Trusted Computing : limitations actuelles et perspectives SSTIC 2010 – Rennes

Frédéric Guihéry, Goulven Guiheux, Frédéric Rémi

AMOSSYS

11 Juin 2010

Plan

Amossys

Amossys

- Localisé à Rennes
- Expertise et conseil en sécurité des systèmes d'information
- Laboratoire d'évaluation (CESTI / CSPN)
- Membre contributeur du TCG

Trusted Computing : où en est-on ?

Trusted Computing : où en est-on ?

Dernières spécifications

- Trusted Platform Module (TCG – 2000/2006)
- Trusted Network Connect (TCG – 2008/2009) : NAC et vérification d'intégrité
- Trusted Storage (TCG – 2007/2009)
- Trusted Execution (Intel/AMD – 2007/2008)

Produits matériels

- TPM : 80 millions en 2008 (source : IDC)
- Full Disk Encryption : Hitachi, Seagate, Samsung, Toshiba
- Trusted Execution : Intel et AMD pour le monde PC

Trusted Computing : où en est-on ?

Principales applications grand public

- Microsoft Bitlocker (chiffrement système)
- HP Protect Tools (chiffrement de fichiers)
- Wave Embassy Trust Suite (gestion des clés, signature et chiffrement)
- IBM Software TPM (implémentation logicielle d'un TPM)
- IBM IMA (mesure d'intégrité des binaires exécutés – intégré dans Linux 2.6.30)
- eCryptfs (chiffrement de fichiers)
- Boot loaders : Trusted Grub (mesure d'intégrité), Trusted Boot (mesure et vérification d'intégrité)
- Fournisseurs cryptographiques : MS TPM CAPI CSP, IBM TrouSerS TSP, TPMJ, JTPM

Trusted Computing : où en est-on ?

Bilan sur l'activité

- Peu de produits commerciaux
- Peu de produits réellement déployés
- Les OS du marché ne tirent pas parti, ou très peu, des dernières technologies (Intel TXT, AMD SVM)
  - Xen (2007) est le premier système à intégrer Intel TXT, à travers Trusted Boot (mais désactivé par défaut)
  - Linux (2009, version 2.6.32) intègre également Trusted Boot (mais désactivé par défaut, même sur un PC équipé de TXT)
  - Qubes (2010) est le premier prototype public intégrant Intel TXT par défaut (et utilisant l'IOMMU pour cloisonner les drivers !)

Trusted Computing : où en est-on ?

Bilan sur les besoins couverts

- Les produits actuels répondent à un ensemble restreint de besoins
  - Protection des données stockées (*Data at rest*)
  - Gestion d'identité (trousseau de clé et signature)
  - Intégrité du système au démarrage (pas encore mature)
- Or, les deux premiers besoins sont déjà couverts par des solutions alternatives (OpenSSL, GnuPG, Truecrypt, etc.) déjà bien déployées

D'autres besoins à couvrir

- Intégrité du système (le noyau notamment) pendant l'exécution
- Confidentialité d'exécution

Trusted Computing : où en est-on ?

Bilan

- Positionnement imprécis par rapport aux solutions alternatives qui rendent un service équivalent ; et difficulté de changer les habitudes
- Malgré la standardisation ISO des spécifications TPM et malgré la stimulation académique, le TC reste très peu implémenté de manière industrielle
- Importance du logiciel libre pour la démocratisation des solutions, ce qui peut apparaitre paradoxal avec l'opposition de la communauté LL à l'époque de TCPA/Palladium
- Solutions libres majoritairement portées par quelques industriels (IBM et Intel) et plusieurs chercheurs académiques (Etats-Unis, Allemagne et Japon)

Trusted Computing : où en est-on ?

Et pourtant...

