Sécurité des systèmes d'exploitation

Virtualisation

Plan

- Principes de virtualisation
  - Hyperviseur
  - Mécanismes techniques
- Classes d'attaques spécifiques
- Hyperviseur, composant de sécurité ?
  - Windows *Virtual Secure Mode*
  - Sécurité du composant Hyper-V

Principes de virtualisation

- Virtualisation : technique permettant de faire fonctionner une machine dite virtuelle sur un matériel abstrait
  - Matériel géré par un hyperviseur (aussi appelé *Virtual Machine Monitor*) qui assure une médiation avec le matériel réel
  - 2 approches :
    - Paravirtualisation : nécessite une modification du système d'exploitation de la VM pour accepter ce matériel
    - Virtualisation complète : l'abstraction doit être représentative du matériel réel et ne pas entraîner de modification du système d'exploitation de la VM
  - Première implémentation à la fin des années 60 (IBM), suivie de travaux de spécification

Principes de virtualisation

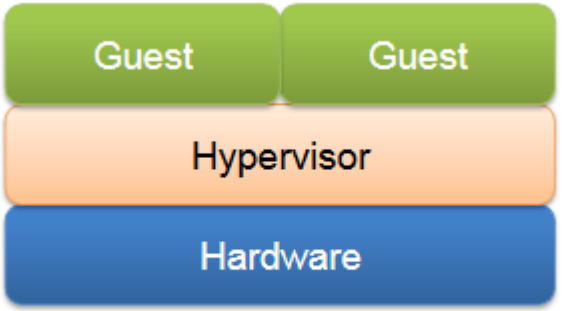
- Cas d'usage
  - Modifier et tester un système d'exploitation
  - Faire fonctionner des systèmes d'exploitations différents ou en version différente
  - Faire fonctionner un système d'exploitation avec une configuration différente du système réel (mémoire, nombre de processeur, …)
  - Modifier la configuration matérielle du système réel sans impacter la configuration d'un système d'exploitation
    - Par ex. obsolescence ou panne du matériel sous-jacent

Hyperviseur

- Assure l'ordonnancement de l'exécution des différentes machines virtuelles invitées
- Assure le contrôle d'accès
  - à la mémoire
    - Réalise aussi l'abstraction de la mémoire physique
  - au processeur
    - Interception de certaines instructions
    - Lecture ou écriture de registres
- Simule le matériel virtuel (disque, réseau, ...)
- Expose des *hypercalls*
  - Dans le cas de la paravirtualisation, permettent à la machine invitée de communiquer avec l'hyperviseur

Hyperviseur

- Type 1 : Xen, Hyper-V, VMWare ESXi
  - Indépendant du système d 'exploitation
    - Parfois appelé bare metal hypervisor
  - Taille réduite, code limité au partage et à la gestion des ressources matérielles entre les VM (dites machines invitées)
  - Utilise souvent un environnement de contrôle qui parle au matériel et multiplexe les requêtes des invités
    - Dom0 pour Xen / partition parente pour Hyper-V
    - Évite d'héberger des pilotes dans le VMM pour renforcer l'isolation et la sécurité

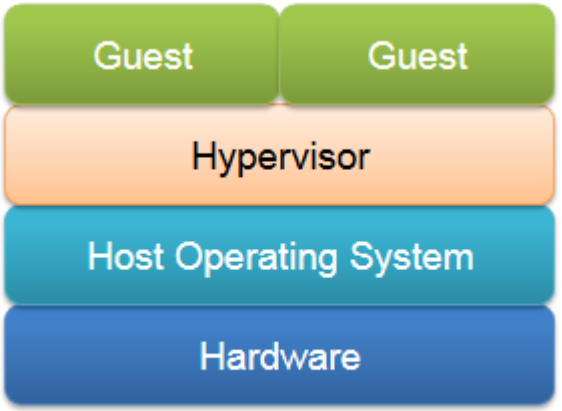


Type 1

Source : Intel

Types d 'hyperviseur

- Type 2 : VirtualBox, KVM (*), VMWare Workstation
  - Installé sur un système d'exploitation hôte, repose sur l'hôte pour les pilotes
  - La sécurité de l'hôte impacte la sécurité des VM



