SIARS v2

Bonnes Pratiques de sécurité GNU/Linux

Matthieu Herrb

6 décembre 2018

Agenda

1. Introduction - Règles générales
2. Protection système
   - SELinux & AppArmor
   - Systemd
3. Sécurité des postes de travail
4. Conclusions

Agenda

Introduction

Sécurisation de systèmes GNU/Linux :

- postes de travail
- serveurs

Plusieurs voies :

- règles générales / bonnes pratiques
- guides spécifiques à une distribution
- outils dédiés (SELinux / systemd / AppArmor / seccomp)

Guides

- RedHat 7 Security Guide
- How to protection CentOS
- Securing Debian Manual
- Ubuntu Basic Security
- Ubuntu Server Guide - Security
- ANSSI - Recommandations de sécurité relatives à un système GNU/Linux
- ANSSI - Guide d'hygiène informatique
- Règles de sécurité de la PSSI du CNRS (EXP-CNF-3)

Principes

Il est fondamental de mettre en place un niveau de sécurité minimal sur l'ensemble du parc informatique de l'entité en implémentant les mesures suivantes :

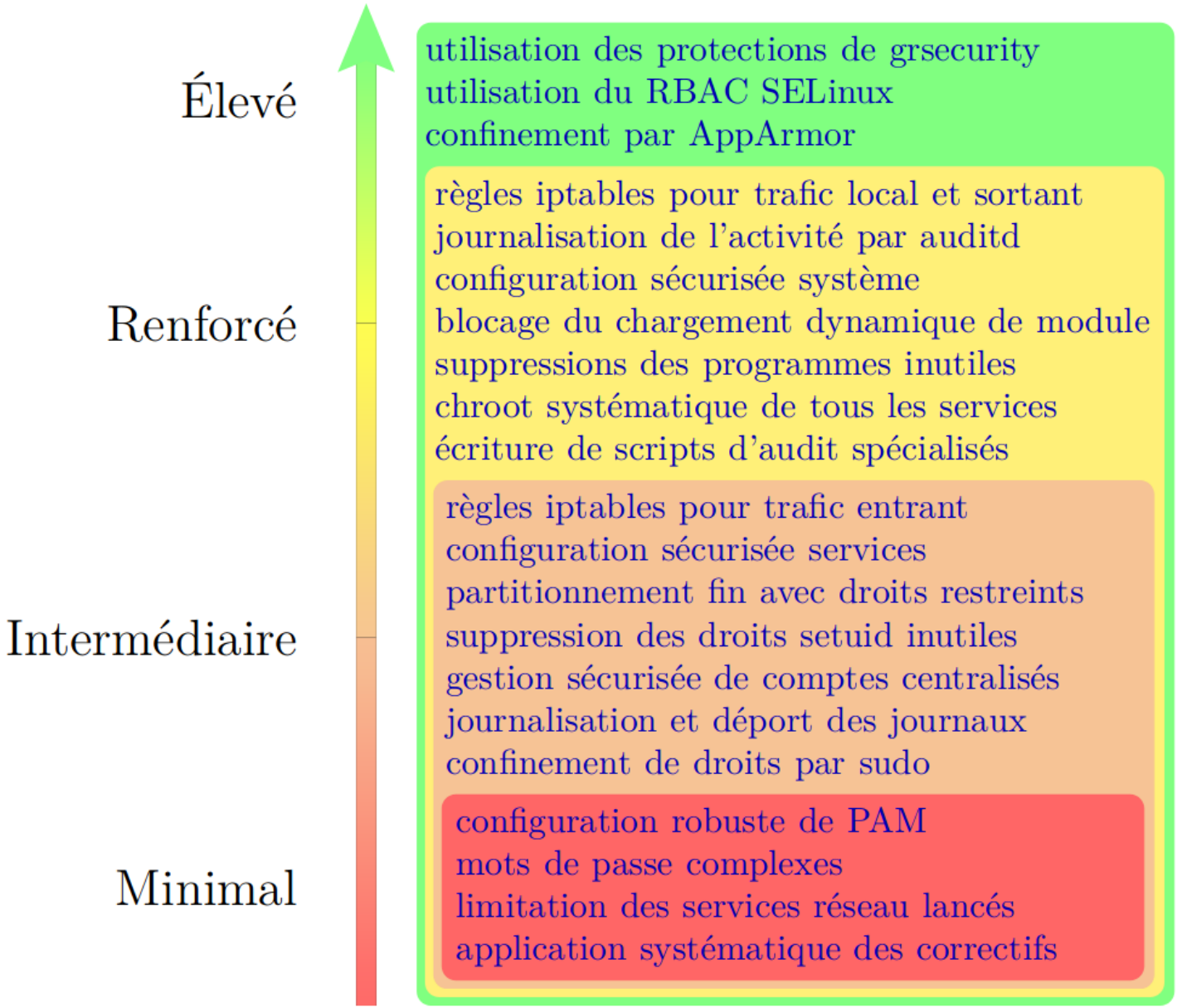
- limiter les applications installées... ;
- doter les postes utilisateurs d'un pare-feu local et d'un anti-virus ;
- chiffrer les partitions où sont stockées les données des utilisateurs ;
- désactiver les exécutions automatiques (autorun).