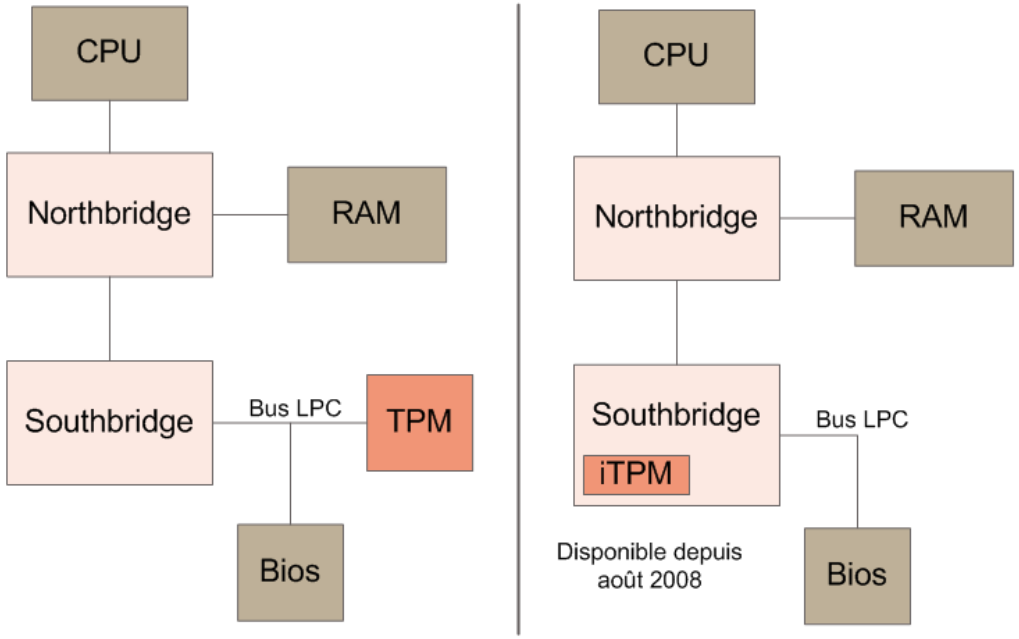
- Intégration de plus en plus importante du TC dans des solutions libres
- Nombreux laboratoires académiques actifs sur le TC aux Etats-Unis (CMU Cylab, MIT CSAIL, Standford) en Europe (Bochum, Graz IAIK, Cambridge) et au Japon
- Conférences dédiées : TRUST, TIW, ATC, ETISS
- Financements européens récurrents : FP6/OpenTC, FP7/Tecom, FP7-ICT Call 5/6, TSC Medea
- Agences nationales impliquées : NSA (HAP), CESG, BSI et ANSSI (animation d'un groupe de travail)

Les technologies

Le composant TPM

Rappels sur le TPM

- Crypto-processeur esclave connecté sur le bus LPC de la carte mère
- Le TPM n'a aucun contrôle sur l'exécution du système
- Il ne manipule que du matériel cryptographique (clés, hachés, données (dé)chiffrées) et n'a aucune compréhension de l'origine des données ni de leur sémantique
- Le TPM est activable et administrable par le propriétaire de la plate-forme
- Principaux fondeurs : Infineon, Atmel, Broadcom, STM, Intel, etc.

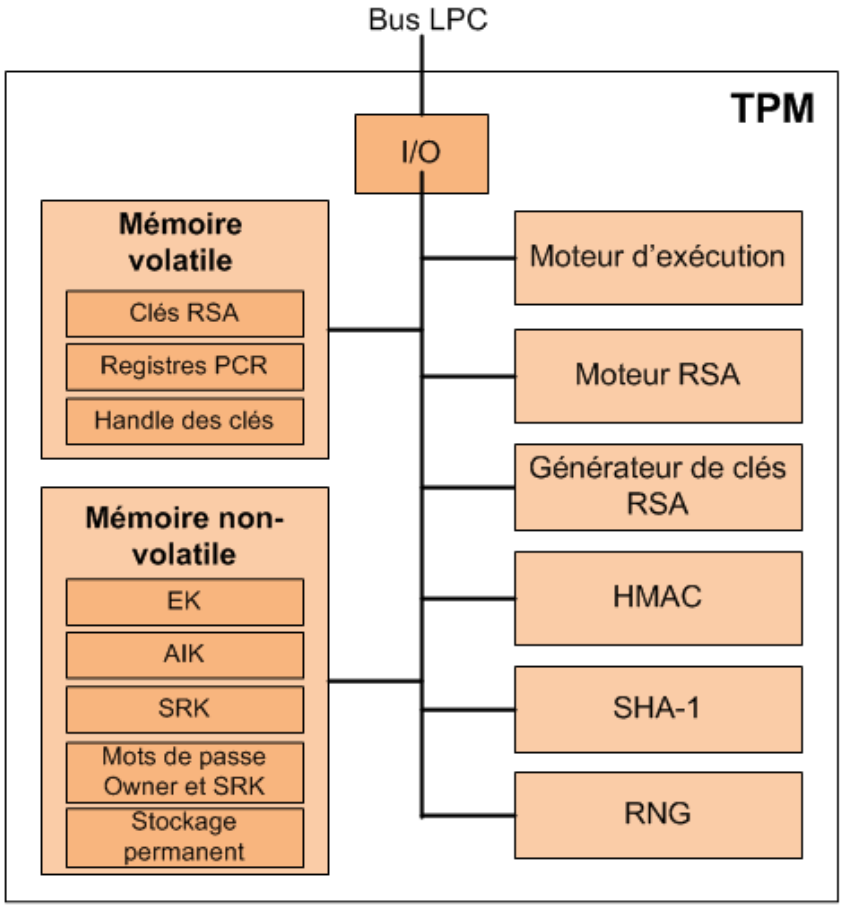


Fonctionnalités du TPM

- Génération d'aléa
- Gestion des clés cryptographiques
- Chiffrement et signature RSA
- Fonctions SHA-1 et HMAC-SHA-1
- Registres PCR pour stocker des valeurs SHA-1
- Registres uniquement modifiables avec la fonction *extend*

$$PCR_{t+1}[i] = SHA1(PCR_t[i]||M)$$

- Mécanisme de signature des registres PCR
- Possibilité de conditionner des opérations cryptographiques à l'état de registres PCR



