CentraleSupélec

Conception et implémentation d'un langage dédié à l'introspection d'une machine virtuelle

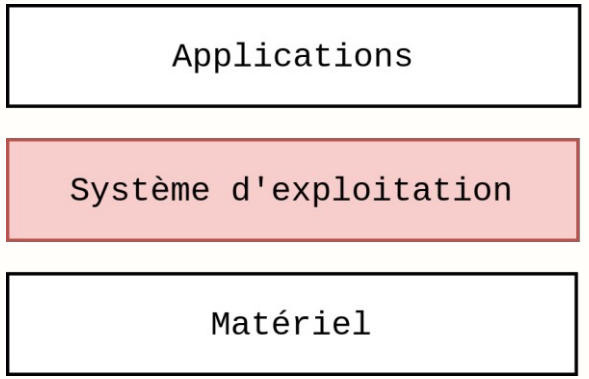
Lionel HEMMERLE

Équipe : CIDRE

Contexte : Intrusions

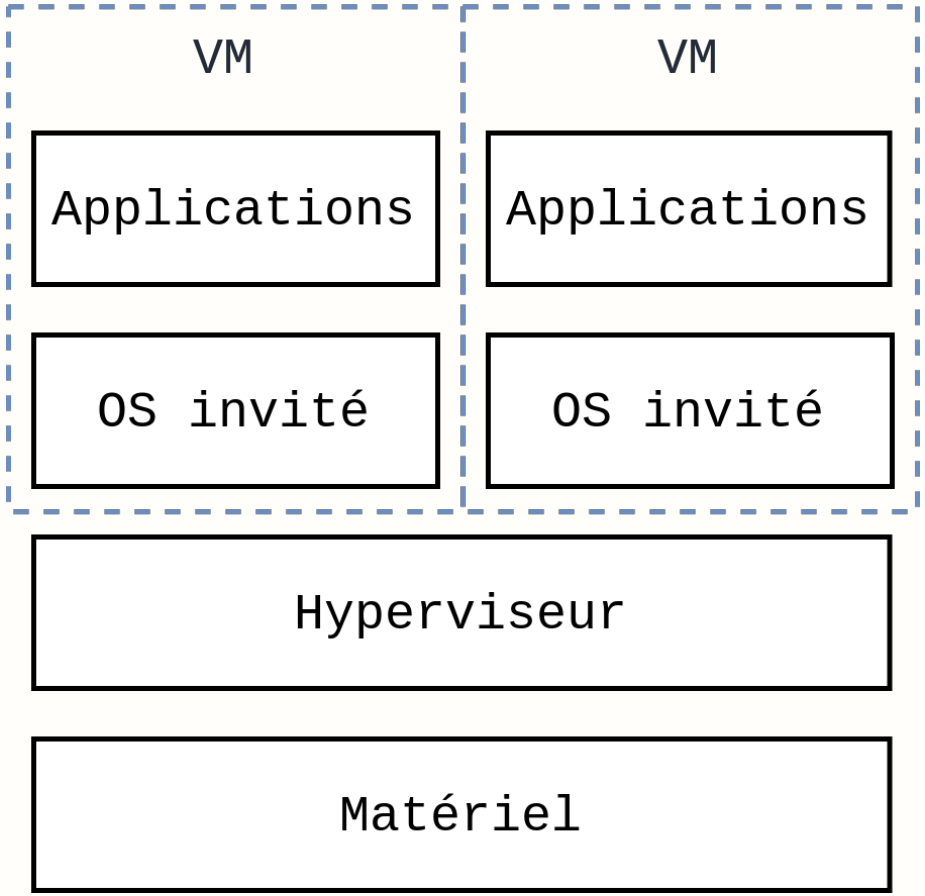
- **Comment détecter des attaques quand l'OS est compromis**
  - L'attaquant peut communiquer des informations fausses aux autres programmes
  - Il peut désactiver les IDS