Principes

- Réduire la surface d'attaque.
- Homogénéiser les installations.
- Permettre mises à jour et suivi.
- Séparer les services.

Utiliser système d'orchestration (Ansible, Puppet, SaltStack)

Niveaux de sécurité



Minimisation des services installés

Seuls les composants strictement nécessaires au service rendu doivent être installés

- Tout service est un élément sensible
  en particulier si écoute sur le réseau
- Seuls ceux connus et requis doivent être présents
- Ceux dont la présence n'est pas justifiée doivent être désinstallés

Défense en profondeur

Combinaison de barrières, indépendantes les unes des autres :

- Filtrage réseau en entrée
- Authentification
- Cloisonnement
- Mécanismes de prévention de l'exploitation des vulnérabilités :
  - ASLR, W^X, canaris & co
  - Filtrage réseau en sortie
  - Bac à sable (Sandbox)
- Journalisation sécurisée

Mises à jour régulières

Procédures de mises à jour de sécurité réactives

- information $\rightarrow$ listes de diffusion (CERT, éditeurs, etc.)
- application automatisée
  - procédure de test (pré-prod) / validation
  - sauvegardes (snapshot) pour retour arrière si besoin
- contrôle (audit) de l'application des mises à jour

Choix des dépôts des paquets

Règle :

Seuls les dépôts officiels à jour de la distribution doivent être utilisés

Si nécessaire d'installer des paquets de dépôts tiers :

- vérifier la provenance,
- vérifier les mises à jour

Robustesse de mot de passe administrateur

- Mot de passe suffisamment robuste (ANSSI)
- Mot de passe unique et propre à chaque machine

Version sans mot de passe administrateur :

- utilisation de sudo en local (avec un compte local dédié)
- utilisation de clés publiques SSH pour connexions à distance

Bannir le sudo sans mot de passe pour les administrateurs.

Applications utilisant PAM

Règle :

Le nombre d'applications utilisant PAM doit être réduit au strict nécessaire afin de limiter l'exposition d'éléments d'authentification sensibles.

- privilégier l'authentification via LDAP / Radius / CAS / Shibboleth,...
- n'exposer que les comptes utiles

Protection des mots de passe stockés

Règle :

Tout mot de passe doit être protégé par des mécanismes cryptographiques évitant de les exposer en clair à un attaquant récupérant leur base.