Avantages et inconvénients du TPM

Avantages

- Les opérations cryptographiques sont réalisées dans un composant matériel
- La clé privée RSA ne peut pas être exportée du TPM
- Composant de base pour des applications de sécurité

Inconvénients

- Pas de chiffrement symétrique
- Les opérations cryptographiques sont lentes (RSA)
- De fait, le chiffrement d'une grosse quantité de données doit se faire de manière logicielle (ce qui expose la clé de session dans la mémoire du système)
- Pas de diversité en termes d'algorithmes : si SHA-1 est "cassé" (attaques en secondes pré-images), alors remise en cause de tous les modèles de sécurité construits autour du TPM
- SHA-1 n'est plus recommandé par les agences

TPM.next

Orientations pour les prochaines spécifications TPM

- Disponibilité de plusieurs suites algorithmiques
- Flexibilité : modèles de sécurité non liés à un seul algorithme
- Prise en compte des spécificités de la virtualisation (ex : hiérarchie de stockage propre à chaque VM)
- Performance : utilisation de clés de session symétriques, elles-même protégées par des clés asymétriques
- Prise en compte des attaques sur les protocoles d'autorisation OSAP et OIAP
- Simplification de l'activation/désactivation du TPM

Référence : "Features Under Consideration For The Next Generation Of TPM"

L'attaque de Tarnovsky

Présentation de l'attaque

- Objectif de l'attaque : accès aux clés privées RSA
- Cible : puce Infineon SLE66 CL PE (personnalisation TPM : SLB 9635 TT 1.2)
- Matériel : microscope électronique (environ 70.000 $)

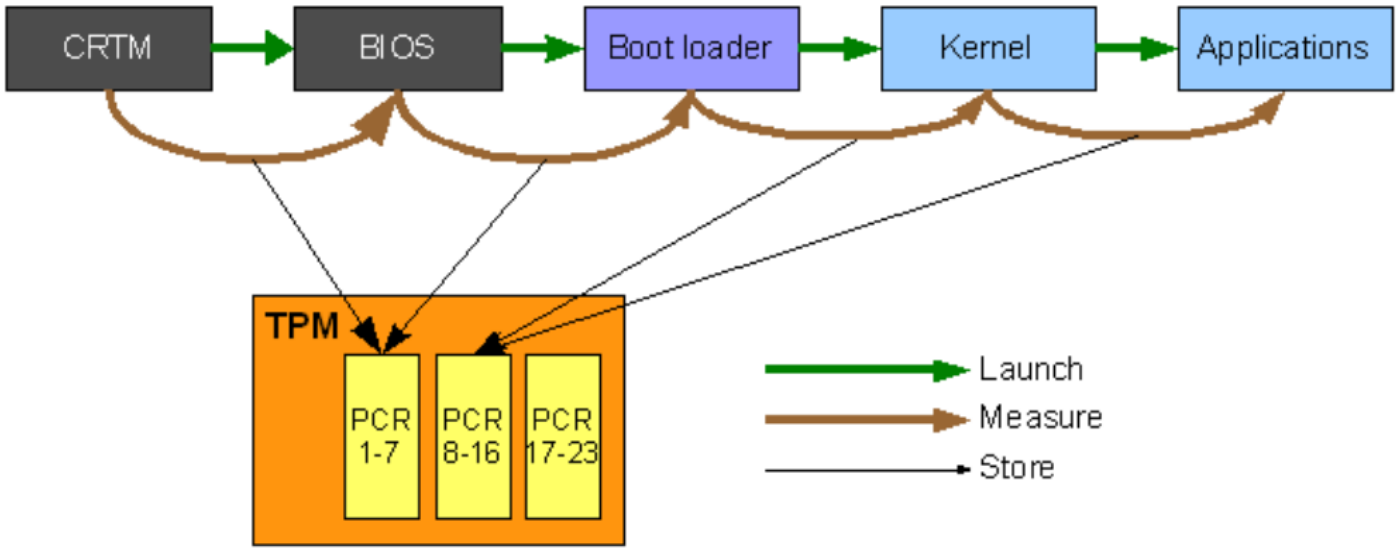
Analyse

- Nécessite un accès physique (ne fonctionne pas sur tous les modèles de sécurité)
- Tarnovky évalue l'investissement à 200.000 $ sur une période de 9 mois
- Puce TPM de 2005, présente dans les Xbox 360
- 16 bits/0,220 microns contre 32 bits/0,130 microns aujourd'hui (SL88)
- Puce SLE66 CX680 PE évaluée CC EAL4+ (qui prend en compte les attaques de niveau modéré)
- Le TPM n'a pas vocation à résister à ce type d'attaques matérielles (le TPM ne coute que quelques dollars)
- Travail intéressant pour évaluer l'état de l'art en termes d'attaque et ainsi calibrer le niveau d'exigences des certifications