Type 2

Source : Intel

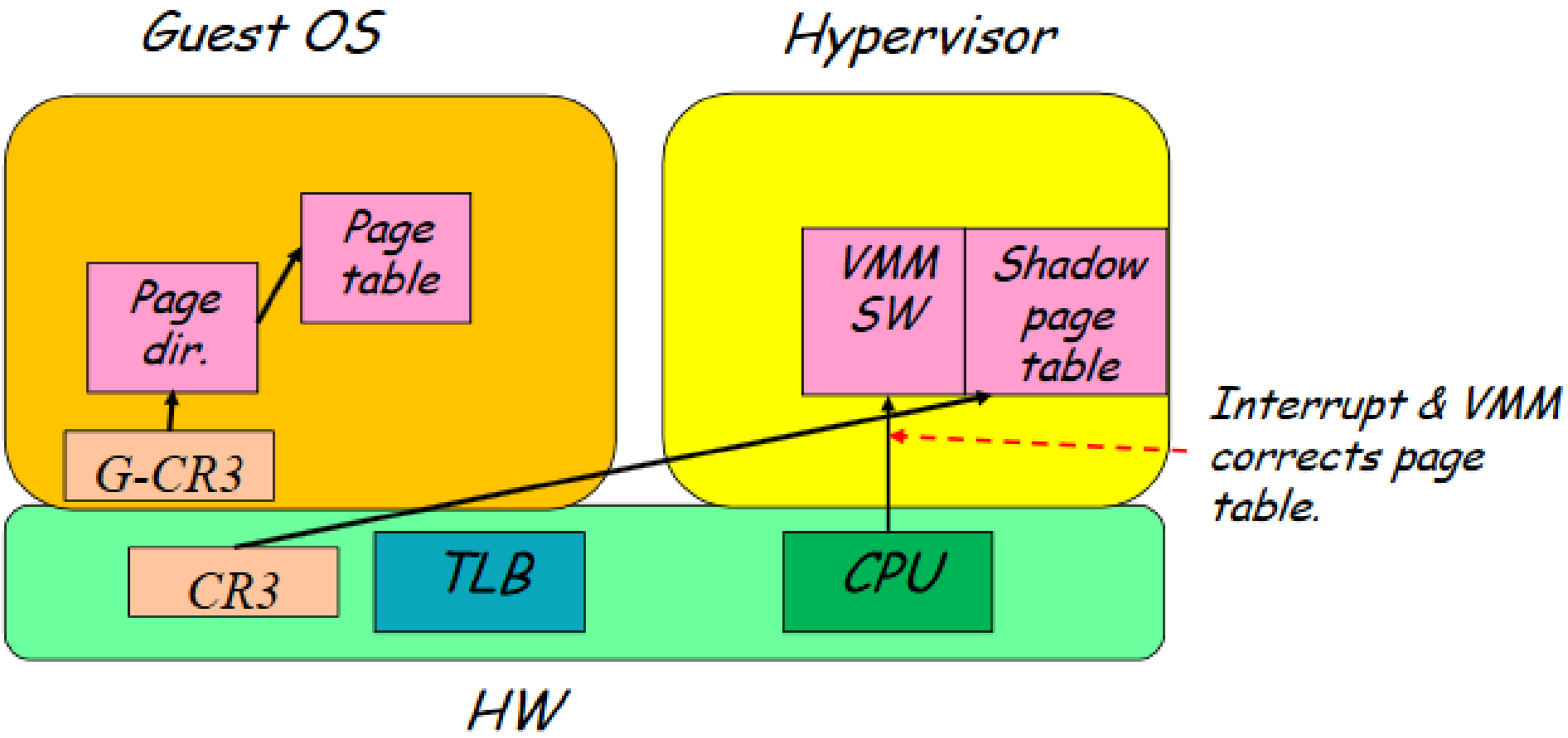
Mécanismes techniques

- Approches historiques
  - Émulation complète
    - Interprète chaque instruction de l'invitée
    - Gestion logicielle des états des invitées
    - Problème : lenteur
  - Translation binaire contrôlée
    - Instructions non-privilégiées sont exécutées sur le processeur physique
    - Instruction privilégiées sont interceptées et émulées par l'hyperviseur
    - Problème : difficile à faire sur une architecture x86
      - Registres : *mov %rax, %cr3* ?
      - Niveaux CPL, DPL
      - GDT, IDT, ...

Mécanismes techniques

- Approches historiques
  - Gestion de la mémoire
    - Découpage la mémoire physique entre l'hyperviseur et les machines invitées
      - Exposer une quantité de mémoire physique inférieure à la réalité
      - Comment empêcher une machine invitée d'accéder à la mémoire d'une autre (ou de l'hyperviseur) ?
        - Le système d'exploitation de la machine invitée ne doit gérer la « vrai » table de pages
    - « Détournement » du mécanisme de pagination
      - Utilisation de *shadow page table* : structure de données qui est activement maintenue et mise à jour par l'hyperviseur
        - La *shadow page table* réplique ce que l'invitée fait dans ses propres tables de page et s'en sert pour faire la translation de l'adresse physique invitée (*guest physical address*) vers l'adresse physique réelle/hôte (*host physical address*)

Mécanismes techniques



Mécanismes techniques

- Nouveaux jeux d'instruction
  - Un hyperviseur fonctionne en interceptant et émulant d'une manière sûre des opérations sensibles réalisées dans la machine invitée
    - Par exemple, un changement de table de pages, qui pourrait donner une accès illégitime à une zone de la mémoire
  - Une assistance matérielle doit permettre d'améliorer la performance et faciliter l'implémentation de l'hyperviseur
  - Intel VT-X / AMD-V
    - Jeux d'instruction incompatibles, implique de choisir une technologie ou de faire 2 implémentations en parallèle

Mécanismes techniques

- Intel VT-X
  - Introduction de 2 « modes » d'opération VMX
    - VMX root operation : exécution de l'hyperviseur
    - VMX non-root operation : exécution d'une machine invitée
  - Le basculement entre VMX root et VMX non-root est marqué par des <u>événements</u> *VM Entry* et *VM Exit*
    - Caractérisent des actions d'ordonnancement entre les VM
      - un *VM Exit* peut aussi intervenir pour gérer une exception

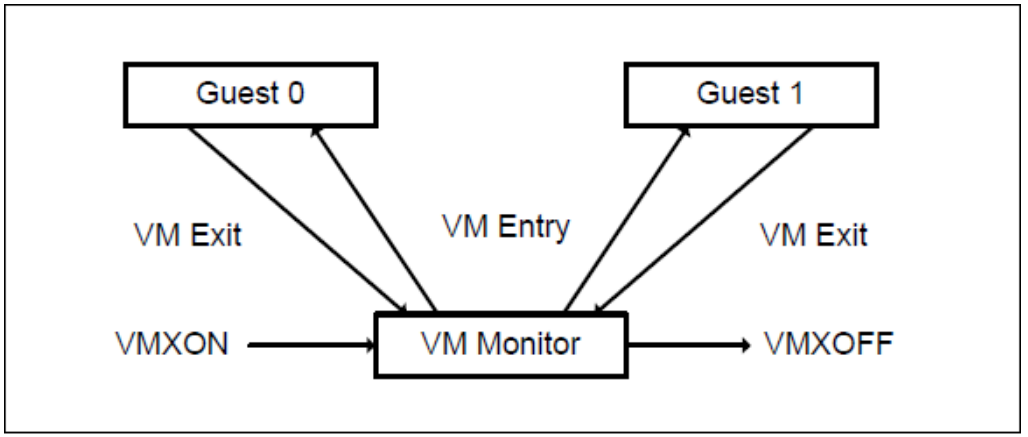


Figure 23-1. Interaction of a Virtual-Machine Monitor and Guests

Source : Intel

Mécanismes techniques

- Intel VT-X (suite)
  - Chaque VM est caractérisée par une structure de données appelée VMCS (*VM Control Structure*)
    - Contient des informations notamment sur
      - Les états (*host-state* & *guest state*) du processeur
      - Des champs de contrôle qui déterminent des conditions (tentative d'exécution des certaines instructions, d'opérations sur certains registres, …) conduisant à des événements *VM exit*
      - Des champs d'informations (la cause précise : interruption, exception, instruction, ...) sur le dernier événement *VM exit*
  - Accès à cette structure VMCS via un pointeur
    - Pointeur manipulé via les instructions VMPTRLD et VMPTRST
    - L'hyperviseur configure une structure VMCS avec les instructions VMREAD, VMWRITE et VMCLEAR

Mécanismes techniques

- Instructions de gestion VMX (Intel VT-X)
  - VMXON : active le cadre *VMX operation* qui va être utilisé par l'hyperviseur
  - VMLAUNCH : charge une machine virtuelle (qui prend le contrôle du processeur) gérée par une structure VMCS → entraîne un événement *VM entry*
  - VMRESUME : redonne le contrôle du processeur à une machine virtuelle (i.e. elle poursuit son exécution) gérée par une structure VMCS → entraîne un événement *VM entry*
  - VMXOFF : quitte le cadre *VMX operation*
    - Marque la fin de l'exécution du code l'hyperviseur

