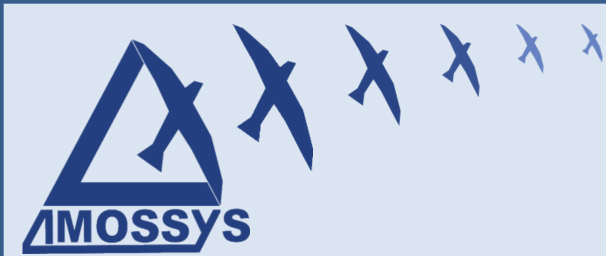
AMOSSYS

Windows 10 VSM
Présentation des nouveautés et implémentations

Guillaume C. ~ contact@amossys.fr

But...but...
VSM
...I don't know what puzzles is
LOLCATS.COM



Sommaire

- VSM, késako ?
  - Description des fonctionnalités
  - Comment l'activer
- VSM en pratique
  - Rappels sur Hyper-V
  - Principes généraux du VSM
  - Secure Kernel Mode
  - Isolated User Mode
  - Communications dans le Secure Mode
  - Communications entre NTOS et le Secure Mode
  - Hypothèses d'attaques
- Conclusion

VSM, késako ?

- Virtual Secure Mode
- Architecture de sécurité basée sur Hyper-V

- Permet l'isolation des partitions (VMs)

- Présentation basée sur le travail de @aionescu, c'est à lui et aux équipes de Microsoft que revient tout le mérite !

- Quelques memes pour vous aider à supporter

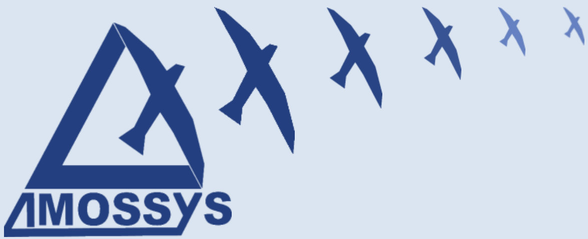
Description des fonctionnalités

- Device Guard, lutte contre les programmes malveillants
  - UMCI/KMCI : Vérification de l’intégrité du code en mode utilisateur/noyau
  - Réglage fin des politiques de CI
- Credential Guard, isolation des identifiants Windows
  - Isolation du processus LSASS côté Secure Mode
- UEFI, Secureboot, TPM
- Guarded Fabric et vTPM

Comment l'activer

- Sur un ordinateur supportant TPM et UEFI
- Installer Windows 10 Entreprise
- Installer la fonctionnalité Hyper-V
- Ajouter le système à un domaine Active Directory déployé sur un Windows Server 2016
- Activer le paramètre : Computer Configuration / System / Device Guard / Turn on Virtualization Based Security
- Ajouter l'option de démarrage suivante :
    - bcdedit /set vsmlaunchtype auto
- http://deploymentresearch.com/Research/Post/490/Enabling-Virtual-Secure-Mode-VSM-in-Windows-10-Enterprise-Build-10130

Parés ?



VSM en pratique

- VSM se base sur Hyper-V
- L'hyperviseur permet de créer la partition « root » (hôte) avec un droit d'exécution non privilégié
- Deux kernels sont exécutés :
  - SKM, environnement sécurisé, gestion des secrets : SMART Kernel ou SK
  - Root partition, normal mode, NTOS non privilégié
- => L'hyperviseur ne fait pas confiance au système hôte

Rappels sur Hyper-V

- Hyperviseur basé en ring-1
  - Hvix64.exe pour Intel VMX
  - Hvax64.exe pour AMD VT-d
- Le cœur d'Hyper-V ne contient aucune logique métier et est petit (~1300Ko)
  - => Surface d'attaque réduite
- Gestion des partitions (VMs)
- Gestion des contextes d'exécutions
- Gestion de la mémoire (EPT)
- Virtualisation de l'APIC