Le principe de SRTM

Static Root of Trust for Measurement

- Principe : chaque élément de la chaine de démarrage mesure l'intégrité de l'élément suivant avant de l'exécuter (*Transitive Trust*)
  - Mécanisme initié par le BIOS (CRTM) qui est la racine de la chaine de confiance
  - Mesures d'intégrité stockées dans les registres PCR
- La sécurité de ce mécanisme repose sur la robustesse du SHA-1 et sur la non-discontinuité de la chaine de confiance



Le principe de SRTM

Faiblesses du SRTM

- TCB "large" : BIOS, microcode des composants PCI
- Mesures d'intégrité liées à la plateforme
- Montée à l'échelle difficile
- Faiblesse sur la protection de la chaine de confiance
  - La compromission du microcode d'une carte PCI (cf : attaque de Duflot, Perez, Valadon et Levillain sur les cartes réseau) expose le SRTM a des accès DMA illégitimes
  - Attaque de type "TOCTOU" : écriture DMA sur un élément de la chaine de confiance mappé en mémoire, entre sa mesure d'intégrité et son exécution

Evil Maid Attack

Scénario

- Idée et implémentation par Joanna Rutkowska et Alex Tereshkin
- Cible : portable avec partition système chiffrée avec TrueCrypt
- Attaque en trois étapes :
  1. Accès physique au portable (Evil Maid) et insertion d'un code malveillant
  2. Récupération de la passphrase de l'utilisateur par le code malveillant
  3. Second accès physique pour voler le portable
- Autre variante : faire une copie du disque lors du premier accès physique (dépend du contexte de l'attaque)

Evil Maid Attack

Et avec un TPM ?

- Idée pour contrer la précédente attaque : conditionner le descellement de la clé de session Truecrypt à un SRTM
- ...mais on reste exposé à une attaque plus complexe et moins furtive :
  - Rien n'empêche le code malveillant de s'exécuter et d'émuler une fausse interface
  - Ce code peut alors récupérer la passphrase, puis s'auto-effacer et forcer un reboot sous un prétexte quelconque
  - Au prochain redémarrage, le SRTM ne fait plus apparaitre de modification d'intégrité

Analyse

- On atteint ici deux limitations importantes de la mesure/vérification d'intégrité au démarrage
  - L'impossibilité d'empêcher un code quelconque de s'exécuter au démarrage d'un PC : le passage par un SRTM n'est pas obligatoire
  - La difficulté d'assurer une communication de confiance entre le PC et l'utilisateur

Trusted Execution : Intel TXT et AMD SVM

Trusted Execution : Intel TXT et AMD SVM

Objectifs

- Démarrage à chaud (*Dynamic Launch*) d'un environnement d'exécution de confiance (MLE : *Measured Launch Environment*)
- Protections mémoire de l'environnement d'exécution, afin d'éviter toute compromission depuis un code externe
- Vérification d'intégrité au démarrage de l'environnement vis-à-vis d'une politique de sécurité (bonus sous Intel)

Trusted Execution : Intel TXT et AMD SVM

Technologies sous-jacentes

- Puce TPM 1.2
- Virtualisation matérielle (Intel VMX ou AMD SVM)
- Processeur supportant le démarrage à chaud (*Dynamic Launch*)
  - Intel TXT / SMX : Trusted eXecution Technology / Safer Mode Extensions (instructions *GETSEC[*]*)
  - AMD SVM / Presidio : Secure Virtualization Mode (instruction skinit)
- Chipset Northbridge supportant TXT et l'IOMMU