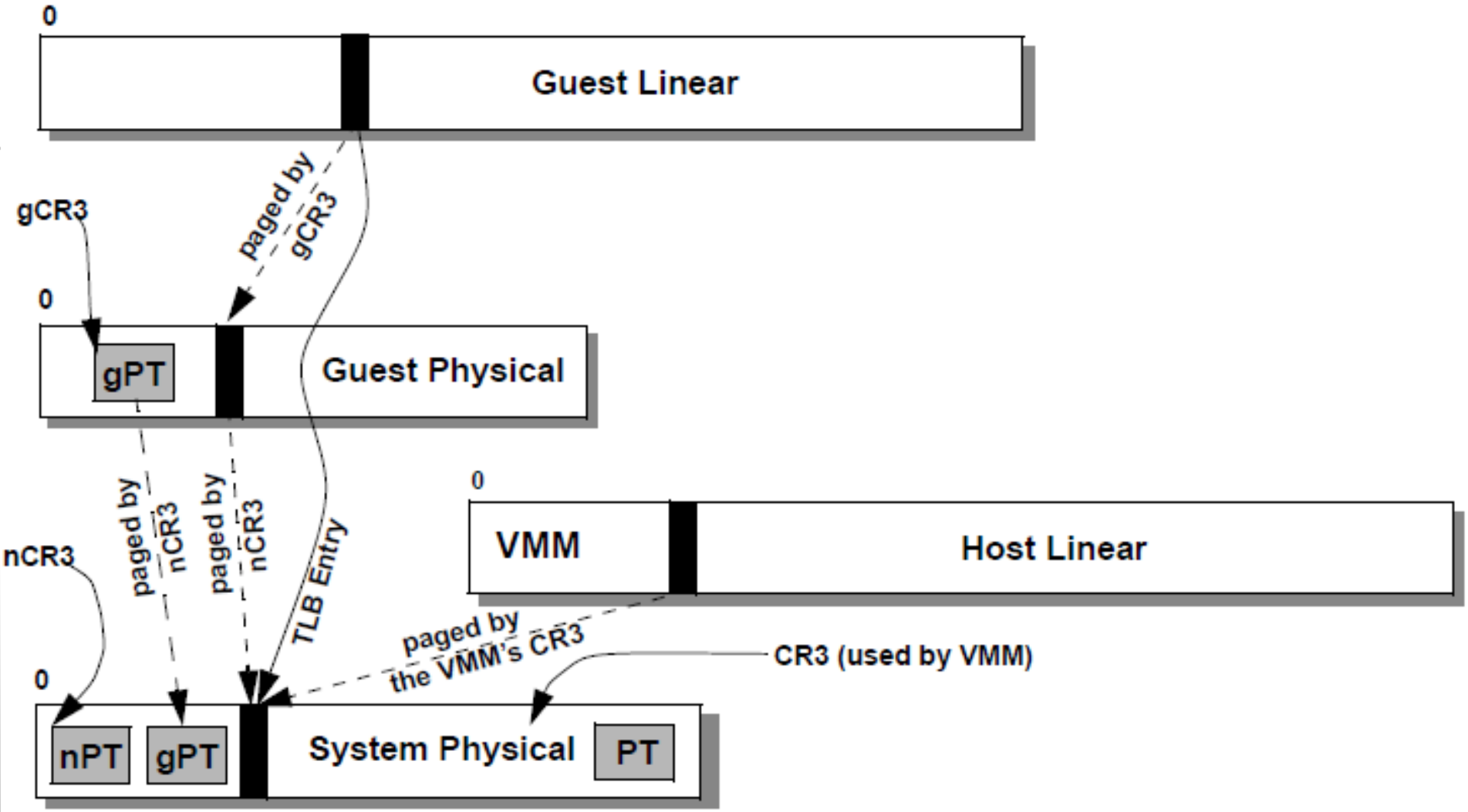
Mécanismes techniques

- Instructions utilisable par la machine invitée (Intel VT-X)
  - VMCALL : permet de solliciter un service offert par l'hyperviseur → entraîne un événement *VM exit*
  - VMFUNC : permet d'invoquer une « fonction de VM » (qui doit être activée et configurée par l'hyperviseur) sans entraîner d'événement *VM exit*
    - Une seule fonction existe dans la documentation Intel
      - Commutation du pointeur EPT (fonction n°0) : permet d'établir une hiérarchie différente de la structure de pagination EPT

Mécanismes techniques

- Gestion des tables de pages
  - L'hyperviseur a besoin de contrôler (et restreindre dans certains cas) l'accès des machines virtuelles à la mémoire physique
- Nouvelle « hiérarchie » de table de pages
  - Principe SLAT (*Second Level Address Translation*) : Intel EPT/ AMD NPT
    - Permet la translation *guest physical address* / *host physical address*
      - Transparent pour l'OS de la VM qui continue à gérer sa mémoire virtuelle
    - L'hyperviseur peut positionner des permissions RWX sur les pages EPT/NPT
      - Les fautes de pages EPT/NPT déclenchent un *VM Exit*

Mécanismes techniques



Source : AMD

Figure 15-13. Address Translation with Nested Paging

Mécanismes techniques

- I/O Memory Management Unit (Intel VT-D, AMD I/O VT)
  - Contrôle (permissions et translation d'adresses) sur les accès DMA des périphériques
  - Peut également être utilisés pour donner accès à d'autres fonctions depuis une VM (périphérique, interruptions, ...)

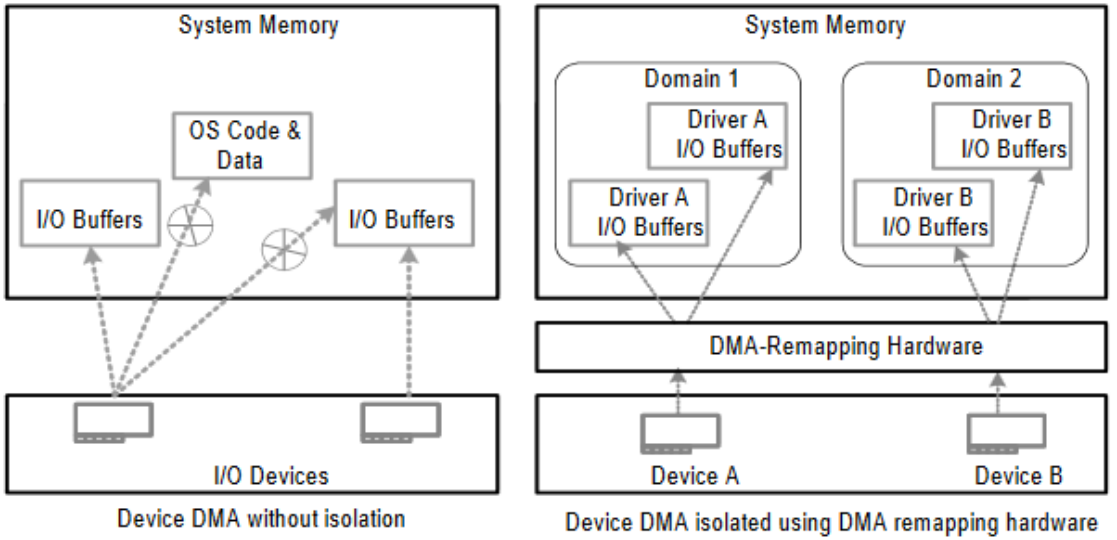
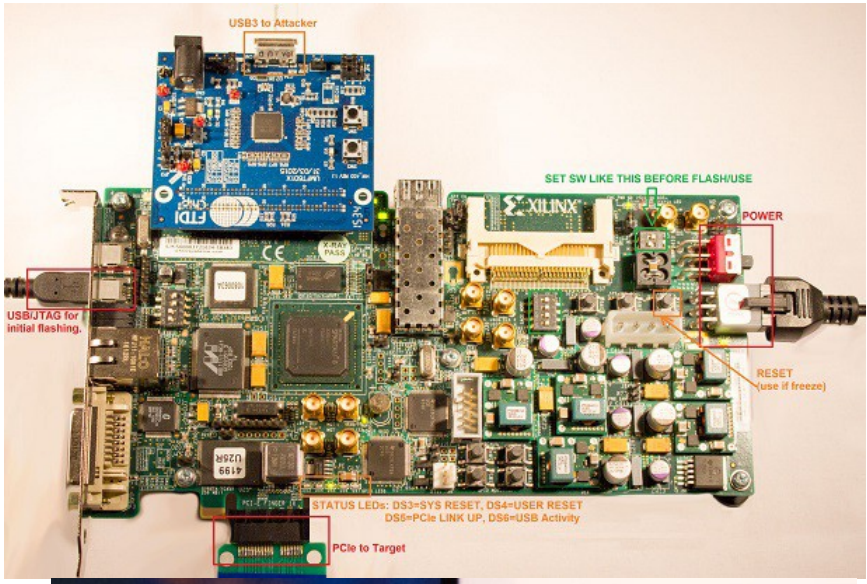


