Forensics mémoire sous Windows

Nicolas RUFF
EADS-IW SE/CS
nicolas.ruff (à) eads.net

Plan

- Introduction
- Le côté offensif
- Principes
- Collecte
- Analyse
- Exemples
- Contre-mesures
- Conclusion
- Bibliographie
- Remerciements

Introduction

- Qui suis-je ?
  - Un "ingénieur / chercheur / consultant / expert" en sécurité
  - Occasionnellement un analyste *forensics*
    - Mais il y a tellement peu d'attaques ☺
- Pourquoi cette présentation ?
  - Un état de l'art sur une discipline en pleine évolution
  - Des idées de recherche
  - La création d'une communauté de gens intéressés par :
    - Le partage d'information
    - Des outils "gratuits"

Le côté offensif

- L'intrusion est le thème majeur de toutes les conférences de "sécurité"
  - L'intrusion "tout en mémoire" est un sous-domaine déjà traité également
  - Quelques exemples
    - François Gaspard, Samuel Dralet : "corruption de la mémoire lors de l'exploitation" [SSTIC'06]
    - Metasploit Anti Forensics Project

Le côté offensif

- Les outils d'intrusion sont disponibles "sur étagère"
  - Metasploit (Framework, Meterpreter, IRB)
  - Immunity (CANVAS, Hydrogen)
  - Core SDI (Core Impact)
  - Cain (Abel)
  - *Etc.*
  - + outils privés

Le côté offensif

- *exec("/bin/sh")* n'a plus de sens
  - N'en a jamais eu sous Windows
    - Avec en plus les *firewalls* personnels, les HIPS, …
  - N'en a pas non plus sous *nix
    - chroot, SELinux, …

- Objectifs d'un *shell* moderne
  - Communications chiffrées (avec PFS)
  - Binaire chiffré (avec PFS)
    - Ex. BurnEye avec mot de passe
    - Ou mieux : aucun binaire laissé sur la machine !

Principes

- Plusieurs écoles
  - "*Dead forensics*" : on débranche la prise
    - La méthode historique
    - Souvent la seule méthode "officielle"
  - "*Live forensics*" : on lance des outils dans le système cible
    - Préconisée par Microsoft depuis le rachat de Sysinternals
  - La voie du milieu (nouveau !)
    - On collecte la mémoire physique, puis on débranche la prise
    - Avantage sur le *dead forensics* :
      - On récupère les informations dynamiques (ex. clés)
    - Avantage sur le *live forensics* :
      - La contre-expertise est possible

Collecte

- Il n'existe qu'une seule méthode "parfaite"
  - La pause dans la machine virtuelle
    - Seule méthode permettant d'obtenir un état cohérent
- Les autres méthodes affectent la cible
  - Carte d'acquisition dédiée
    - Prérequis important
    - Technologie jeune et peu répandue
    - Contournable (!) – cf. travaux de Joanna Rutkowska
  - Bus FireWire
    - Passe par des *hacks* – peu fiable
    - Ne permet pas d'accéder à toute la mémoire (ex. UMA)

Collecte

- Accès à "\Device\PhysicalMemory"
  - Nécessite un driver à partir de Windows XP SP2 / Windows 2003
  - Contournable également (technique des *Split TLB*)
  - "nc | dd" peut s'avérer (très) long …
- *CrashDump*
  - Accessible par plusieurs méthodes : clavier, EMS, *driver* …
  - Sauvegarde en plus l'état du processeur
  - Mais ne fonctionne pas dans certains cas particuliers (ex. RAM > 4 Go)
  - Le *CrashDump* remplace le fichier d'échange
    - Mais il est possible de copier le fichier d'échange auparavant

Collecte

- L'hibernation
  - Une piste à creuser …
  - Mais la mémoire non allouée ne s'y trouve probablement pas
- Le *Hard Reset* (!)
  - De nombreuses études tendent à démontrer que la mémoire physique est relativement "stable" avec le temps
    - Ex. 85% de la mémoire est conservée après 10h d'activité
  - La mémoire serait même conservée à froid !

Collecte

- Configuration des méthodes de collecte
  - La plupart des paramètres peuvent être changés "à chaud"
    - En particulier grâce à WMI
    - Se reporter au papier pour plus de détails ☺
- Dans tous les cas on se retrouve avec :
  - La mémoire physique
  - Jusqu'à 16 fichiers d'échange
  - Parfois aucune information sur l'état du processeur
  - Et le tout dans un état potentiellement incohérent …

Analyse

- Principe des outils connus
  - "Objectif CR3"
    - La mémoire virtuelle de chaque processus est un gruyère de pages physiques
    - 1 Go de RAM = 262 144 pièces de 4 Ko
    - La clé : la table de pages du processus (*PDE*)
      - Pointée par le registre CR3
    - Le mode protégé x86 en une ligne
      - PDE -> PTE
        - -> page physique (si Valid = 1)
        - -> fichier d'échange (si Valid =0)