- **On ne peut pas dépendre des données fournies par l'OS**



Contexte : Virtualisation

- **Extension de virtualisation** :
  - Permet d'exécuter plusieurs Machines Virtuelles (VM)
  - Un logiciel nommé hyperviseur supervise ces VM
- **Avantages :**
  - Les VM sont isolées les unes des autres
  - Même si l'attaquant contrôle entièrement une VM, l'hyperviseur reste sécurisé
  - On peut placer l'IDS au niveau de l'hyperviseur pour le protéger
  - Extensions très répandues



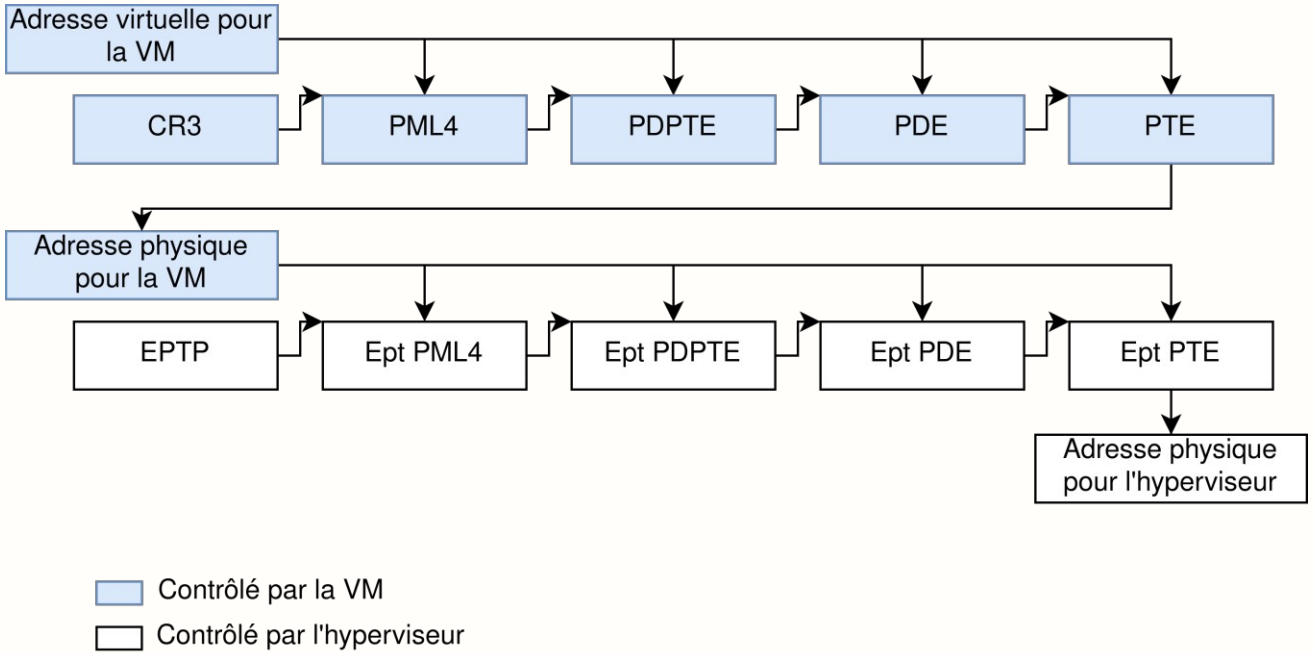
Contexte : Fossé sémantique

- **Fossé sémantique : Comment retrouver les informations dans la mémoire de la VM**
  - Comment elles sont stockées ?
  - Où elles sont stockées ?
  - Quand elles sont modifiées ?