Figure 2-2. Example OS Usage of DMA Remapping

Source : Intel

Mécanismes techniques

- Attaques DMA
  - Utilisation de périphériques équipés d'un contrôleur DMA pour accéder à la RAM sans le contrôle de la MMU
  - Exemple : PCI Leech
    - Utilisation de cartes FPGA PCIe ou de cartes USB3 mini-PCIe
    - Lecture/écriture de la RAM
    - Injection de code noyau
    - Récupération et injection de fichiers



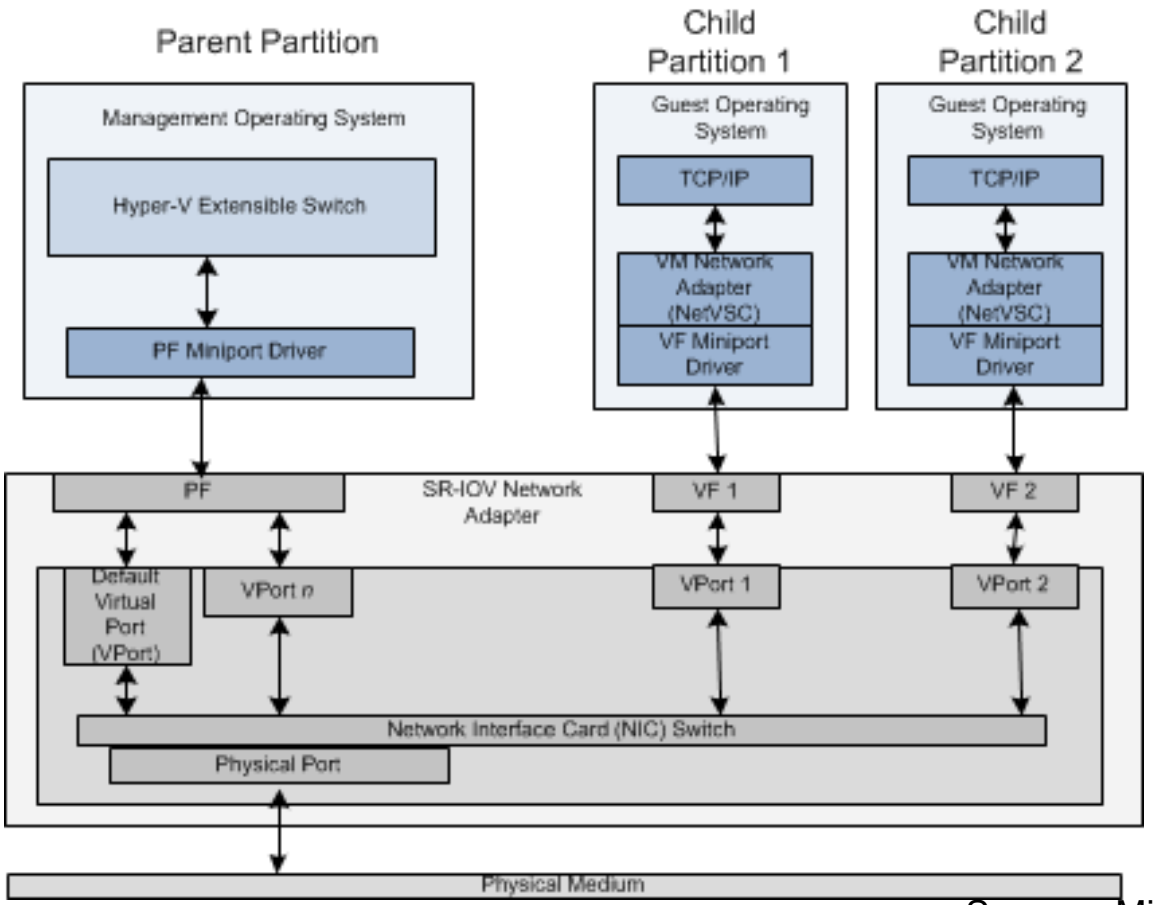
Source : Ulf Frisk

Mécanismes techniques

- SR-IOV (*Single Root I/O Virtualization*)
  - Extension PCI-Express
  - Permet à un périphérique, comme une carte réseau, de séparer les accès à des ressources de fonctions matérielles
    - Une *physical function* (PF) et une ou plus *virtual function* (VF)
  - Permet à du trafic réseau d'outrepasser la couche logicielle de l'hyperviseur et créer une lien direct entre la *virtual function* et la machine virtuelle
    - Augmente les performances est évitant de dédier une carte réseau physique à chaque VM

Mécanismes techniques

- Mécanisme SR-IOV dans un contexte d'interfaces réseaux



Source : Microsoft

Classes d'attaques spécifiques

- Attaquer les autres VM
  - Un attaquant peut profiter d'un contrôle d'accès réduit ou d'une communication inter-VM légitime.
    - Cette attaque dépend de la configuration de l'hôte et du contrôle d'accès
- Attaquer l'hyperviseur
  - Généralement initié depuis une VM et dépendant de l'hyperviseur
    - Les pilotes de paravirtualisation, le partage de presse-papier, la sortie d'affichage et le trafic réseau sous des cibles potentielles

Classes d'attaques spécifiques

- Attaquer le matériel de l'hôte (depuis la VM)
  - Les plateformes matérielles reçoivent des mises à jour de micrologiciel
  - Si le mécanisme est accessible depuis la VM un attaquant peut charger un micrologiciel malveillant
    - L'hyperviseur devrait filtrer ce type de commande
- Attaquer l'architecture de l'hôte
  - Cas typique d'une attaque par canaux auxiliaires sur un composant partagé.
    - Par exemple, l'utilisation malveillante de l'allocation dynamique de mémoire (*memory-ballooning*)

Classes d'attaques spécifiques

- Attaque des VM par l'hôte
  - Peut mener à de la fuite d'informations, une altération du fonctionnement ou une interruption de service
    - Les faiblesses exploitées sont généralement des défauts de validation des entrées ou des certificats, l'élévation de privilège et des problèmes de manipulation des données
- Attaquer l'abstraction de la plateforme
  - Peut mener à des échappements de la VM ou des dénis de service (par l'arrêt de l'hyperviseur)
    - Par exemple, défaut d'implémentation du code qui simule un matériel ou du micrologiciel au niveau de l'hyperviseur