Analyse

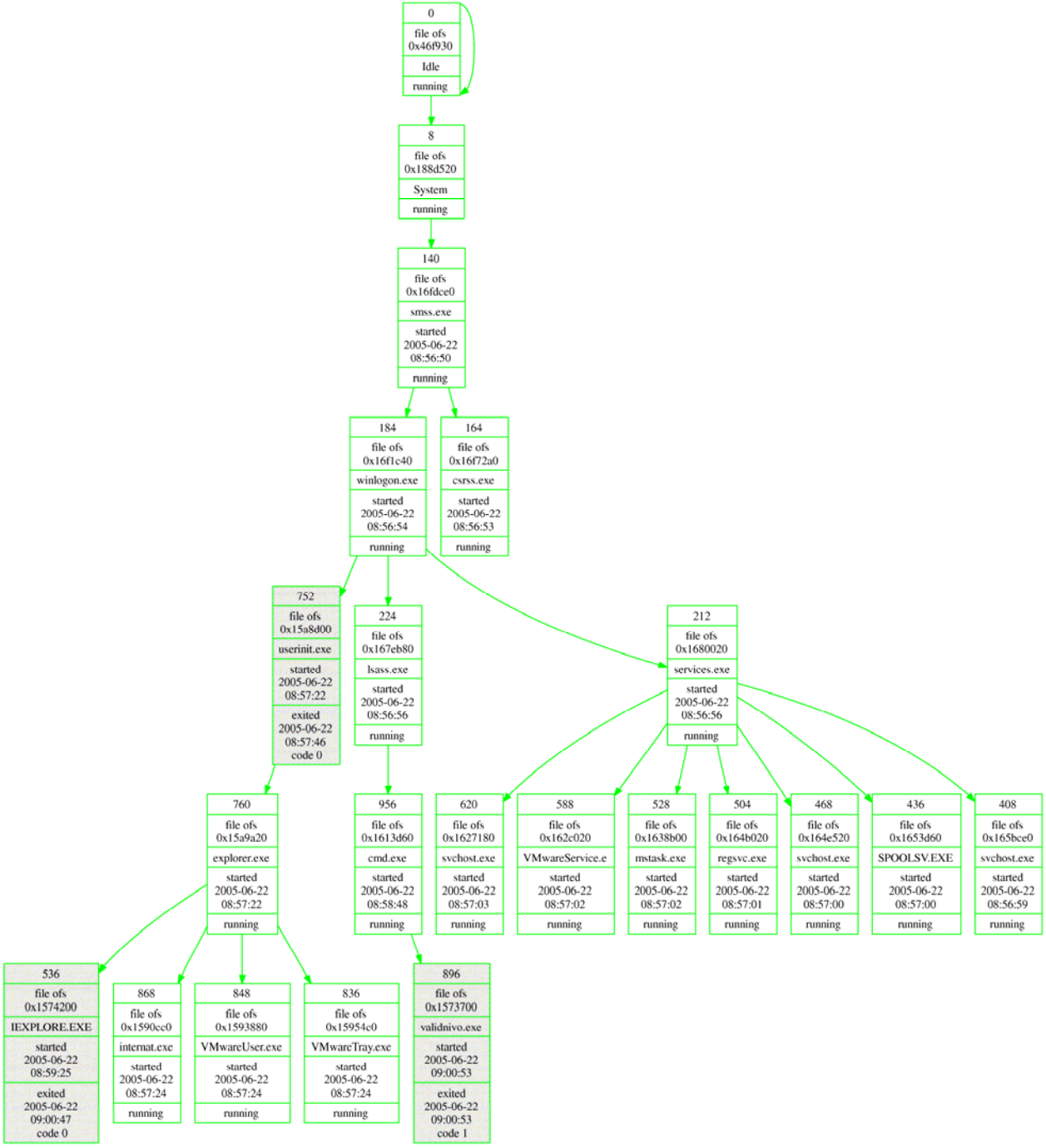
- Principe général = la recherche de motifs
  - Quelques structures importantes
    - Surtout EPROCESS (contient CR3)
    - Mais également les structures réseau, etc.
  - Semi-documentées
    - Grâce aux symboles du noyau utilisables dans WinDbg
  - Potentiellement modifiées à chaque Service Pack !
    - D'où un effort initial important

Exemples

- L'outil "PTFinder"
  - Auteur : Andreas Schuster
  - Historiquement l'un des premiers outils publics
    - Challenge DFRWS 2005
  - Permet de reconstruire la liste des processus
    - Et leur espace d'adressage virtuel
    - Y compris les processus terminés !
- D'autres outils sont venus par la suite
  - MemParser (Chris Betz)
  - VolaTools
  - KnTTools (payant)
  - *Etc.*

Exemples

- Challenge Securitech 2005
  - Auteur : Kostya Kortchinsky
  - "Cas d'école" pour PTFinder
    - Un *snapshot* VMWare est fourni
    - Le fichier d'échange est désactivé
  - Objectifs
    - Analyser l'attaque
    - Récupérer et relancer un processus terminé