- **Ex : liste des processus**

Contexte : Fossé sémantique

- **Exemple : Où sont stockées les données ?**
  - Mécanisme de traduction d'adresse partiellement effectué par la VM



Approche intéressante : Lares

- **Lares**
  - Intègre des hooks dans le code de la VM
  - Ils permettent de détecter des événements particuliers (appels systèmes, …)
  - Quand ils se déclenchent, ils communiquent des informations à l'hyperviseur

- **Inconvénient**
  - Les hooks doivent être protégés pour ne pas être désactivés

Bryan D. Payne et al. « Lares: An Architecture for Secure Active Monitoring Using Virtualization ». 2008 IEEE Symposium on Security and Privacy

Approche intéressante : VMI-PL:

- **Langage dédié à l'introspection de VM**
  - L'hyperviseur exécute des programmes fournis par l'utilisateur
  - Permet de faire de l'introspection sans se soucier du fonctionnement de la VM ou de l'hyperviseur

- **Inconvénient**
  - L'hyperviseur a besoin de fichiers de configuration externes permettant de franchir le fossé sémantique pour chaque OS sur lequel on souhaite exécuter les programmes

Florian Westphal et al. « VMI-PL: A monitoring language for virtual platforms using virtual machine introspection ». Digital Investigation 11 (2014)

Approche intéressante : Hyperupcalls

- **Les VMs envoient des programmes eBPF à l'hyperviseur**
  - L'hyperviseur exécute ces programmes
  - Ils permettent d'avoir des informations sur le fonctionnement de la VM
  - Utiliser pour que l'hyperviseur utilise mieux les ressources matérielles

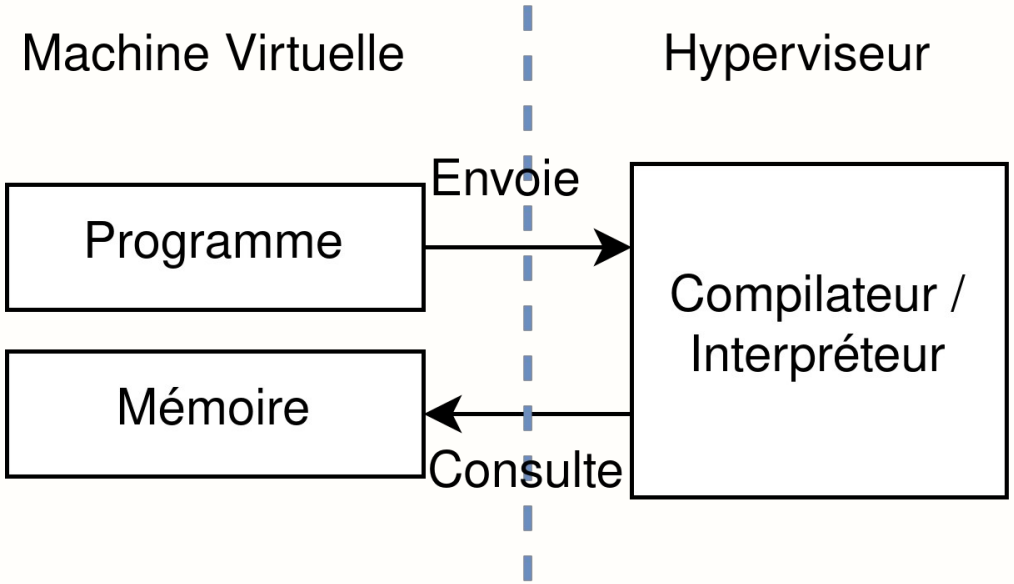
- **Inconvénient**
  - Pas assez protégé pour faire de la sécurité
  - Les programmes peuvent être envoyés n'importe quand par la VM
  - Un attaquant peut modifier l'agencement des structures en mémoire pour désactiver les programmes

Michael Wei et Nadav Amit. « Leveraging Hyperupcalls To Bridge The Semantic Gap: An Application Perspective ». IEEE Data Eng. Bull. (2019).

