Projet ANR SPACLik

Jérémy Briffaut, Martial Szpieg, Christian Toinard

Equipe Sécurité et Distribution des Systèmes

ENSI Bourges - LIFO

Plan

Conception d'un OS sécurisé

- Système d'exploitation basé sur Linux
- Objectifs :
  - Protéger le système des erreurs/abus de l'utilisateur
  - Protéger l'utilisateur contre l'exploitation de failles présentes dans ses applications
- Mise en oeuvre :
  1) Appliquer les patchs de sécurité connus
  2) Utiliser du Contrôle d'Accès Mandataire
  3) Définir et garantir des Propriétés de Sécurité

Principes de l'OS sécurisé (1)

- Renforcement de la sécurité via une chaîne de compilation sécuritaire
  - Basé sur une distribution Gentoo Hardened
  - GCC + Glibc renforcés
    - Options de compilation
      - -fstack-protector-all
      - -Wformat -Wformat-security
      - -D_FORTIFY_SOURCE=2
    - Patchs de sécurité
      - **PIE** : Position Independant Executable
      - **SSP** : Stack Smashing Protector
  - Système entièrement recompilé

Principes de l'OS sécurisé (2)

- Renforcement de la sécurité par un système de contrôle d'accès mandataire
  - SELinux
    - Politique **strict** et en mode **enforcing**
    - Politique modifiée pour fonctionner avec X
- Définition de modules SELinux spécifiques
  - Firefox
  - Clawsmail
  - OpenOffice

Principes de l'OS sécurisé (3)

- Noyau spécifique
  - Patch grsecurity&PaX
    - Protection contre la mémoire exécutable
    - Divers correctifs de sécurité
  - Patch Dakuzo
    - Recherche de virus lors d'accès aux fichiers
    - Partie userland : clamav
  - **Patch PIGA-Kernel**
    - Contrôle de propriétés de sécurité
    - Partie userland : PIGA-UM

Plan

I. Introduction
    I. Généralité
    II. Principes de l'OS sécurisé
II. Description du Système
    I. Système de base
    II. Noyau
    III. Environnement utilisateur
III. Garantie de propriétés de sécurité : PIGA-SP
    I. Principe général
    II. Partie Noyau : PIGA-Kernel
    III. Partie Userland : PIGA-UM
    IV. Exemple de propriétés définies
IV. Conclusion

Système de Base

- Distribution GNU/Linux Gentoo
  - Variante LFS (Linux From Scratch)
  - Suite GCC/Glibc hardened
    - Compilation du système avec cette suite
  - Aucun service superflus installé
  - Pare-feu iptables
    - Packets sortant : Web Mail DNS
    - Packets entrant : rien

Noyau

- Noyau 2.6.25
  - Branche gentoo-hardened
    - Noyau vanilla + patchs de sécurité
  - Monolithique
  - SELinux activé
  - Patchs appliqués
    - PaX & grsecurity
      - Toutes les options activées
    - Dakuzo
      - Fonctionnant avec clamav
    - PIGA-Kernel