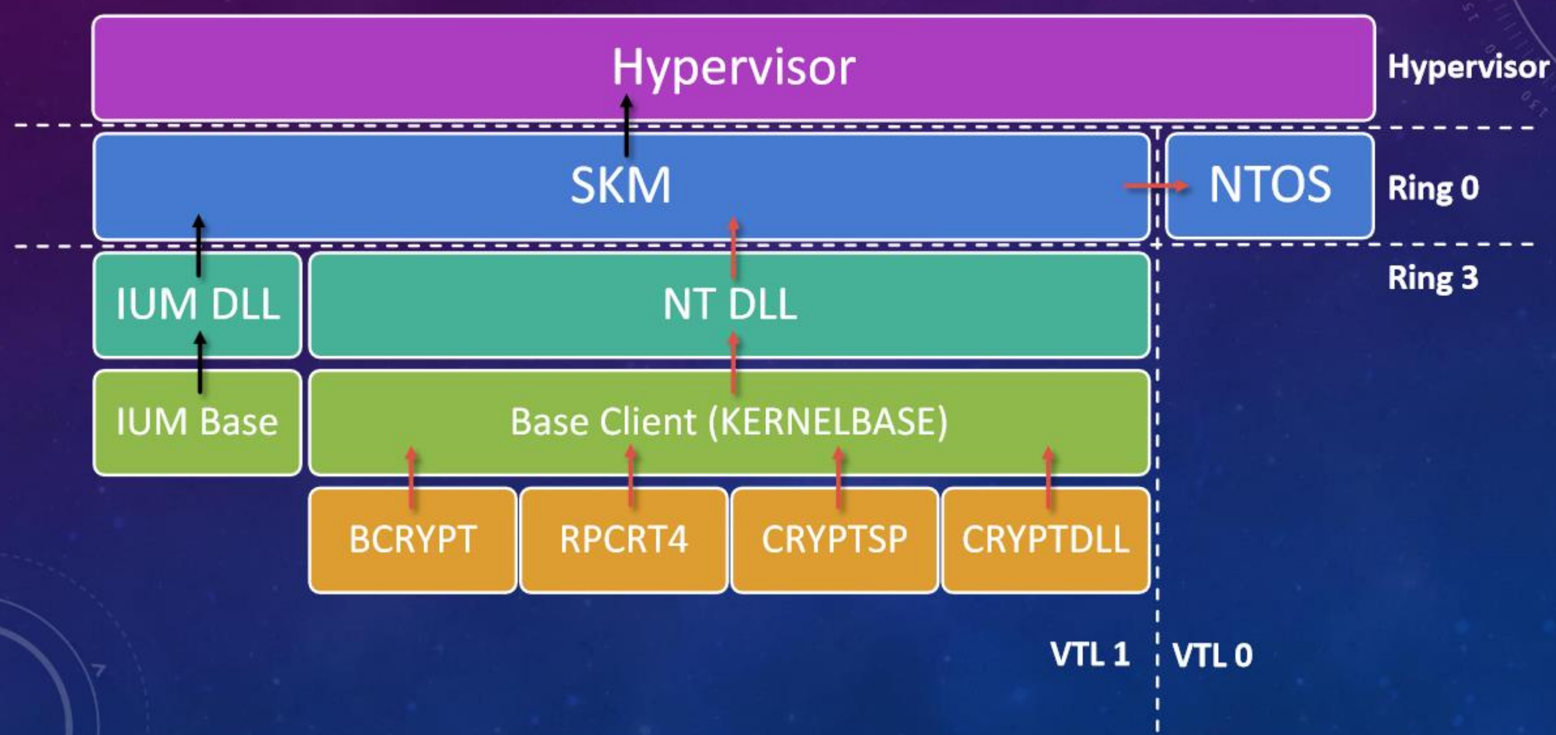
Principes généraux du VSM

- Chaque Virtual Processor est assigné à un Virtual Trust Level
  - VTL0 : Normal World (High Level OS)
  - VTL1 : Secure World (SKM/IUM)
- La mémoire est gérée par ce qui est appelé des Enhanced Page Table, associées elles aussi à un VTL

- Limiter les accès en lecture : Credential Guard
- Limiter les accès en exécution : Device Guard

- Permet de limiter l'accès aux MMIO via les EPT (NX)
- Filtrage des MSR exposés au kernel en VTL0 :
  - TLB attack, split-TLB tricks, filtre les MSR relatifs à la virtualisation ou spécifiques à un modèle de processeur

Principes généraux



Secure Kernel Mode

- Exécuté en VTL1 : securekernel.exe, SKCI.dll, CNG.SYS
- Version allégée du kernel et du module de vérification de l'intégrité du code
- Structures internes différentes du Normal World : *SKPCR, SKTHREAD, SKPROCESS*
- Si Strong Code Guarantees est activé :
    - Les pages tables entry userland ne peuvent pas être exécutable
    - Toute exécution doit être vérifiée en intégrité
    - SMEP est émulé même sur les système dont le processeur est incompatible
    - Les pages MMIO sont non-exécutable (*MmMapIoSpaceEx*)
    - Aucune page kernel ne peut être alloué en tant qu'exécutable
    - Si SLAT : PTE limitées en exécution par les EPT
- Seul SKM peut rendre une EPT exécutable, seulement si validé par HVCI

Isolated User Mode

- Les processus s’exécutant en IUM sont appelés des Trustlets, défini par un ID unique
- Ils sont isolés les uns des autres ainsi que du Normal Mode
- Fortes contraintes d’intégrité du code, même pour les DLLs chargées
- Pas d’accès à CSRSS
- Syscall limités par SKM
- Des trustlets ayant la même Instance GUID peuvent échanger via des Storage Blobs

AMOSSYS

Ce n'est pas fini…

D'abord, un schéma moche(comme moi)