- hachage du mot de passe avec une fonction de hachage sûre (SHA256,...), un sel et suffisamment d'itérations
- chiffrement par une clé secrète (éventuellement protégée par un HSM)

Exécutables avec bits setuid et setgid

Règle :

Seuls les programmes spécifiquement conçus pour être utilisés avec les bits setuid (ou setgid) peuvent avoir ces bits positionnées.

- vérifier la liste des exécutables setuid

```
find . -type -f \( -perm -2000 -o -perm -4000 \)
```

- si possible monter les partitions (/home par ex.) avec l'option nosuid

Services et démons actifs

Règle :

Seuls les démons réseau strictement nécessaires au fonctionnement du système et du service doivent être résidents et n'être en écoute que sur les interfaces réseau adéquates.

- écouter sur un socket Unix (ou sur ::1) pour les services locaux (base de données, redis, etc.)
- vérification des services en écoute avec la commande :

```
netstat -l -A inet -A inet6
```

- liste des processus associés via `lsof`

Sécurisation des journaux

- centralisation sur un serveur dédié
- via rsyslog / syslog-ng
- via systemd-journal-upload/journald-remote
- séparer l'exploitation des logs (elk) de leur archivage

Systèmes de fichiers distants (NFS)

- limiter/contrôler les exportations
- utiliser `root_squash`
- centraliser les utilisateurs
- authentification kerberos si possible
- attention : trafic en clair - protéger via IPsec si nécessaire
- montage `nosuid`

Agenda

1. Introduction - Règles générales
2. Protection système
   - SELinux & AppArmor
   - Systemd
3. Sécurité des postes de travail
4. Conclusions

Agenda

SELinux - Présentation

Security Enhanced Linux est un système de contrôle d'accès obligatoire (MAC).

- Développé initialement par la NSA puis RedHat
- Module noyau qui s'interpose sur les appels système
- Ensemble de règles d'autorisation manipulant des sujets, des objets des rôles et des contextes.

Permettre un contrôle plus fin que les droits Unix classique, y compris pour `root`.

États de SELinux

Trois états possibles :

- **disabled** - non activé
- **permissive** - journalise les règles sans les appliquer
- **enforcing** - applique les règles

```
# getenforce
Disabled
# sestatus
SELinux status: disabled
```

Configuration dans `/etc/sysconfig/selinux` (reboot)

SELinux en pratique

1. activer selinux en mode **permissive**
2. tester son service
3. récupérer les erreurs dans `/var/log/audit/audit.log`
4. la commande `audit2allow(1)` permet de créer une politique pour l'utilisation normale du service
5. charger la nouvelle politique : semodule -i new.pp
6. itérer (2, 3, 4, 5) jusqu'à plus d'erreur
7. passer en mode **enforced**
8. mettre en production

AppArmor - Présentation

AppArmor est un système de contrôle d'accès obligatoire (MAC).

- développé par Canonical (Ubuntu)
- alternative « plus simple » à SELinux
- modes de fonctionnement similaires à SELinux : *enforcement* et *complain*
- politiques en mode texte dans `/etc/apparmor.d/`

AppArmor - Commandes

- État d'AppArmor

```
sudo aa-status
```

- Passer une politique en mode complain

```
sudo aa-complain /usr/bin/evince
```

- Passer une politique en mode enforce

```
sudo aa-enforce /usr/bin/evince
```

- Recharger une politique

```
sudo apparmor_parser -r /etc/apparmor.d/usr.bin.evince
```

- Désactiver AppArmor

```
sudo systemctl stop apparmor
sudo systemctl disable apparmor
```

Agenda

systemd et la sécurité

Systemd permet de :

- Tirer profit des fonctionnalités de sécurité du noyau Linux
- Pour simplifier le durcissement de la maintenance d'un système.

systemd - rappels

- Remplaçant de *SysVinit* intégré dans la plupart des distributions
- Chargé du *démarrage* et de la *gestion* des services système
- Remplace les *scripts d'init* par des fichiers de configuration déclaratifs : les **units**.