Classes d'attaques spécifiques

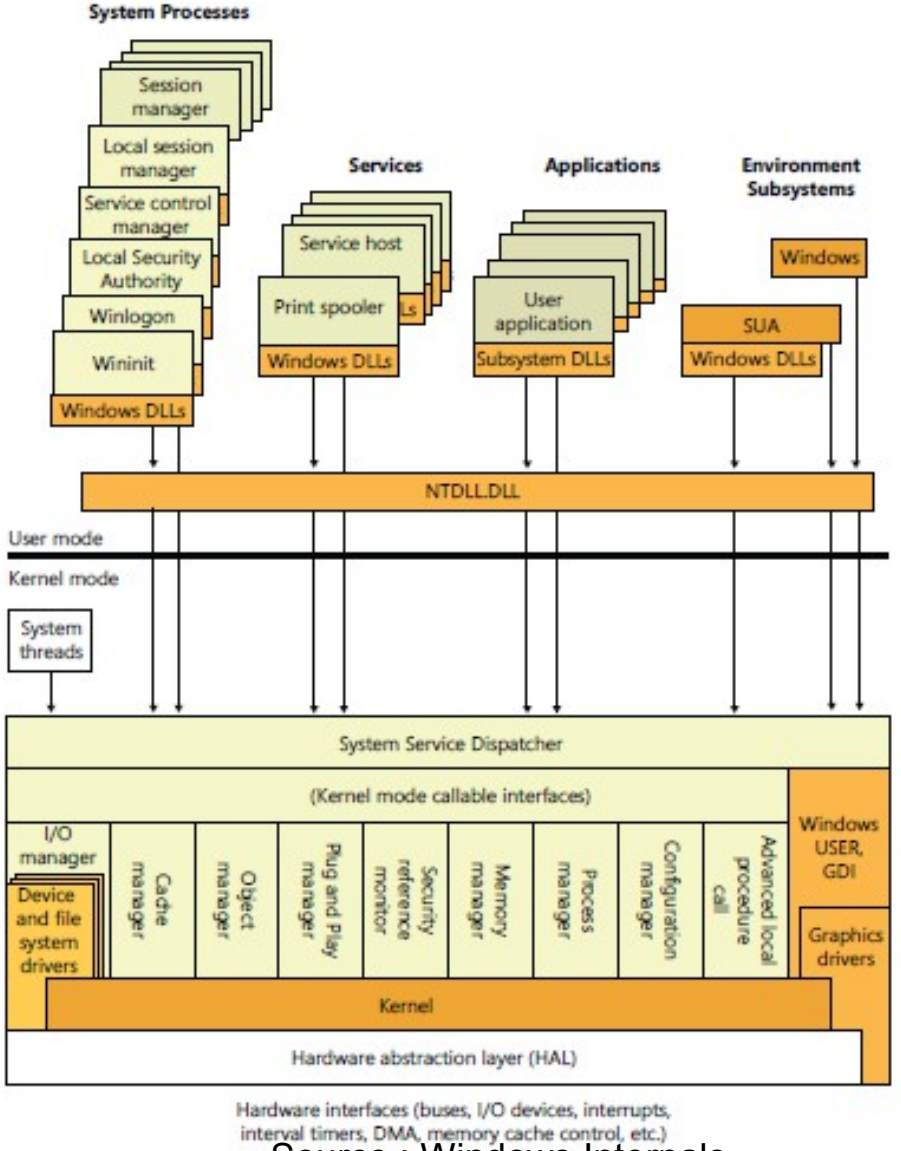
- Interception de données réseau
  - Pas spécifique mais peut-être amplifiée dans un environnement virtuel avec le partage de matériel
    - Par exemple, mutualisation d'une carte réseau ayant accès à plusieurs VLAN
- Attaque sur la couche de gestion du stockage
  - Peut mener à des interruptions ou des usurpations en s'appuyant sur des faiblesses classiques comme la validation d'entrées, l'injection ou du *cross-site scripting*

Hyperviseur, composant de sécurité ?

- L'hyperviseur met en œuvre des fonctions d'isolation de la mémoire et de contrôle des ressources
  - intérêt pour la sécurité
- Hyperviseur = logiciel
  - Taille et complexité du code ?
  - Conception prouvée ?
  - Développement sécurisé ?
  - ...
- Utilisation malveillante possible : rootkit en mode hyperviseur

Windows *Virtual Secure Mode*

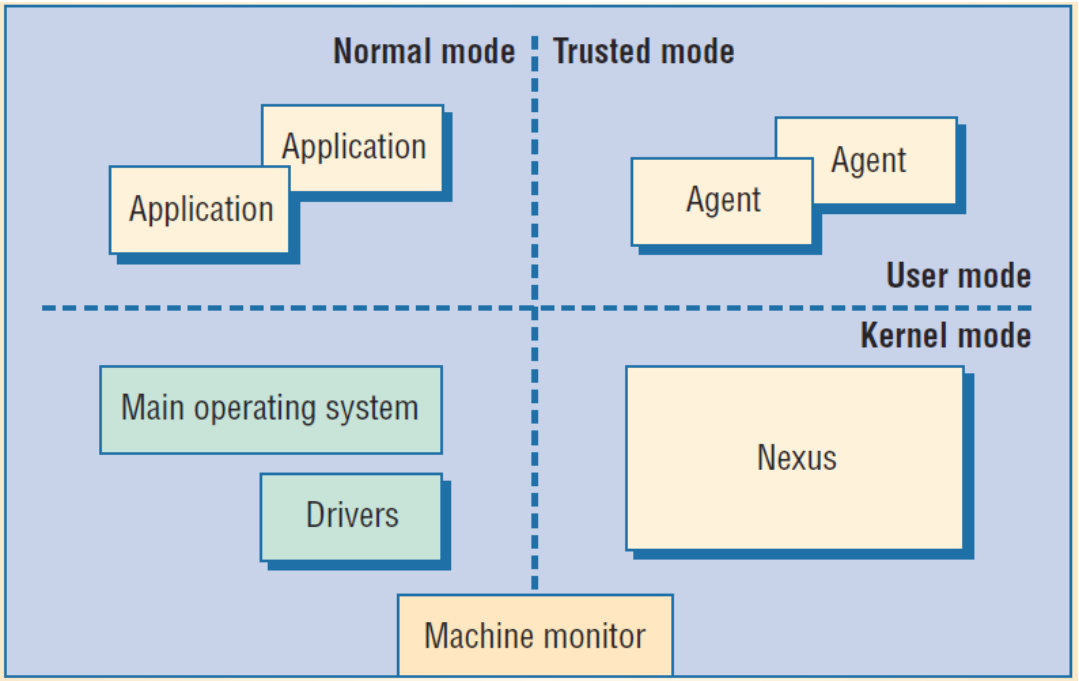
- Quelle TCB pour Windows ?
  - Idéalement le matériel, le noyau, le gestionnaire de mémoire, le gestionnaire d'objet et le moniteur de référence (Security reference monitor)
  - En pratique, tout ce qui est dans l'espace noyau (y compris les pilotes tiers) et le service LSASS



Source : Windows Internals

Windows *Virtual Secure Mode*

- En 2003, plusieurs chercheurs de Microsoft (Lampson et al.) ont publié un article intitulé *A Trusted Open Platform* présentant des pistes pour hausser le niveau de sécurité de Windows
  - Notamment, une approche faisant cohabiter 2 niveaux de sécurité
    - Le Nexus est un petit noyau avec un haut niveau d'assurance
    - La séparation entre les 2 niveaux de sécurité est faite à l'aide d'un hyperviseu