0
file ofs 0x46f930
Idle
running
8
file ofs 0x188d520
System
running
140
file ofs 0x16fdce0
smss.exe
started 2005-06-22 08:56:50
running
184
file ofs 0x16f1c40
winlogon.exe
started 2005-06-22 08:56:54
running
164
file ofs 0x16f72a0
csrss.exe
started 2005-06-22 08:56:53
running
752
file ofs 0x15a8d00
userinit.exe
started 2005-06-22 08:57:22
exited 2005-06-22 08:57:46
code 0
224
file ofs 0x167eb80
lsass.exe
started 2005-06-22 08:56:56
running
212
file ofs 0x1680020
services.exe
started 2005-06-22 08:56:56
running
760
file ofs 0x15a9a20
explorer.exe
started 2005-06-22 08:57:22
running
956
file ofs 0x1613d60
cmd.exe
started 2005-06-22 08:58:48
running
620
file ofs 0x1627180
svchost.exe
started 2005-06-22 08:57:03
running
588
file ofs 0x162c020
VMwareService.e
started 2005-06-22 08:57:02
running
528
file ofs 0x1638b00
mstask.exe
started 2005-06-22 08:57:02
running
504
file ofs 0x164b020
regsvc.exe
started 2005-06-22 08:57:01
running
468
file ofs 0x164e520
svchost.exe
started 2005-06-22 08:57:00
running
436
file ofs 0x1653d60
SPOOLSV.EXE
started 2005-06-22 08:57:00
running
408
file ofs 0x165bce0
svchost.exe
started 2005-06-22 08:56:59
running
536
file ofs 0x1574200
IEXPLORE.EXE
started 2005-06-22 08:59:25
exited 2005-06-22 09:00:47
code 0
868
file ofs 0x1590cc0
internat.exe
started 2005-06-22 08:57:24
running
848
file ofs 0x1593880
VMwareUser.exe
started 2005-06-22 08:57:24
running
836
file ofs 0x15954c0
VMwareTray.exe
started 2005-06-22 08:57:24
running
896
file ofs 0x1573700
validnivo.exe
started 2005-06-22 09:00:53
exited 2005-06-22 09:00:53
code 1

Exemples

- Détection du Meterpreter
  - Principe : injection de DLL "tout en mémoire"
    - LoadLibrary() est une API *userland*
    - Les fonctions de NTDLL.DLL utilisées sont :
      - NtOpenSection()
      - NtQueryAttributesFile()
      - NtOpenFile()
      - NtCreateSection()
      - NtMapViewOfSection()
    - => émulation de ces fonctions en *userland*

Exemples

- Détection
  - Parcourir la liste des DLLs chargées (dans le *PEB*)
    - En fait 3 listes (!)
  - Détecter par nom / signature
  - Ou corréler mémoire / disque
- Attention
  - Les *hooks* NTDLL.DLL sont retirés après usage
  - Il serait possible de se retirer des listes de DLLs
  - Il serait possible d'effacer l'entête PE en mémoire
  - … mais toute *thread* doit avoir un quota d'exécution !
    - (Ou pas : utilisation des exceptions)

Contre-mesures

- Les contre-mesures sont infinies …
  - … dès que l'attaquant connaît les points de contrôle du défenseur
  - Cf. Rootkits vs. Anti-rootkits
- Attaques "matérielles"
  - Rootkits de boot (ex. BootKit de eEye)
  - Rootkits matériels (ex. firmware Tigon2)
  - Fonctions des chipsets
    - Reprogrammation NorthBridge, code SMM, code ACPI, virtualisation, ...

Contre-mesures

- Attaques "logicielles"
  - *Hook* des points d'accès à la mémoire
    - Ex. "\Device\PhysicalMemory", IoAllocateMdl(), …
  - *Split TLBs*
  - Threads injectées de manière non évidente
    - Ex. à la fin de la section de code d'un processus
  - Virtualisation logicielle (ex. segmentation)
  - … et d'autres qui restent à inventer !

Conclusion

- L'analyse mémoire …
  - … n'est pas fiable à 100%
    - Problème de la collecte
    - Problème des heuristiques de reconstruction
  - … n'est qu'un complément des techniques antérieures
    - Corréler les informations mémoire avec le disque
- Les outils disponibles sont :
  - Chers (bienvenue dans le monde du *forensics*)
  - Et/ou limités (ex. WinDbg)
  - Et/ou difficiles à configurer (ex. adresses dépendantes du Service Pack)

Conclusion

- Mais l'analyse mémoire …
  - … est totalement indispensable aujourd'hui !
    - Détection des intrusions "tout en mémoire"
      - Ex. Meterpreter
    - Détection des modifications *runtime*
      - Ex. NtAccessCheck()
    - Récupération de clés volatiles
      - Ex. clés Diffie-Hellmann, mot de passe TrueCrypt, …

Bibliographie

- Auteurs
  - Andreas Schuster
  - Mariusz Burdach
  - George M. Garner
  - Aaron Walters
  - Harlan Carvey

  - … et tant d'autres …
    - Cf. papier

Remerciements

- Les experts *forensics*
  - Alexandre Garaud
  - Samuel Dralet (Lexfo)
  - Laurent Dupuy (FreeSecurity)

- Les autres
  - L'équipe EADS-IW SE/CS
  - Agnès Ruff

Questions ?