Objectifs de la thèse

- **Créer un langage dédié à l'introspection**
  - Les VM fournissent des programmes
  - L'hyperviseur les compile et les exécute
  - Ils permettent alors de lever des alertes pour détecter des attaques

- **Premiers objectifs (stage)**
  - Déterminer quelles attaques peuvent être détectées
  - Déterminer comment les détecter depuis un hyperviseur
  - Déduire des fonctionnalités du langage



Hypothèses

- **L'attaquant peut compromettre intégralement une VM, incluant l'OS**
  - On se limite à la détection de rootkits modifiant le noyau de l'OS
- **L'hyperviseur est considéré de confiance**
- **Les VM disposent d'une période initiale où elles sont sécurisées**

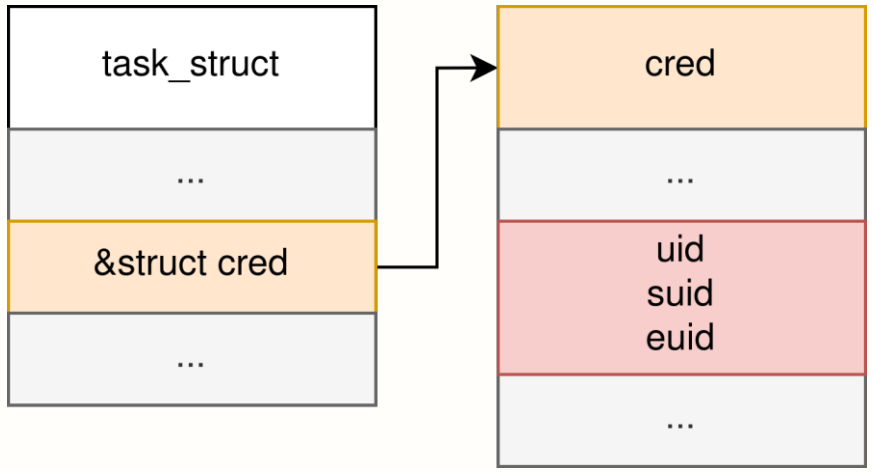
Attaques implémentées

- **Masquer un processus :**
  - Manipulation de la table d'appel systèmes
  - Manipulation du système de fichier virtuel
  - Manipulation des PID

- **Élévation de privilège :**
  - **Changement des identifiants d'un processus**

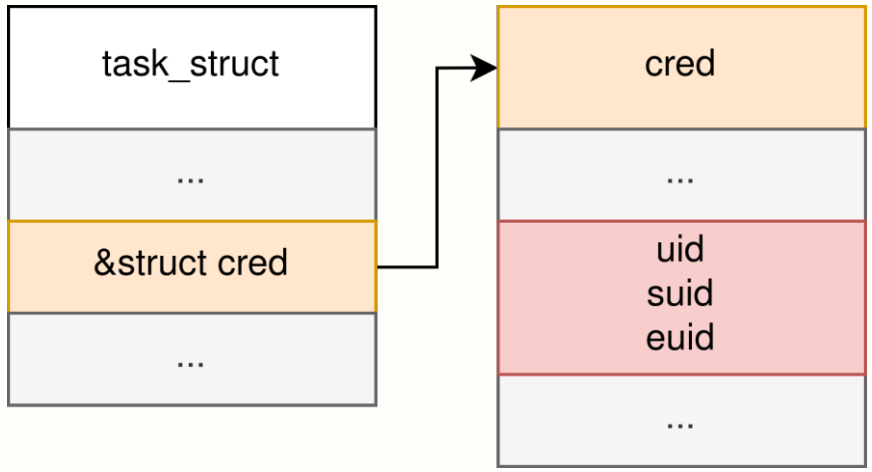
Élévation de privilège

- **Attaque :**
  - Modifier ou remplacer la structure cred
  - Le noyau considérera alors que le processus est exécuté par l'utilisateur root