Environnement Utilisateur

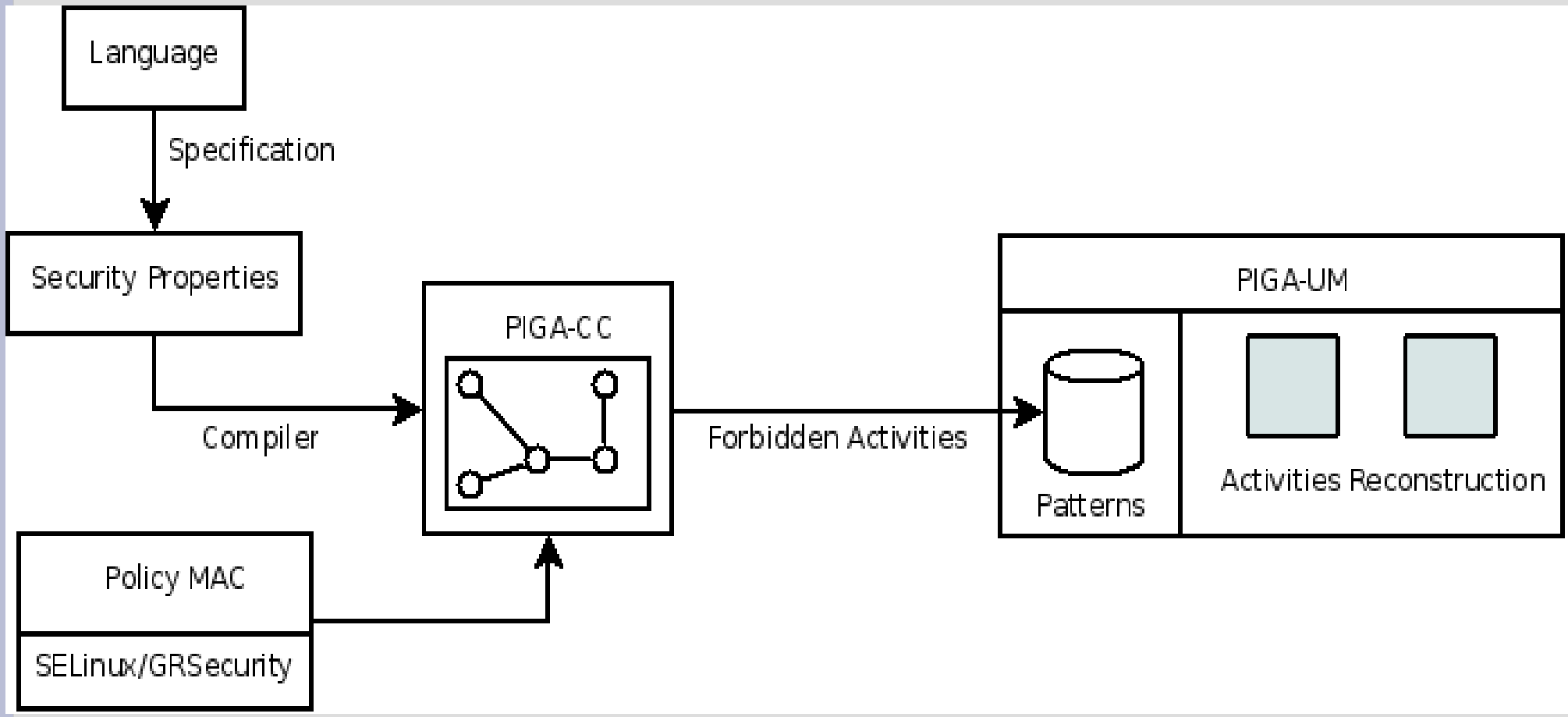
- Environnement graphique LXDE
  - Simple
  - Léger
- Applications :
  - Firefox
  - Clawsmail
  - OpenOffice

Plan

Principe Général

- Garantir des propriétés de sécurité ...
  - Confidentialité/Intégrité/Abus de privilèges
- … via un langage de haut-niveau
  - Spécification aisée des besoins de sécurité
    - Couvre aisément toutes les propriétés de la littérature
    - Permet de spécifier de nouvelles propriétés
- Contrôle mandataire des propriétés
  - PIGA-Kernel : contrôle des appels système
  - PIGA-UM : prise de décision en espace utilisateur