Trusted Execution : Intel TXT et AMD SVM

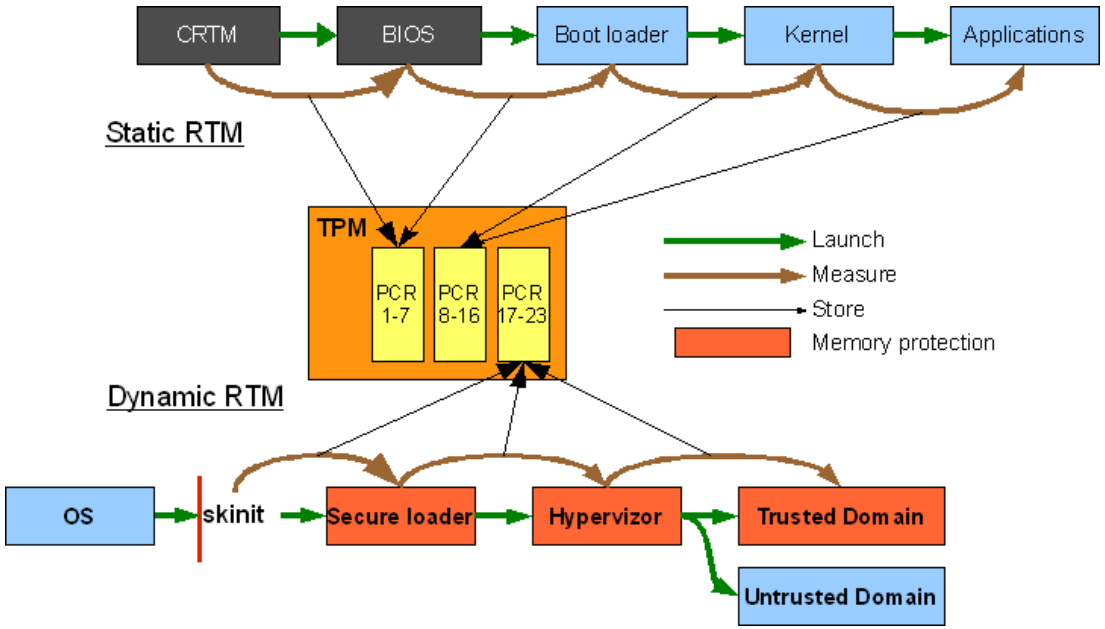
Mécanismes sous-jacents à Intel TXT

- **DRTM**
- **Measured Launched Environement (MLE)** : permet la mesure d'un environnement d'exécution lors d'un DRTM
- **Verified Launch** : permet l'application de la politique de sécurité définie (LCP) en fonction de l'environnement mesuré lors d'un MLE
- **IOMMU** : mise en œuvre d'une MMU dédiée aux I/O afin de bloquer l'accès DMA aux zones mémoires du MLE
- **Memory Teardrop** : déclenchement de routines d'effacement sécurisé dans certaines situations (transition vers S3, erreurs pendant le DRTM), afin d'éviter de laisser des informations sensibles en mémoire

Le principe de DRTM

Dynamic Root of Trust for Measurement

- Censé repondre aux limitations du SRTM
- Une chaine de confiance initiée dynamiquement
- La racine de la chaine de confiance est constituée d'un Secure Loader



Le principe de DRTM

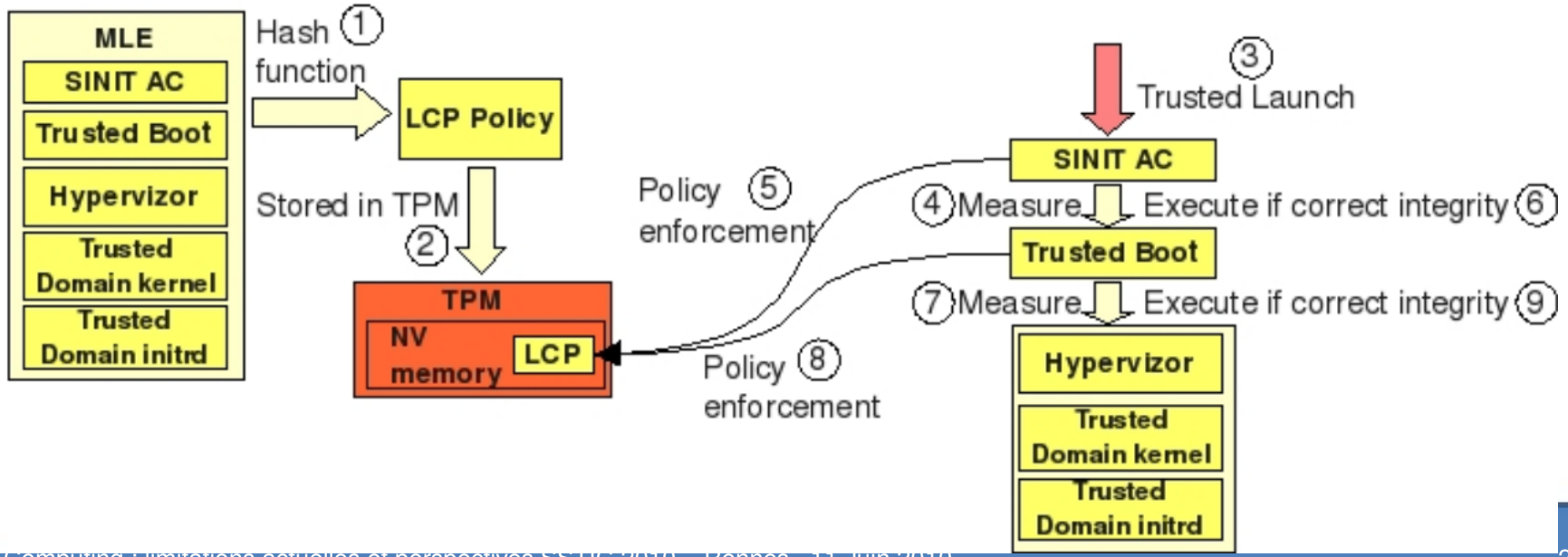
Le DRTM sous Intel

- Lors du lancement du DRTM, plusieurs protections sont mises en place :
  - Désactivation des cœurs/processeurs secondaires ; l'exécution est réalisée par le processeur BSP (*Boot-Strap-Processor*)
  - Le code est chargé dans le cache du processeur
  - Désactivation des interruptions (INIT, NMI et SMI comprises)
  - Activation de la protection IOMMU contre les accès DMA des périphériques
  - Blocage des mécanismes de débogage matériel
  - Communication avec le TPM sur la localité 4