Source : Microsoft

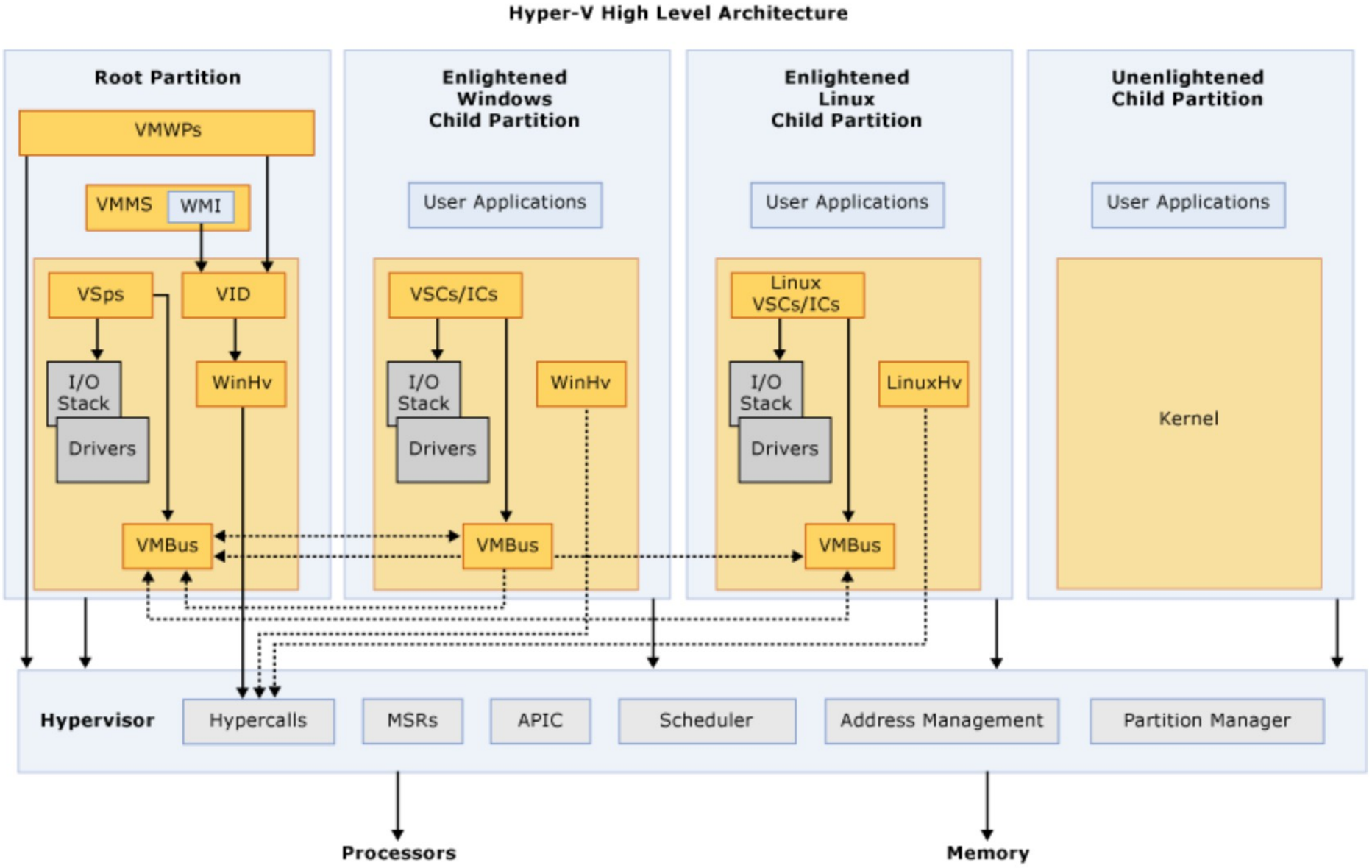
Windows *Virtual Secure Mode*

- Repose sur Hyper-V
  - Disponible dans Windows 10 (variable selon les versions et éditions) et Windows Server 2016/2019
  - Nécessite que le Secure Boot UEFI soit activé
  - Tient compte de la présence d'une IOMMU
- Terminologie
  - VBS : Virtualization Based Security
  - VTL : Virtual Trust Level
    - 16 niveaux définis (0 est le plus bas niveau)
      - A ce jour, seuls VTL0 et VTL1 sont utilisés
  - *Truslet* : services hébergés par le VTL1
    - *Credential Guard*, *Device Guard*, vTPM

Windows *Virtual Secure Mode*

- Hyper-V
  - Hyperviseur de type 1 introduit dans Windows Server 2008
    - Repose sur les jeux d'instructions spécifiques d'Intel ou AMD
  - Les machines virtuelles (appelée parfois partition enfante dans la terminologie) peuvent utiliser des « services invités » (approche paravirtualisation)