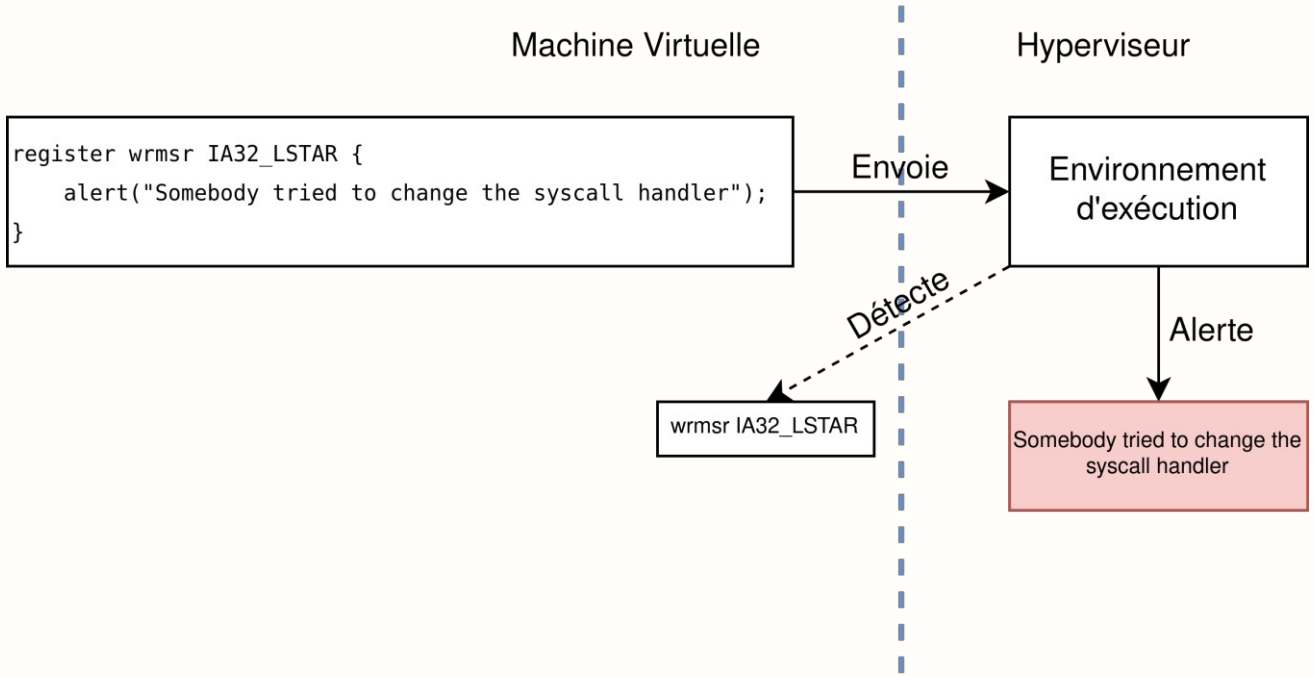
Élévation de privilège

- **Détection :**
  - Surveiller les modifications de la structure cred
  - Vérifier que ces modifications sont normales
- **Difficulté :**
  - Ces structures peuvent évoluer sans qu'une attaque ai lieu (par ex. avec l'appel système setreuid)



Événements détectables

- **C'est l'hyperviseur qui exécute les programmes fournis par la VM**
  - Il faut déterminer quels événements il peut détecter pour identifier les attaques



Événements détectables

Évènements générés par les attaques qui sont détectables au niveau de l'hyperviseur :

Attaques	Ecriture en mémoire	Modification registre MSR	Modification CR3	Appels de fonctions	Analyse périodique
Appels systèmes	X	X			
Système fichier	X				
PID	X		X	X	(X)
Identifiants	X		X	X	

Fonctionnalités du Langage

- **Se rattacher à des événements**
  - exécuter du code pour chaque événement
  - Pouvoir lire la mémoire de la VM
  - Lever une alerte si nécessaire
- **Stocker un état interne**
  - Exemples : liste des processus actifs, dernier appel système fait par un processus, …
- **Se rattacher dynamiquement à de nouveaux événements**
  - Par exemple : création d'un processus