Trusted Execution : Intel TXT et AMD SVM

Launch Control Policy (LCP)

- L'intégrité d'un état connu est sauvegardée dans une politique de sécurité (stockée dans le TPM)
- Au prochain démarrage (DRTM), une mesure d'intégrité est réalisée, puis comparée avec la politique
- Possibilité de continuer ou stopper l'exécution en cas d'intégrité non conforme



Attaques sur Intel TXT

Présentation des attaques

- Loïc Duflot et Olivier Levillain démontrent la possibilité de compromettre un TCB initié par TXT (DRTM et IOMMU) en s'aidant de routines ACPI ou SMI malicieuses
- Travaux équivalent de Rafal Wojtczuk et Joanna Rutkowska sur l'utilisation du vecteur SMM pour contourner les protections TXT

Analyse

- Difficulté d'avoir un environnement de confiance entièrement protégé sur les architectures x86
- Nécessité d'inclure les tables ACPI et les routines SMI dans le TCB

Trusted Execution : Perspectives

Trusted Execution : Perspectives

Afin de renforcer la sécurité d'un poste local, il est pourtant possible de puiser dans les technologies d'informatique de confiance pour assurer :

L'intégrité du système pendant son exécution. Justifications :

- Protection du noyau, des modules, des exécutables, des bibliothèques, des fichiers de données, etc.
- Protection contre la menace des codes malveillants et des accès illégitimes

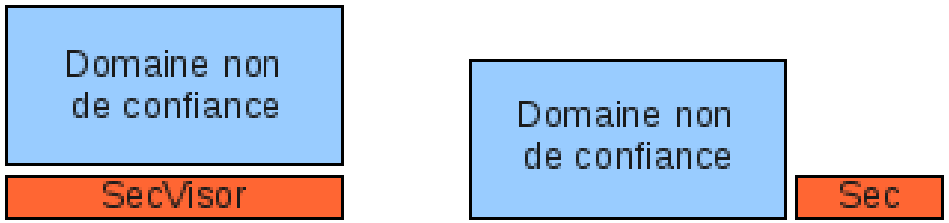
La confidentialité d'exécution. Justifications :

- Exécution cloisonnée dans un environnement partagé (serveurs mutualisés, *Cloud Computing*)
- Exécution de processus sensibles (AV, algos crypto)
- Protection de la propriété intellectuelle (empêcher la rétro-ingénierie de programmes, protection d'œuvres)

Intégrité du système pendant l'exécution

Solutions existantes

- Intégrité des applications
  - IMA : mesure d'intégrité des exécutables au moment de leur lancement
  - EVM : vérification d'intégrité sur les mesures d'IMA
- Intégrité du noyau. Deux approches :
  - Hyperviseur et domaines virtuels surveillés (ex : SecVisor)
  - DRTM (et contrôle d'intégrité) à intervalles de temps réguliers



Travaux relatifs

- HIDS (Tripwire, Samhain)
- Hyperviseur Hitux
- VMWare VMSafe

Intégrité du système pendant l'exécution

Hyperviseur SecVizor

- Développé par le laboratoire CMU Cylab
- Permet de vérifier plusieurs propriétés d'intégrité du noyau liées à
  - La cohérence des adresses mémoire pointées par le registre IP
  - La non-compromission de la mémoire du noyau
- Hyperviseur initié par un DRTM sur architecture AMD
- L'IOMMU protège :
  - Le noyau invité
  - L'hyperviseur SecVisor
  - Le vecteur DEV – Device Exclusion Vector
- Code de l'hyperviseur (5000 lignes) vérifié formellement

Intégrité du système pendant l'exécution

Analyse (1/2)

- L'emploi d'un hyperviseur minimaliste a deux intérêts :
  - Possibilité de le vérifier formellement
  - Exposition moindre aux vulnérabilités
- L'emploi d'un DRTM à intervalles de temps réguliers est séduisante du fait d'un TCB très minimaliste. Mais :
  - Comment assurer le lancement régulier du DRTM dans un environnement (noyau) compromis ?
  - Comment assurer l'intégrité du processus de vérification ?