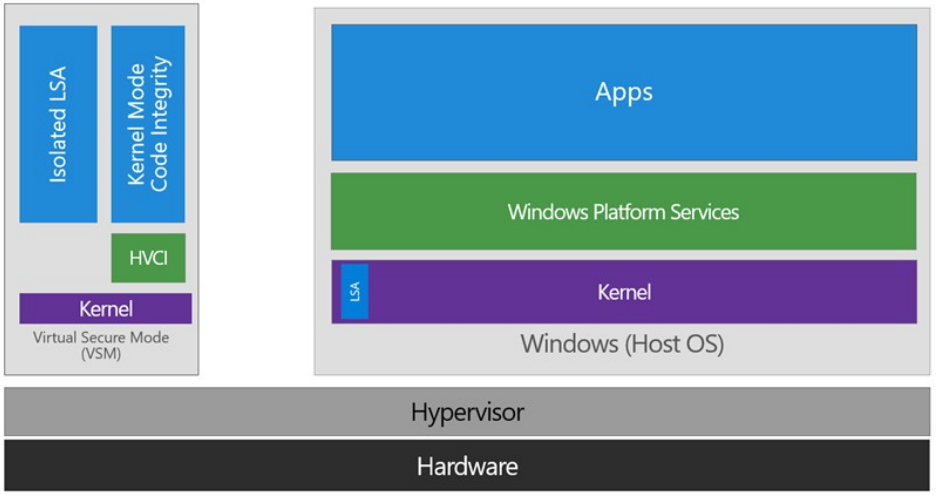
Windows *Virtual Secure Mode*



Source : Microsoft

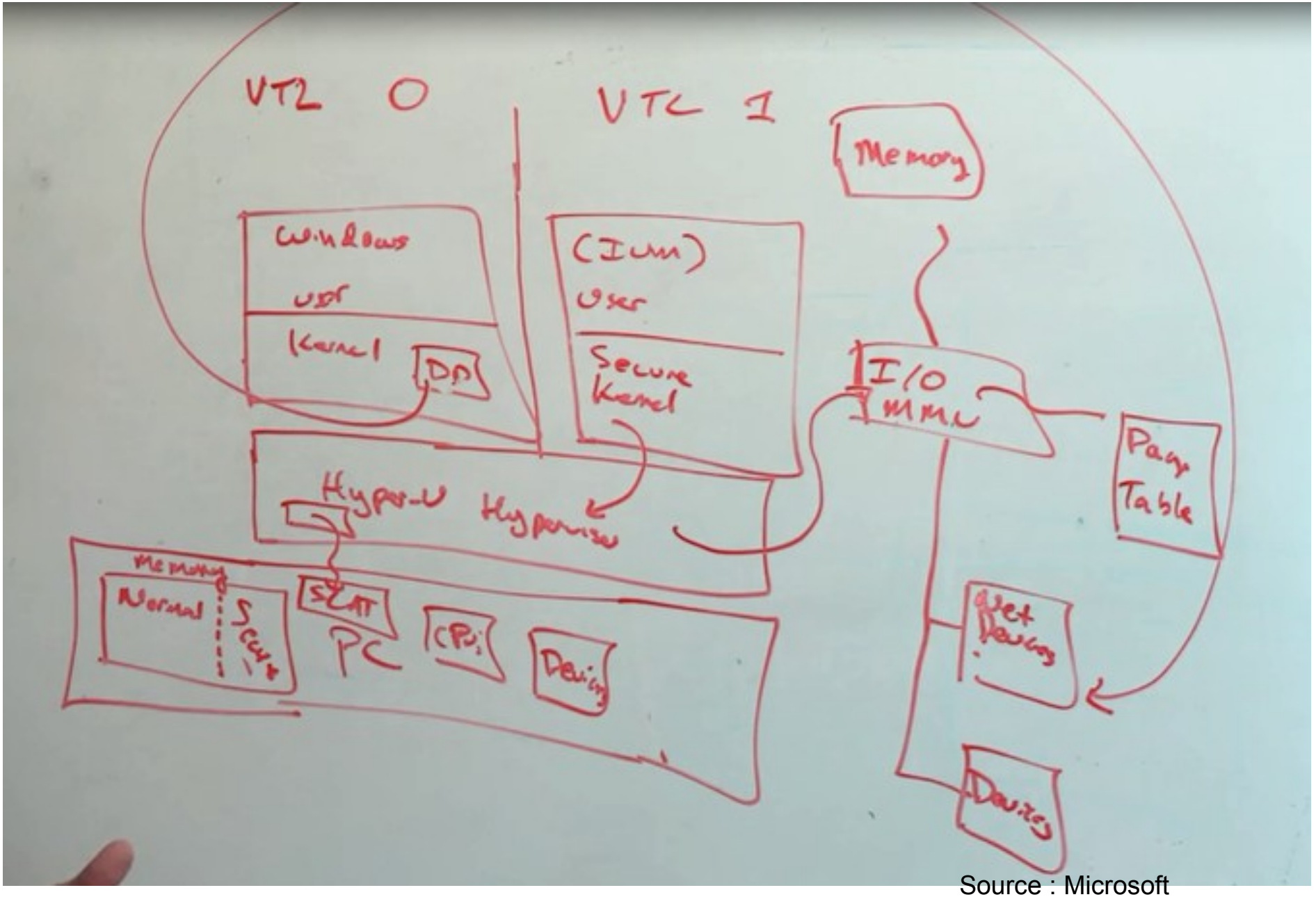
Windows *Virtual Secure Mode*

- Séparation entre 2 environnements :
  - VTL0 qui contient le noyau NT et les applications
  - VTL1 qui contient le *SecureKernel* et des services protégés dans le *Isolated User Mode*
    - Ne contient pas de code tiers comme des pilotes



Source : Microsoft

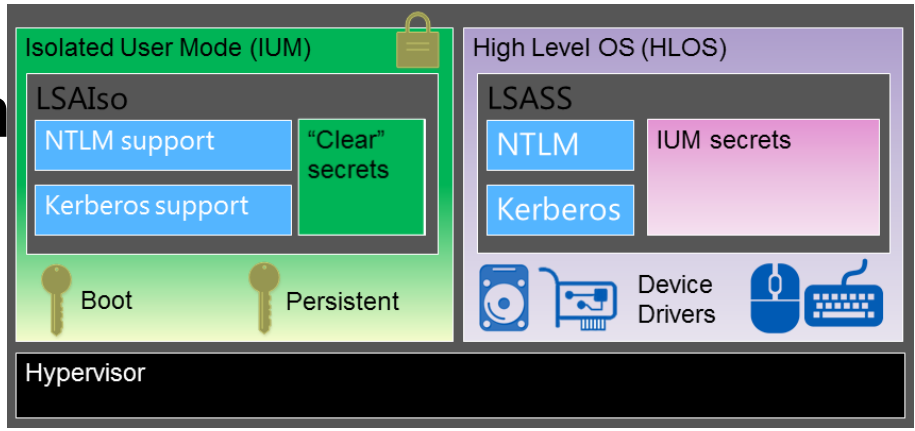
Windows *Virtual Secure Mode*



Source : Microsoft

Windows *Virtual Secure Mode*

- Credential Guard
  - Protection le vol des éléments d'authentification (hash NTLM, ticket Kerberos) dans la mémoire du processus LSASS
    - « anti-Mimikatz »
- Application Guard
  - Exécution de certains onglets du navigateur Edge dans le *Isolated User Mode* (VTL1)