Unit

Afficher la configuration actuelle d'un service :

```
# systemctl cat php-fpm.service
```

```
# /usr/lib/systemd/system/php-fpm.service
[Unit]
Description=The PHP FastCGI Process Manager
After=syslog.target network.target

[Service]
Type=notify
PIDFile=/run/php-fpm/php-fpm.pid
EnvironmentFile=/etc/sysconfig/php-fpm
ExecStart=/usr/sbin/php-fpm --nodaemonize
ExecReload=/bin/kill -USR2 $MAINPID
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

Exemple : utilisateur et groupe non privilégiés

Éditer la configuration d'un service

```
# systemctl edit php-fpm.service
```

pour ajouter :

```
[service]
User=http
Group=www
```

et rendre les modifications effectives :

```
# systemctl daemon-reload
# systemctl restart php-fpm.service
```

Filtrage d'appels système avec seccomp-bpf

Concept

- Filtrage des appels système disponibles pour un processus
- S'applique aussi aux processus fils

Example

```
[Service]
SystemCallFilter=~chroot
SystemCallFilter=~@obsolete
```

À noter :

- Contournable sur les noyaux < 4.8 avec *ptrace*.
- Solution : filtrer l'appel système *ptrace* :

```
[Service]
SystemCallFilter=~ptrace
```

Capacités Linux

Concept

- Restriction des droits accordés à un processus (potentiellement) *root*
- Ajout de permissions à un processus non *root*

Example

```
[Service]
CapabilityBoundingSet=CAP_NET_BIND_SERVICE
AmbientCapabilities=CAP_NET_BIND_SERVICE
```

À noter

- Certaines capacités sont équivalentes à *root*
- Préférer une liste blanche des capacités réellement nécessaires

Espace de nom des points de montage

Concept

Arborescence du système de fichiers distincte pour chaque service

Example

```
[Service]
InaccessiblePaths=/etc/secrets
ProtectSystem=full
```

À noter

- Réversible si `CAP_SYS_SYSADMIN` :

```
[Service]
CapabilityBoundingSet=~CAP_SYS_SYSADMIN
SystemCallFilter=~@mount
```

Mise en pratique : Dirty CoW

- Vulnérabilité CVE-2016-5159
- Local *root* rendue publique en octobre 2016
- Présente depuis la version 2.6.22 du noyau (2007)
- Situation de compétition dans le mécanisme de *copy on write*

Réduire l'impact (1)

Vulnérabilité

Situation de compétition déclenchée avec l'appel système `madvise`

Options pour réduire l'impact

- Bloquer l'appel système `madvise`

Configuration

```
[Service]
SystemCallFilter=~madvise
```

Réduire l'impact (2)

Vulnérabilité

Utilisation de l'appel système `ptrace` et de `/proc/self/mem`

Options pour réduire l'impact

- bloquer l'appel système `ptrace`
- Supprimer l'accès au système de fichiers virtuel `/proc`

Configuration

```
[Service]
SystemCallFilter=~ptrace
InaccessiblePaths=~/proc
```

Réduire l'impact (3)

Vulnérabilité

Certains drivers de périphériques matériels potentiellement concernés

Options pour réduire l'impact

- Supprimer l'accès aux périphériques matériels exposés dans /dev

Configuration