Analyse (2/2)

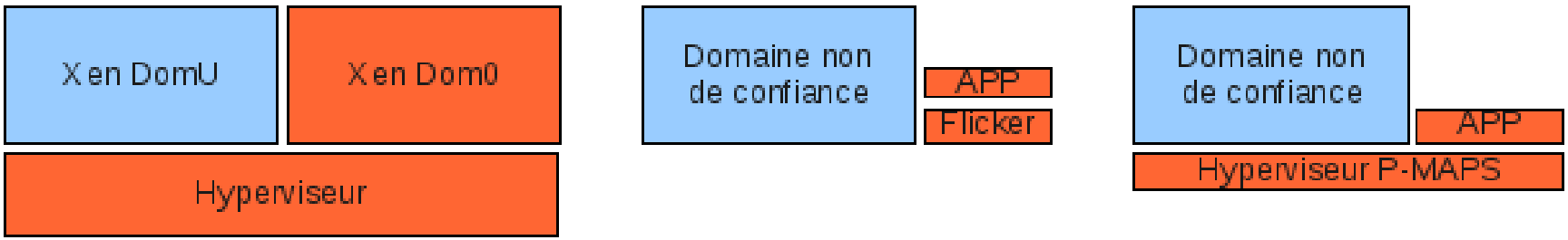
- Les mécanismes de DRTM et d'IOMMU permettent de compléter les technologies de virtualisation classiques afin de renforcer la sécurité du système invité
- L'hyperviseur apparait comme un environnement idéal pour la surveillance des noyaux invités
- Mais il est important d'éviter l'inflation de la taille de l'hyperviseur, sinon on ne fait que déporter le problème...
- Problématique générique : quid de la communication de confiance des résultats vers l'utilisateur ?
  - Tirer partie de l'attestation distante (lien avec TNC)
  - Utiliser un module tiers local (ex : une carte à puce avec afficheur intégré)
  - Bannière sécurisée
  - Intel Trusted I/O

Confidentialité d'exécution

Confidentialité d'exécution

Solutions existantes

- Deux modèles possibles :
  - Modèle à base de virtualisation (ex : TrustVisor, Intel P-MAPS)
  - Modèle à base de DRTM réguliers (ex : Flicker)



Travaux relatifs

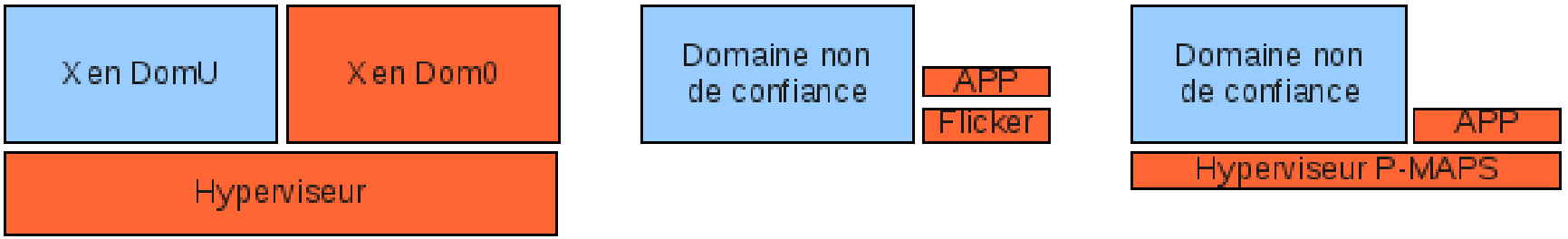
- Isolation par simple virtualisation (ex : VMWare, Virtualbox, Xen, KVM)
- Processeurs sécurisés du marché (ex : processeur Cell dans Xbox 360 et PS3)
- Processeurs sécurisés académiques (ex : Aegis, Hide, Dallas, CryptoPage)
- Co-processeurs cryptographiques (ex : carte IBM 4764)
- Cartes à puce

Confidentialité d'exécution

P-MAPS

- **Objectif** : exécution d'une application sensible de manière isolée de l'OS
- Projet d'Intel, similaire au projet TrustVisor de CMU Cylab
- Met en œuvre les mécanismes de DRTM et d'IOMMU sur Intel TXT
- **Fonctionnement** :
  - L'hyperviseur P-MAPS est initié par un DRTM afin d'assurer son intégrité
  - Un module P-MAPS est chargé dans l'OS invité
  - Les applications souhaitant s'exécuter de manière isolée de l'OS s'enregistrent auprès du module
  - Ensuite, lorsque ces applications accèdent à leur fonctions sensibles, une faute de page est provoquée
  - L'hyperviseur reprend la main et initie le mode d'exécution protégé : domaine virtuel dédié à l'application et isolé de l'OS invité et des accès DMA
  - Possibilité d'utiliser le TPM pour sceller des données lorsque l'OS invité récupère la main