Contraintes de sécurité

- **Envoi des programmes**
  - Un attaquant pourrait essayer d'envoyer ses propres programmes

- **Contremesure**
  - La VM est initialement considérée saine
  - La VM envoie un signal après avoir envoyé le dernier programme
  - Tout programme envoyé après est ignoré par l'hyperviseur

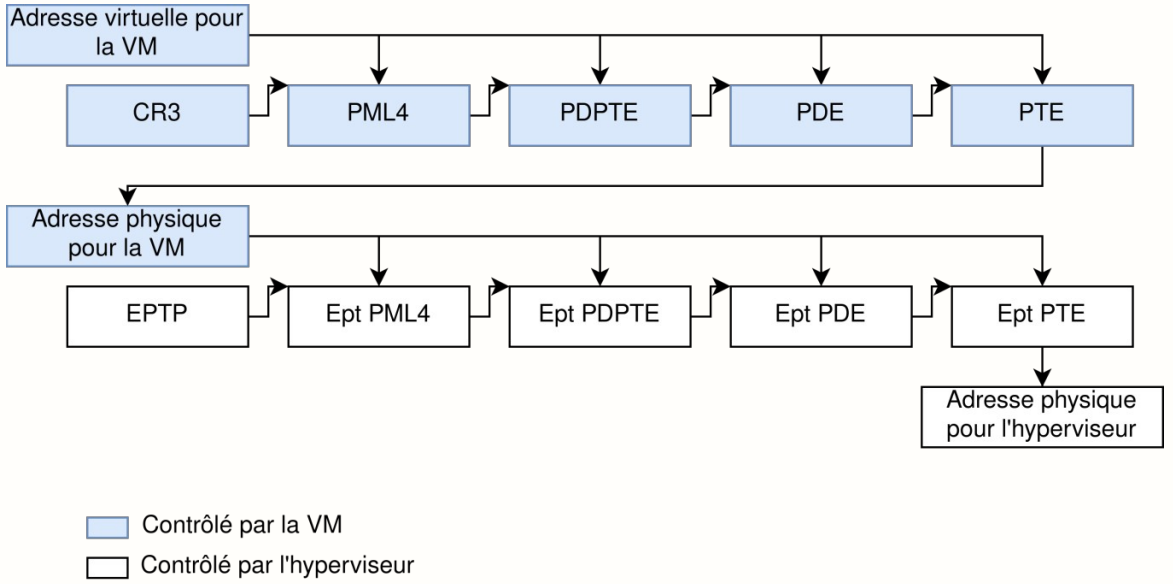
Contraintes de sécurité

- **Manipulation des entrées**
  - Un attaquant peut modifier la mémoire de la VM, donc les entrées des programmes exécutés
    - Ex : créer beaucoup de processus, …

- **Contremesures**
  - Garantir que les lectures et écritures en mémoire des programmes sont légales
  - Empêcher les boucles infinies (en ajoutant un compteur à chaque boucle)
  - Limiter le nombre total d'événement pouvant être créés

Contraintes de sécurité

- **Désactivation des événements**
  - Un attaquant peut modifier les structures liées au mécanisme de traduction d'adresse pour changer les adresses physiques des zones de la mémoire surveillées
  - L'hyperviseur ne surveille alors plus les bonnes zones de la mémoire
- **Contremesure**
  - Surveiller les structures liées à la traduction d'adresse à chaque changement de contexte



Conclusion

- **Stage :**
  - Implémentation et détection d'attaques (dump mémoire et hyperviseur)

- **Thèse :**
  - Création du langage
  - Implémenter pour la VM un module kernel envoyant les programmes
  - modification de XVisor
  - Implémenter un compilateur ou un interpréteur dans l'hyperviseur