Source : Microsoft

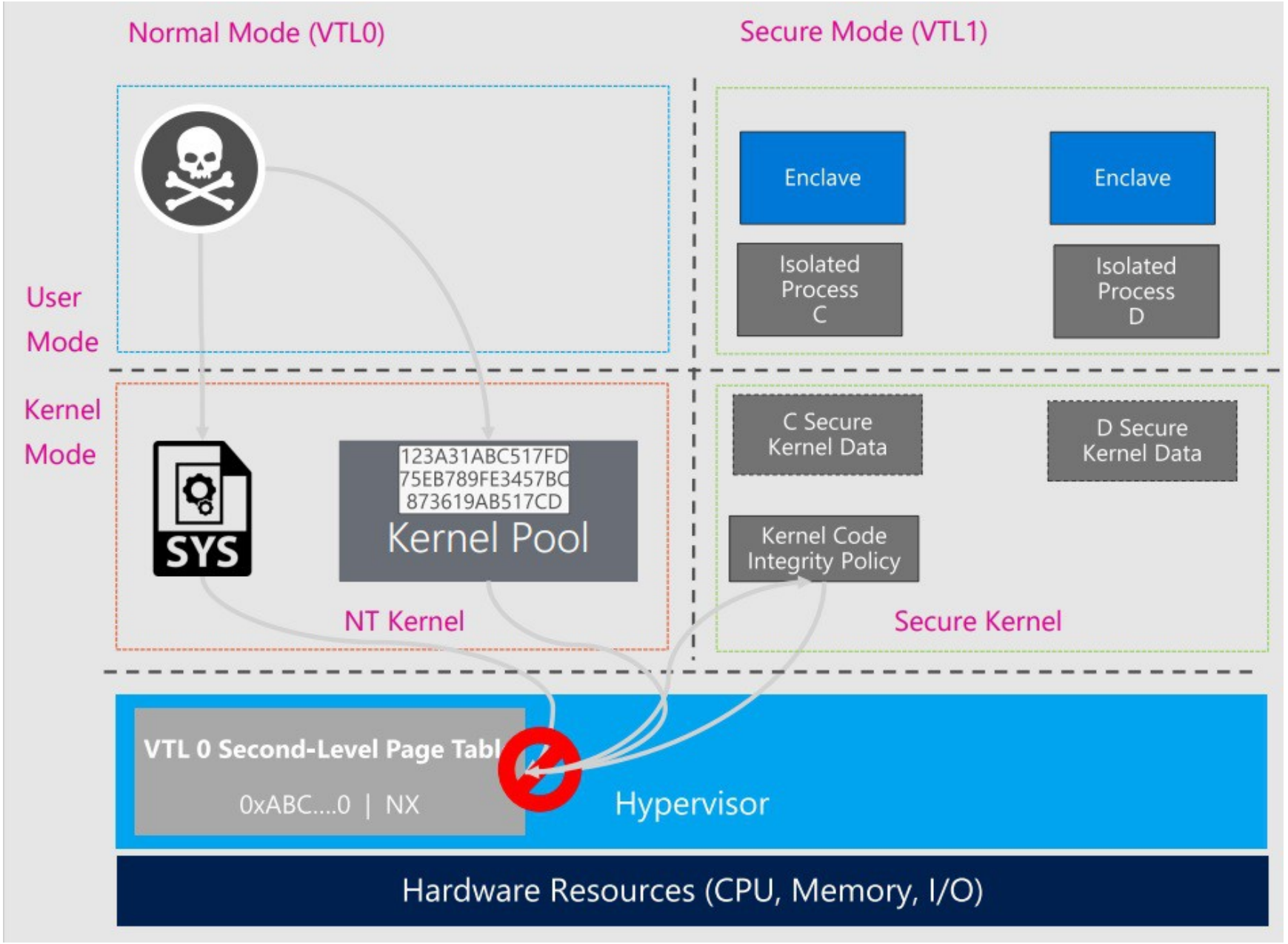
Windows *Virtual Secure Mode*

- Device Guard fournit une solution de contrôle du code exécuté (approche liste blanche)
  - Réutilisation de composants présents depuis Windows 8
    - UEFI Secure Boot : vérification de la signature du chargeur de démarrage
    - KMCI et UMCI : vérification de la signature du code exécutable (noyau et espace utilisateur)
  - Extension
    - Introduction d'un module de vérification de la signature du code qui fonctionne au niveau hyperviseur : HVCI
    - KMCI est déplacé au niveau VTL1 avec HVCI
    - La liste blanche de Code Integrity doit être configurée
      - via des cmdlets PowerShell

Windows *Virtual Secure Mode*

- Code Integrity indique (depuis VTL1) à l'hyperviseur quelles sont les pages contenant du code signé
  - Utilisation des permissions de pages sur les tables EPT
- Si l'OS tente d'exécuter une page définie comme non-exécutable au niveau EPT, déclenchement d'un VMExit
  - Même si la page est vue comme exécutable au niveau de l'OS

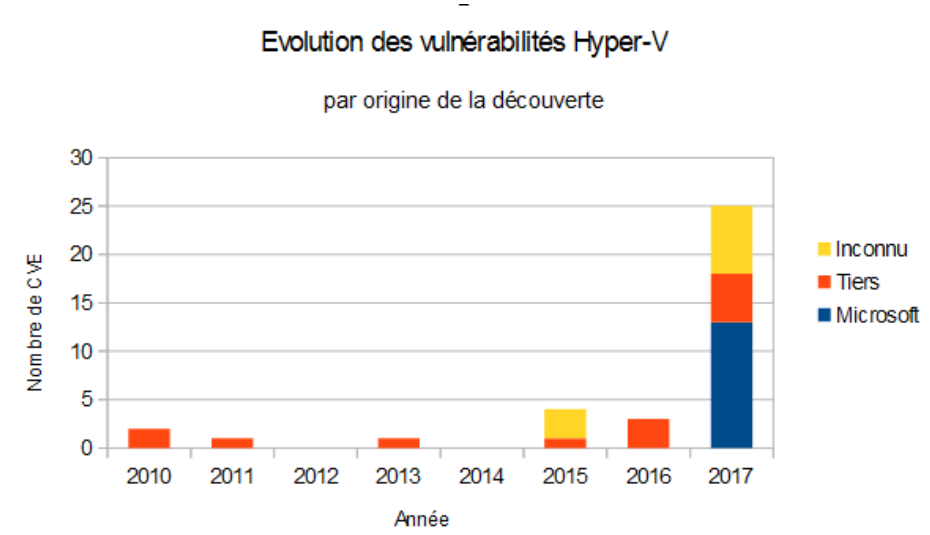
Windows *Virtual Secure Mode*

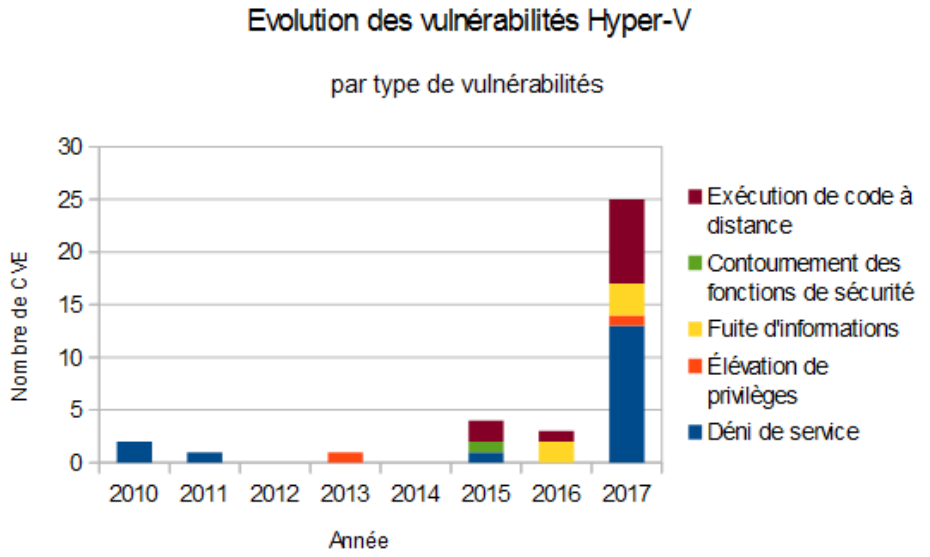


Source : D. Weston & A. Ionescu

Sécurité du composant Hyper-V

- Vulnérabilités Hyper-V
  - Forte hausse en 2017
    - Mobilisation des équipes de l'éditeur
  - Vues dans un contexte Hyper-V classique (hébergement de VM), quel impact sur VSM ?
    - Peu d'informations
    - Limité dans certains cas :
      - le « Secure kernel » n'utilise pas de réseau virtuel ou le VMBus





Sécurité du composant Hyper-V

- « Bug bounty » Hyper-V depuis le 31/05/2017
  - Windows Server 2012 R2 et 2016, Windows 10 (Insider Preview uniquement)
  - Exécution de code à distance : de 5.000$ à 150.000$
  - Déni de service ou fuite d'informations : de 5.000$ à 15.000$

Pour approfondir (1/2)

- Intel® 64 and IA-32 Architectures Software Developer's Manual (Vol. 3, ch. 23 à 33)
- Security aspects of virtualization, ENISA, 2017
- Problématiques de sécurité associées à la virtualisation des systèmes d'information, ANSSI, 2013
- Protection des systèmes informatiques vis-à-vis des malveillances : un hyperviseur de sécurité assisté par le matériel, Thèse de doctorat, Benoît Morgan

Pour approfondir (2/2)

- Windows 10 Virtual Secure Mode with David Hepkin, Channel 9
- Hypervisor Top Level Functional Specification v5.0c, Microsoft
- First Steps in Hyper-V Research, Mircosoft Security Research & Defense Blog
- Analysis Of The Attack Surface Of Windows 10 Virtualization-Based Security, R. Wotjtzuck, Black Hat USA 2016
- Introduction to Windows Device Guard, Matt Graeber, 2016

Questions ?