De l'importance de la SSDT [INTRO] – TECH

Comme tout système d'exploitation, Windows possède certaines fonctions systèmes propres au noyau. Ce sont les appels systèmes, utilisées par les fonctions API de Windows. En effet, les API ne sont qu'une interface avec le noyau pour les programmes utilisateurs. Comme exemple, l'appel de la fonction API CreateFile engendre une cascade d'appels de fonction. Lorsqu'un programme nécéssite un accés disque via l'appel CreateFile, cette API initialement exportée par la bibliothèque Kernel32.dll va par la suite faire appel à la fonction NtCreateFile exportée, elle, par NtDll.dll. Ces deux appels de fonction successifs sont réalisés dans le User Land.

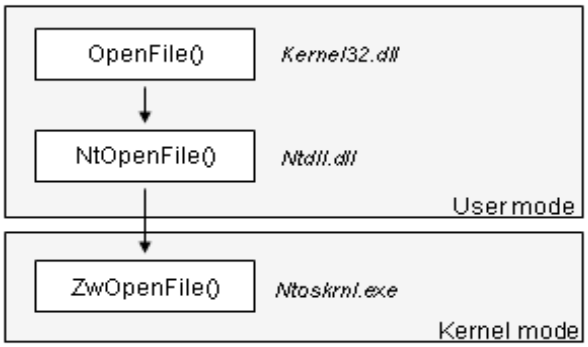
Les appels systèmes interviennent à ce niveau-là. En effet, la transition API / Syscall va s'effectuer lors de l'exécution de l'API native NtCreateFile. Le rôle de l'API native (NtCreateFile dans ce cas) est de faire appel à une interruption spécifique à Windows, afin que la fonction noyau correspondante soit appelée ; elle effectue donc un appel système. On peut situer le code permettant cette transition en analysant la fonction NtCreateFile dans NtDll.dll.

- Désassemblage de ZwCreateFile dans NtDll.dll (IDA Pro) :

```
public ZwCreateFile
ZwCreateFile proc near
mov     eax, 25h          ; numéro de service (offset) pour la SSDT
mov     edx, 7FFE0300h
call    dword ptr [edx]   ; appel de l'interruption (INT 2E)
retn    2Ch
ZwCreateFile endp
```

Ici, on a désassemblé ZwCreateFile, en effet NtCreateFile et ZwCreateFile sont toutes les deux exportées par NtDll.dll et pointent sur les mêmes instructions. On voit que ZwCreateFile stocke la valeur 25h dans le registre EAX, puis par la suite effectue un appel de fonction, à l'adresse stockée dans [7FFE0300]. En réalité, cette instruction CALL est une interruption logicielle nécéssaire à l'appel de la fonction noyau qui gère les accés disques.

En effet, lors de l'interruption logicielle, le processeur va rechercher dans l'IDT (Interrruption Descriptor Table) le gestionnaire d'interruption correspondant à celle-ci. Dans notre cas, ce gestionnaire d'interruption se basera sur le registre EAX, dans lequel ZwCreateFile a stocké l'offset pour la SSDT. En effet, il relèvera dans la SSDT (qui est un tableau de DWORD) un pointeur de fonction correspondant au service d'accés disque. Toutes APIs natives possède cette forme, la différence réside dans la valeur qu'elles stockent dans le registre EAX, pour faire appel au bon service. Le graphique représente la cascade d'appels pour l'appel d'une API.



Ce mécanisme d'appel aux fonctions noyau est similaire à celui mise en place dans les systèmes linux. En effet, ils possèdent une interruption (la 0x80), qui permet de lancer des fonctions systèmes du noyau linux,

pour cela il suffit de placer une valeur également dans le registre EAX, correspondant à la fonction système souhaitée (telle que sys_execve, sys_open, sys_fork etc...).

L'adresse de la SSDT est obtenue par l'intermédiaire d'un tableau de ServiceDescriptorEntry, lui-même contenu dans la structure ServiceDescriptorTable. En effet, son adresse correspond à SDT.ServDescEntry[0].ServiceTable. L'adresse ce premier membre du tableau de structures ServiceDescriptorEntry peut-être obtenue par un symbole exporté par ntoskrnl.exe, le noyau du système Windows, le symbole KeServiceDescriptorTable, de là, on a directement accés au tableau de DWORD ; la SSDT.

L'intégrité de cette SSDT est donc très importante, car le bon fonctionnement du système, des fonctions APIs en dépend. C'est sur cela que se base les rootkits, en effet ils modifients la SSDT de façon à détourner les appels systèmes, de cette manière ils peuvent falsifier les données retournées par ces fonctions, dans le but de cacher leur activité (invisibilité de processus, de fichier / dossiers, de ports tcp / udp, etc...). Un logiciel anti-rootkit vérifie donc l'intégrité des entrées de la SSDT (donc les différents membres du tableau SDT.ServDescEntry[0].ServiceTable) à la recherche de d'entrées falsifiés (par d'éventuels rootkits).

Cependant, de fausses alertes peuvent être soulevées, car en effet, les pare-feu (entre autres) utilisent le SSDT hooking afin de maintenir une protection et une surveillance.

Différents outils peuvent vérifier l'intégrité de cette SSDT dont KProCheck, et aussi Vice.
Voici aussi une documentation intéressante : http://3psilon.info/Les-rootkits.html.

En conclusion, une modification des entrées de la SSDT pour détourner les appels systèmes est une méthode bas niveau puissante, utilisées par les logiciels rootkits et logiciels de solution de sécurité. Cependant, cette technique n'est pas indétectable, une vérification de la SSDT permet de détecter ces détournements.

Prochainement, je travaillerai sur une source d'un driver illustrant ce billet.

Me contacter : touron.guillaume@gmail.com
Microsoft Student Partner