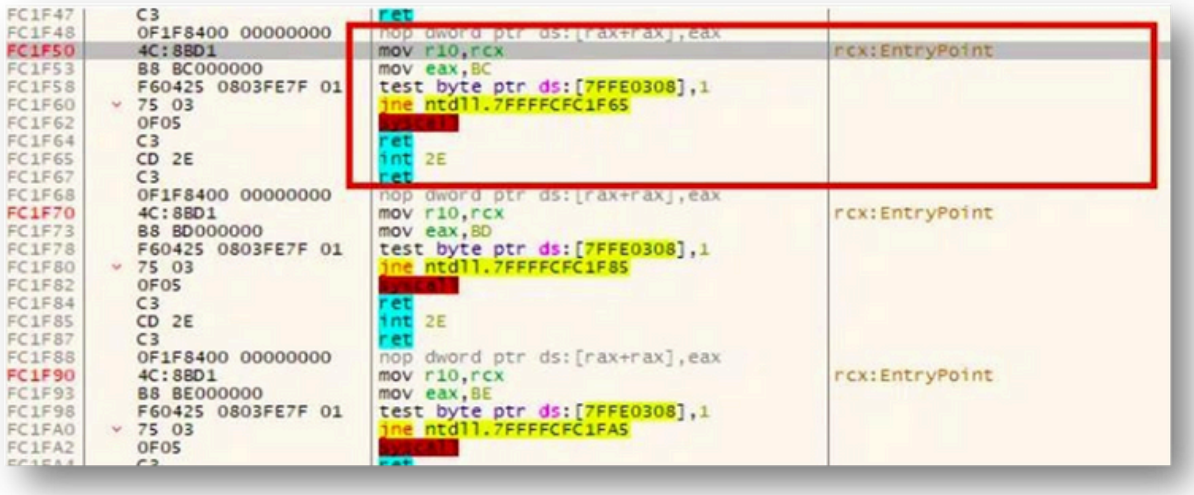


*Figure 12 - Absence de hooks dans les fonctions de la NTDLL d'un process suspendu via xdbg*

# 2. UNHOOKING NTDLL DEPUIS UN PROCESSUS SUSPENDU

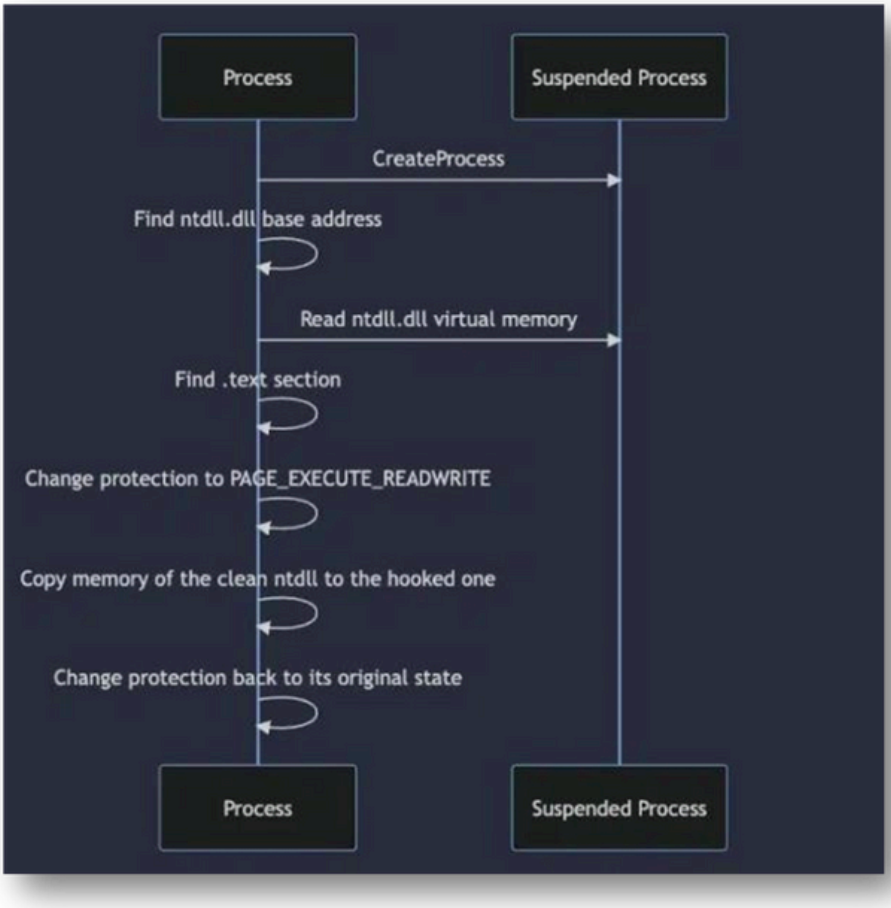


*Figure 12 - Unhooking de la NTDLL depuisun processus suspendu*