Windows Vista™

Kernel Security

Matthieu Suiche | < matt@msuiche.net > | www.msuiche.net

Qui suis-je?

- Lycéen ☺ (le bac j'y crois !)
- Website/Blog: www.msuiche.net
- Microsoft Student Partner (MSP)
- Security Fanatics !
  - Reverse Engineering
  - Analyse de vulnérabilités
  - Recherche sur des binaires malicieux
  - Programmation d'outils liés à la sécurité
  - Recherche sur le Noyau des O.S. Windo
- TinyKRNL Project
  - Kernel Developer (ATAPI)

Agenda

- Kernel Hooking, pourquoi faire ?
- Patchguard
- Code Integrity
- Drivers signés
- Windows Vista (noyau 32 bits)
  - SSDT
  - KIDT
  - MSR
- Windows Vista (noyau 64 bits)
  - SSDT
  - KIDT
  - MSR

Kernel Hooking, pourquoi faire ?

- Essentiellement des rootkits !
  - Modification de tables système comme la SDT
    - Fonctions NtCreateProcess, NtSystemInformation, ...
  - Modification de structures internes
    - PsLoadedModuleList
  - Modification de l’IDT pour détourner l’usage des débogueurs
  - Modification de l'interruption 0x2E pour intercepter les syscalls (Win2K)
  - Modification des registres MSRs pour intercepter les syscalls (WinXP)
  - Modification du prologue des fonctions système

Patchguard

- Auteurs : Windows Core Team
- Introduit dans Windows 2003 x64
  - Cf. analyse de Matt Miller & Ken Johnson
- Vérifie les tables et zones sensibles de l'O.S.
    - Fonctions
    - IDT
    - GDT
    - SDT
    - Liste des processus
    - MSRs
- 25, Octobre 2006 – annonce Authentium
- 8, Novembre 2006 – Windows Vista RTM

Code Integrity (CI.DLL)

- Auteurs : Windows DRM Team
- Nouveauté de Windows Vista
- Plusieurs étapes
  - Un bootloader vérifie l’authenticité du code du noyau, HAL, et drivers lancés au démarrage
  - Vérification de l'intégrité de l'import table de NTOSKRNL.EXE
  - Suppression ou altération de CI.DLL => impossible de démarrer la machine
  - Remarque : WINLOAD.EXE vérifie également l'intégrité de NTOSKRNL.EXE au démarrage
- Actif avant PatchGuard
- Désactivable par l'utilisateur

Drivers signés (KMD)

- Objectif : empêcher l'installation d'un driver non contrôlé
- Obligatoire dans Vista 64 bits
- Code signé par un certificat
- Désactivable au démarrage (option BOOT.INI)

- Juillet 2006 – J. Rutkowska / BlackHat
  - Attaque en 3 étapes
    - Consommer toute la mémoire physique
    - Accéder au fichier de pagination (pagefile.sys) par accès direct au disque (\\.\PHYSICALDRIVE0)
    - Patcher un driver existant
  - Autre attaque utilisant la virtualisation matérielle
    - Pacifica (AMD SVM extensions) / Vanderpool (Vt-x)
- Attaque sur pagefile.sys corrigée depuis Vista RC2

On s’était donné rendez vous dans 10bytes, même offset, même kernel ...

Windows Vista 32 bits

System Service Descriptor Table

- Objectif : retrouver la SSDT

- La méthode décrite par "90210" est toujours valable
  - KeServiceDescriptorTable est toujours exporté

  - KiServiceTable est initialisée dans KiInitSystem()
    - mov ds:_KeServiceDescriptorTable, offset _KiServiceTable

  - Importer KeServiceDescriptorTable
  - Lister les références
  - Chercher celle qui correspond à "mov [mem32], imm32"

  - On obtient bien un pointeur vers
    PVOID KiServiceTable[KiServiceLimit]

Interrupt Descriptor Table

- Objectif : retrouver l'IDT

- Toujours pareil aussi !
  - Preuve de concept : IDTGuard 0.1
    - Publié le 10 décembre 2006 ☺

  - Chercher la fonction exportée KiSystemStartup()
  - GetMachineBootPointers() retourne les offsets de IDT, GDT et LDT
  - NTOSKRNL manipule alors les interruptions
    - Ajout du pointeur vers l'IDT dans KPCR.IDT (+0x38)
    - Copie des adresses initiales depuis la section INIT
      - mov edi, [ebp+IdtEntry]
      - mov esi, offset INIT.IdtRawOffset
      - mov ecx, 2048
      - shr ecx, 2
      - rep movsd
  - Certaines de ces interruptions sont ensuite modifiées par HAL.DLL (KPCR), d'autres sont des pointeurs sur KINTERRUPT

Memento : Sysenter !

```
KiFastSystemCall proc near
        mov     edx, esp
        sysenter
KiFastSystemCall endp
```