Hyperviseur

Secure Mode Service Call

SKM securekernel.exe

NTOS ntoskrnl.exe

Normal Mode Service Call

IUMDLL

Secure System Call

IUMDLL

System Call

Trustlet

Trustlet

Process

Storage Blobs

Mailboxes

Communications dans le Secure Mode

- Utilisation des exports *Ium** pour les Secure System Call Interfaces (IUN vers SKM)
- Limitations via les SKM Capabilities
- *SecureStorageGet* et *SecureStoragePut* sont limités par des contraintes hardcodées (Get pour TID 2, Set pour TID 3)
- Utilisation de Storage Blobs pour la communication inter-Trustlet
- SKM expose 6 secure system calls via la DLL native IUMDLL.DLL :
  - Identification : *IumGetIdk*, *IumSetTrustletInstance*
  - Cryptographie : *IumCrypto*
  - Communications : *IumPostMailbox*, *IumStoragePut*, *IumStorageGet*

Communications dans le Secure Mode

- IumCrypto expose 5 fonctions :
  - *0 : Encrypt Data*
  - *1 : Decrypt Data*
  - *2 : Decrypt IDK-bound data*
  - *3 : Get SKM-generated seed*
  - *4 : Get SKM FIPS Mode*
- L'appel à l'une ou l'autre de ces fonctions est définies par la structure interne : *IUM_CRTYPTO_PARAMETERS*
- Cependant la DLL native IUMDLL.DLL expose les APIs suivantes qui se chargent du remplissage de cette structure :
  - *DecryptData, EncryptData*
  - *DecryptSKBoundData*
  - *GetSecureIdentifyKey*
  - *GetSeedFromIumKernelState*
  - *GetFipsModeFromIumKernelState*
- 48 syscalls sont actuellement autorisés depuis un Trustlet couvrant : la synchronisation, les threads/workers, ALPC, gestion mémoire, gestion des exceptions et *NtQuery**

Communications dans le Secure Mode

- Donc, les trustlets ne peuvent :
  - Interagir avec le FS
  - Interagir avec le Registre
  - Réaliser des Mutex
- Sous-entend l'utilisation d'un agent dans le Normal Mode afin de traiter les données obtenus depuis le Secure Mode (RPC) :
  - Les arguments subissent un *marshalling* et une *sanitization*, selon leur type : *ALPC_MESSAGE_ATTRIBUTES, OBJECT_ATTRIBUTES, PORT_MESSAGE, SID, UNICODE_STRING, WORKER_FACTORY_DEFERRED_WORK, GENERIC*
  - Le *marshalling* se fait une fois avant le call afin de déduire la taille des arguments et une seconde fois avec la *sanitization* afin de nettoyer les données
- Reprends le principe de Broker utilisé par les AppContainer, Chrome, etc…

Dernière ligne droite

Communications entre NTOS et le Secure Mode

- Depuis le Normal Mode :
  - Secure Mode Calls : *SkCallSecureMode* avec une structure contenant un Operation Code, et un Service Code
  - L'Operation Code a 3 valeures possibles :
    - ✓ 0 : ResumeThread
    - ✓ 1 : Secure Service Call
    - ✓ 2 : TLB Flush
- Depuis le Secure Mode :
  - Normal Mode System Calls : *SkCallNormalMode*, 4 possibles :
    - ✓ 0 : Normal Service Calls
    - ✓ 2 : Normal System Calls from IUM (depuis ring3)
    - ✓ 3 : Normal System Calls from SKM (depuis le ring0)
    - ✓ 4 : Virtual Interrut Assertions
- Utilisation des *Mailboxes* pour la communications avec les Trustlets :
  - *IUM : PostMailbox*
  - *NTOS : VslRetrieveMailbox*
- RPC via le protocole *ncalrpc*
  - Les vulnérabilités liées à l'utilisation de services RPC ainsi qu'au parsing de ces communications peuvent toujours être présentes
  - L'exploitation nécessiterait une seconde vulnérabilité pour le passage en SKM ou le retour en Normal Mode

Communications entre NTOS et le Secure Mode

NTOS

Secure Kernel

User

Kernel

Normal User Mode Thread

Isolated User Mode Thread

Trap Code

Trap Code

NtCreateFile

NtReadFile

NtCreateThread

NtAlpcConnectPort

NtWaitForSingleObject

NtWaitForSingleObject

KiServiceTable

KiSystemCall64

CrossModeDispatch

SKSyscall

Secure Kernel Services

Marshaller / hardener

Unmarshaller / hardener

CrossModeDispatch

Shared Buffers

Hypervisor

Hypothèses d'attaques

Hyperviseur

Code exec ?
HVCI

SKM
securekernel.exe

NTOS
ntoskrnl.exe

Seul possible

IUM->SKM
exploit

Pas de privilèges

Trustlet

Trustlet

Process

Nope.

Attaque via RPC

Code exec ?
HVCI

Conclusion

- Aucune faille ne semble présente dans le design de VSM
- La séparation des privilèges semble cohérente
- Surface d’attaque très limitée
- La sécurité globale dépend des sécurités physiques de la plateforme :
  - Principalement Secure Boot

AMOSSYS

Questions ?

- Merci **Alex @aionescu Ionescu**, cette présentation est quasi intégralement basée sur ses travaux, principalement :
- http://www.alex-ionescu.com/blackhat2015.pdf