Fonctionnement Général



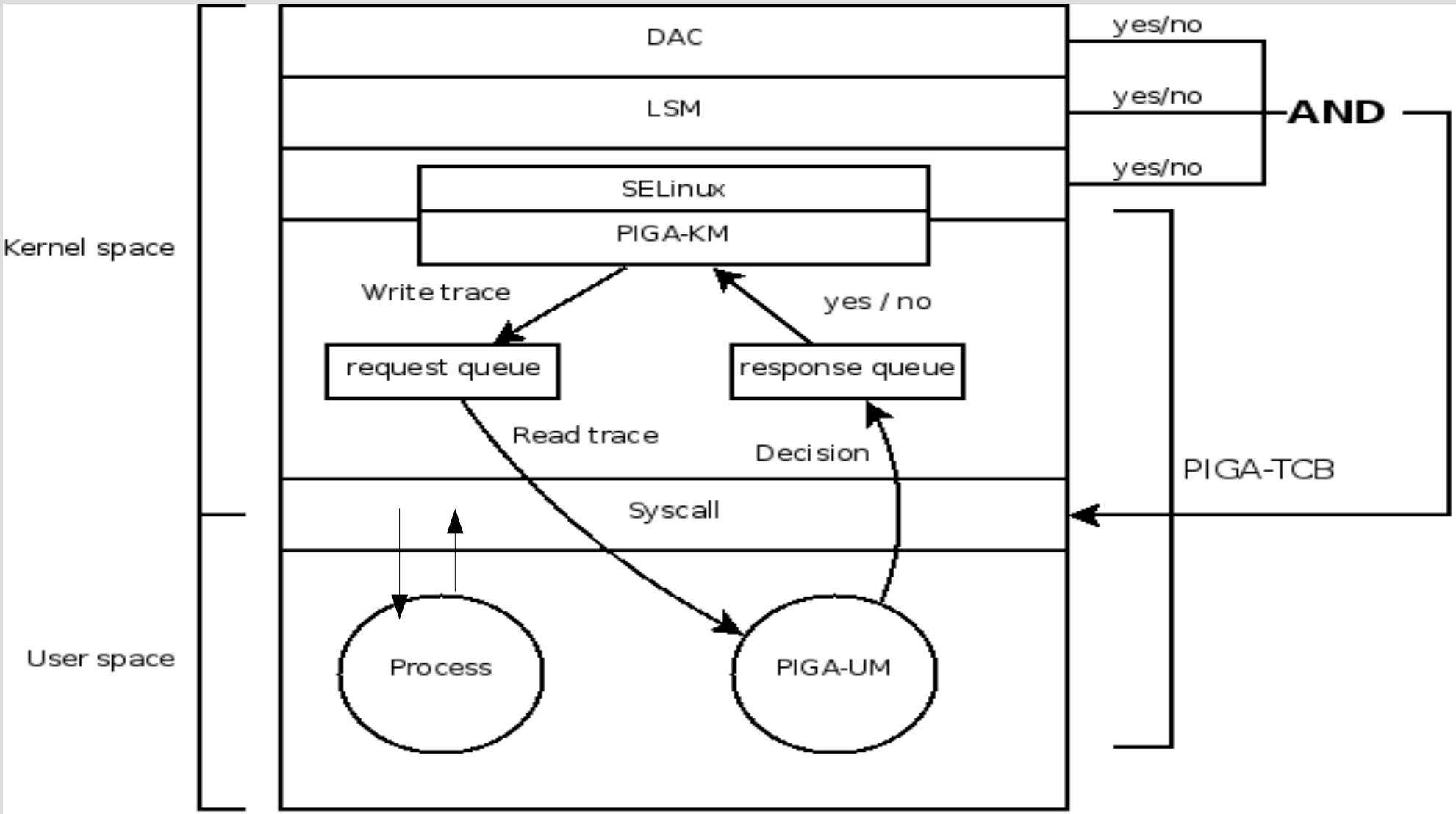
Partie Noyau : PIGA-Kernel

- Ajoute un contrôle mandataire en supplément du DAC Linux et du MAC SELinux
  - Détourne les appels système (hooks LSM)
  - Envoi des informations (traces) à la partie userland (PIGA-UM)
  - Applique les décisions prises par PIGA-UM
- Patch Noyau (en C), s'intègre dans LSM
- 2 modes de fonctionnement
  - Permissif (pour le développement/test)
  - Actif (pour l'OS livré)

Partie Userland : PIGA-UM

- Contrôle du respect des propriétés de sécurité
  - Reconstruction des activités en cours
  - Vérification de la légalité de ces activités
  - Pour chaque trace d'interaction envoyée par PIGA-Kernel :
    - autorise/interdit l'interaction
- Application codée en Java (60 000 lignes)
- S'installe sous forme d'un service système

Interaction entre PIGA-Kernel & PIGA-UM



Langage de spécification de propriétés de sécurité

- Terminaux du langage :
  - Interaction : un appel système
  - Séquence : suite causale d'appels systèmes
- Opérateurs
  - Combinaisons des terminaux
  - Structures de contrôle
- Supporte la définition de canevas
- Permet la réutilisation

Exemple de propriétés de sécurité

- Canevas de séparation des privilèges de modification et d'exécution

```
def ne dutiesseparation( $sc1 IN CS ) [
      Foreach $eo1 IN IS, Foreach $eo2 IN IS, Foreach $sc2 IN CS
            SuchThat { ($sc1 -> { $eo2 } $sc2 o $sc1 -> { $eo1 } $sc2)  } ,
                  { not( (is_write_like($eo1) AND is_execute_like($eo2)) ) };
      Foreach $eo1 IN IS, Foreach $eo2 IN IS, Foreach $sc2 IN CS
            SuchThat { ($sc1 => { $eo2 } $sc2 o $sc1 => { $eo1 } $sc2)  } ,
                  { not( (is_write_like($eo1) AND is_execute_like($eo2)) ) };
];
```

- Utilisation

```
dutiesseparation( $sc1:=user_u:user_r:user_t );
```

Plan

Conclusion

- La première version fournit :
  - Un système linux + une interface graphique sécurisé
  - Le contrôle des propriétés de sécurité au niveau noyau Linux
    - Patch Noyau + Application userland
  - Quelques outils graphiques (firefox, clawsmails, ooffice, …)
    - Module SELinux + propriétés dédiées