Model Specific Registers

- Opcodes : SYSENTER / SYSRET
- Initialisation de 3 MSRs :
  - IA32_SYSENTER_ESP
    - Pointeur de la pile en kernel-land
  - IA32_SYSENTER_CS
    - Registre CS pour le code kernel-land
  - IA32_SYSENTER_EIP
    - Point d'entrée du code après un sysenter
- KiLoadFastSyscallMachineSpecificRegisters()
  - WRMSR(IA32_SYSENTER_CS, 0x08, NULL);
  - WRMSR(IA32_SYSENTER_EIP, KiFastCallEntry, NULL);
  - WRMSR(IA32_SYSENTER_ESP, Unknow.u1988, NULL);
- L'initialisation dispose d'une signature très facilement identifiable

Conclusion 32 bits

- Même organisation que sous Windows 2000/XP/2003

- Les mêmes outils devraient continuer à fonctionner
  - T. Chew Keong - SDTRestore v0.2
  - M. Suiche – IDTGuard v0.1
  - M. Russinovich – Rootkit Revealer v1.71
  - J. Rutkowska - System Virginity Verifier (SVV) v2.3

Je suis kernel et je le reste, dans le coding et dans le geste...

Windows Vista 64 bits

System Service Descriptor Table

- KeServiceDescriptorTable où es-tu ?
  - Le symbole n’est plus exporté mais toujours dans la section ALMOSTRO
  - Tout est fait dans INIT.KiInitSystem()
    - lea rax, KiServiceTable
    - mov cs:KeServiceDescriptorTable, rax
  - KiServiceTable est toujours dans la section ".text"
  - Une signature de code plus grande est nécessaire
  - Une localisation manuelle de KiInitSystem() est nécessaire
    - L’utilisation d’un LDE 64 bits serait préférable
    - (LDE = Length Disassembly Engine)

Interrupt Descriptor Table

- KiSystemStartup()
  - Initialisation du segment GS (GS_BASE)
  - Copie de la base de l’IDT dans [GsBase+0x38]
  - KiInitializeBootStructures()
    - xor r10, r10
    - lea r12, (INIT.KiInterruptInitTable+8)
    - lea r9, KxUnexpectedInterrupt0
  - Copie des interruptions depuis NTOSKRNL
    - 0 à 19 plus quelques autres
  - L'IDT est plus facile à trouver que la SSDT
    - Elle peut être une cible pour le hooking

Memento : Syscall !

```
Ntxxxxxxxxxxxxx proc near
                 mov     r10, rcx ; Ntxxxxxxxxxxxxx
                 mov     eax, FunctionID
                 syscall
                 retn
Ntxxxxxxxxxxxxx endp
```

Syscall / Sysexit

- IA32_LSTAR (0xC0000082)
  - KiSystemCall64
- IA32_CSTAR (0xC0000083)
  - KiSystemCall32

- KiInitializeBootStructures()
  - lea rax, KiSystemCall32
  - mov ecx, 0C0000083h
  - mov rdx, rax ; CSTAR
  - shr rdx, 20h
  - wrmsr

  - lea rax, KiSystemCall64
  - mov ecx, 0C0000082h ; LSTAR
  - mov rdx, rax
  - shr rdx, 20h
  - wrmsr

- Conclusion : l'initialisation des MSRs est facilement identifiable

Conclusion 64 bits

- Réalisation d'un PatchGuard 64 bits non Microsoft ?
  - Aucune recherche publique sur le sujet
  - Mais un article en préparation de mon côté
- Remarque :
  - L'émulation WoW (Windows-on-Windows) continue à utiliser l'interruption 0x2E

Références

- Matthew Conover (2006), Windows Vista Kernel Mode Security
  - http://www.symantec.com/avcenter/reference/Windows_Vista_Kernel_Mode_Security.pdf
- Matthieu Suiche (Décembre 2006) IDTGuard v0.1 PublicBuild
  - http://www.msuiche.net/?p=9
- Joanna Rutkowska (Juillet/Aout 2006), Subverting Vista Kernel
  - http://invisiblethings.org/papers/joanna%20rutkowska%20-%20subverting%20vista%20kernel.ppt
- Mark Russinovich (Novembre 2006), RootkitRevealer 1.7.1
  - http://www.microsoft.com/technet/sysinternals/utilities/RootkitRevealer.mspx
- Joanna Rutkowska (2006) “System Virginity Verifier”
  - http://www.invisiblethings.org/papers/rutkowska_bhfederal2006.ppt
- Authentium (Octobre 2006), Microsoft Patchguard
  - http://blogs.authentium.com/sharp/?p=12
- Matt Miller, Ken Johnson (Décembre, 2005) Bypassing Patchguard on Windows x64
  - http://www.uninformed.org/?v=3&a=3
- Protected-Mode Exceptions and Interrupts (5-3)
  - IA-32 Intel Architecture Software Developer's Manual. System Programming Guide
- Microsoft (Janvier 2006), “Digital Signatures for Kernel Modules on x64-based Systems Running Windows Vista”
  - http://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/x64KMSigning.doc
- Microsoft (Avril 2005), “Benefits of Microsoft Windows x64 Editions”
  - http://download.microsoft.com/download/D/A/A/DAA7245D-E01D-46A4-AB70-3A95ED3F6934/Windowsx64BenefitsWP.doc
- M. Conover (Mars 2006), “Analysis of the Windows Vista Security Model”
  - http://www.symantec.com/avcenter/reference/Windows_Vista_Security_Model_Analysis.pdf

Questions and Answers