Confidentialité d'exécution



Analyse

- Limitations du modèle Flicker
  - Temps de chargement important
  - Pas de parallélisme : l'exécution de Flicker met en pause l'OS
  - Ne peut pas être lancé en mode SMM (idée pourtant intéressante pour initier un DRTM indépendamment de l'OS)
  - Exposé à un déni de service si le noyau invité est compromis
- Limitations du modèle P-MAPS
  - Temps de chargement de l'hyperviseur important (mais temps de "basculement" acceptable)
- Problématique générique
  - Le TPM agit comme goulot d'étranglement, du fait de la lenteur des opérations cryptographiques

Conclusion

Conclusion

- Nécessité et possibilité de couvrir d'autres besoins de sécurité (intégrité du système pendant l'exécution et confidentialité d'exécution)
- Intel TXT apparait comme très prometteur pour renforcer la sécurité d'un poste local (soit du noyau, soit vis-à-vis du noyau)
- Intel TXT offre la possibilité de nouveaux types d'architectures de sécurité (modèles Flicker et P-MAPS)
- Mais technologie non encore mature :
  - Intel doit résoudre le risque lié au mode SMM (STM ?)
  - *Problème* de la validation des tables ACPI
  - Nécessité d'améliorer les performances du TPM (TPM.next devrait répondre à cette attente)
  - Problématique de la vérification "locale" d'intégrité et de la communication de confiance vers l'utilisateur

Questions ?

Annexe

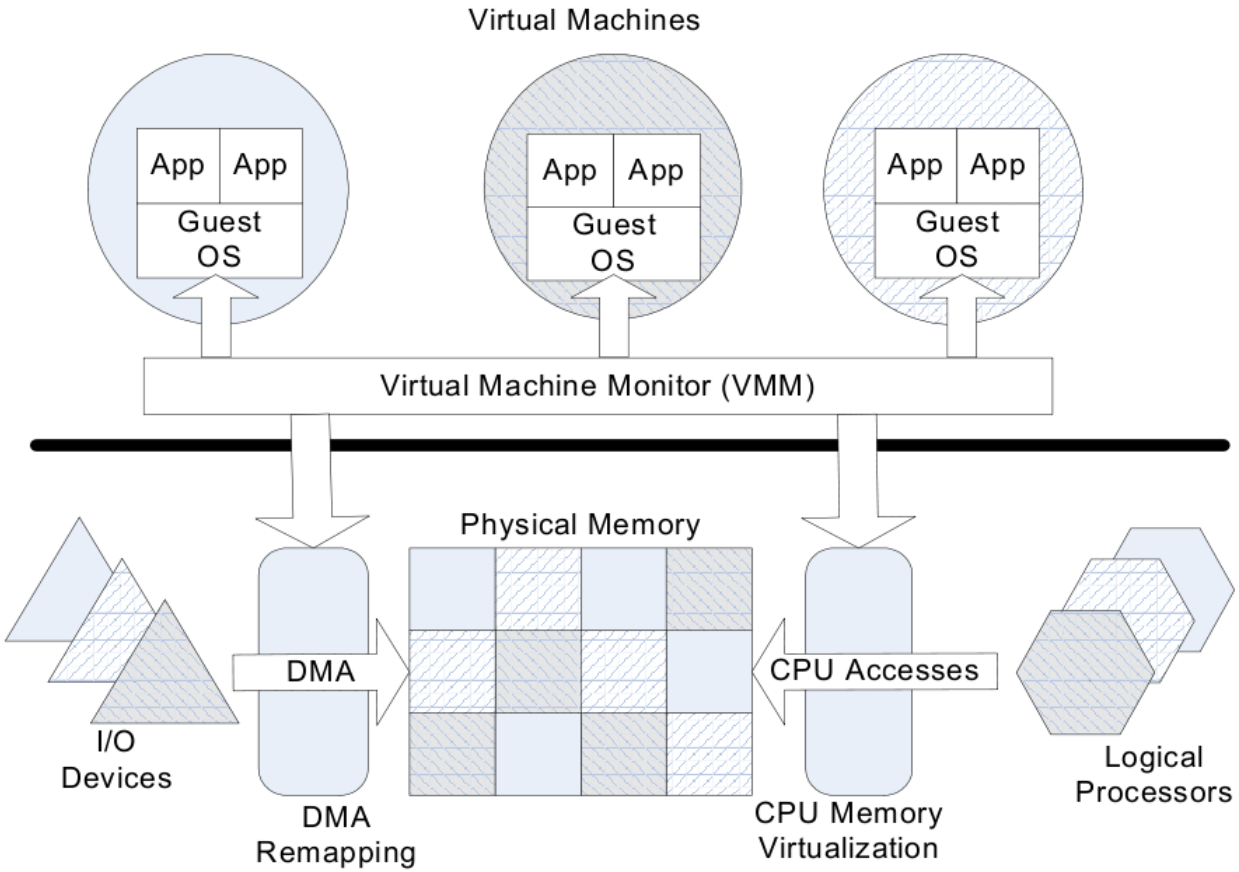


Figure 2-4. Interaction Between I/O and Processor Virtualization

Source : "Intel® Virtualization Technology for Directed I/O—Overview"