```
[Service]
PrivateDevices=yes
```

Agenda

Problématique

Sécurité d'un poste de travail GNU/Linux au niveau de l'interface utilisateur

Clavier, souris,...

Accès aux événements des périphériques d'entrée

Raccourcis clavier ou keylogger ?

Copie d'écran

Screencast ou espionnage de l'affichage ?

Mode Plein écran

Video plein écran ou fausse interface ?

Problématique (2)

Alertes et notifications

Quelle est l'application qui demande un mot de passe ?
$\rightarrow$ authentification de l'origine des pop-ups

Copier/Coller

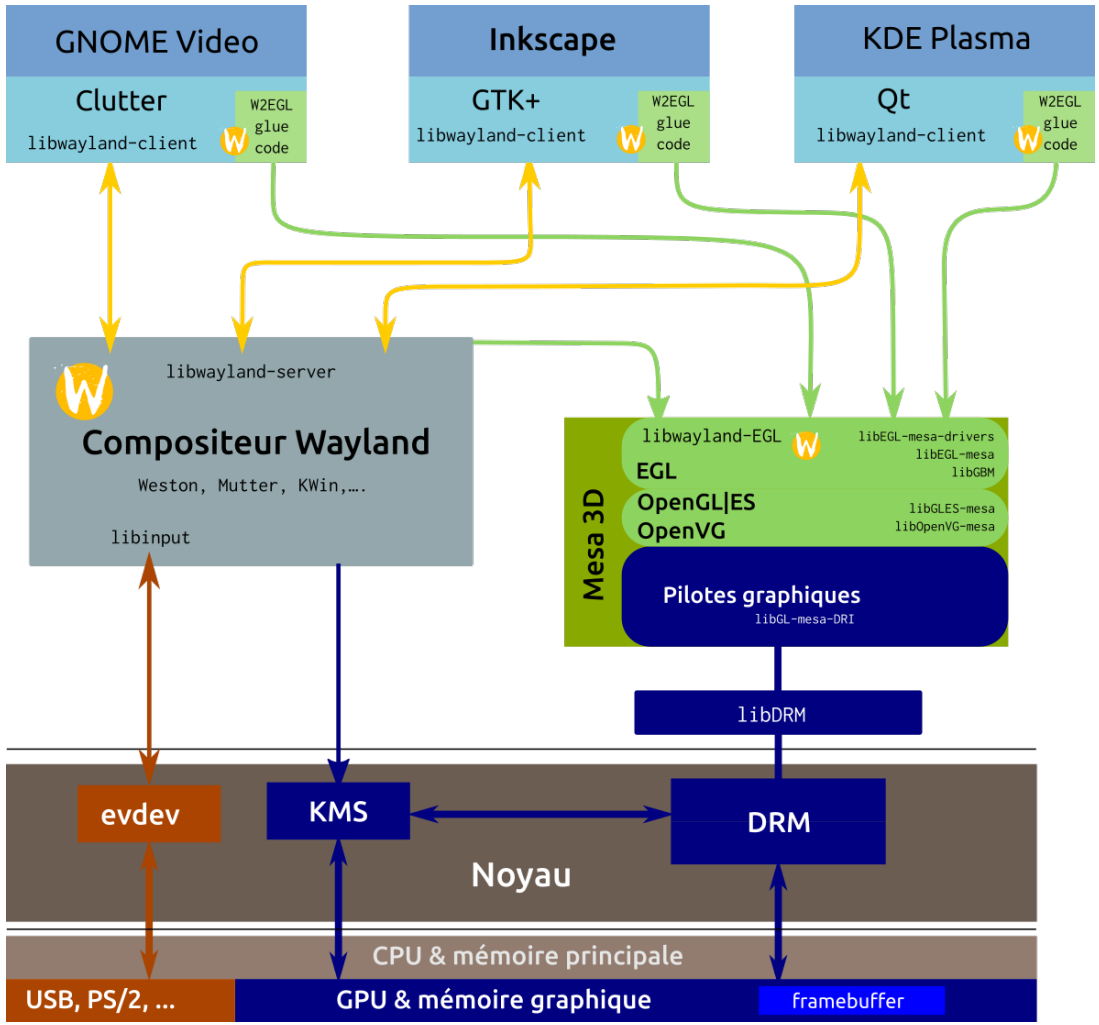
Qui peut accéder aux données du presse-papiers ?
S'il contient un mot de passe super secret ?

X Windows

- Communications inter-clients via ICCCM et EWMH
- Pas de sécurité : tous les clients ont accès à tout.

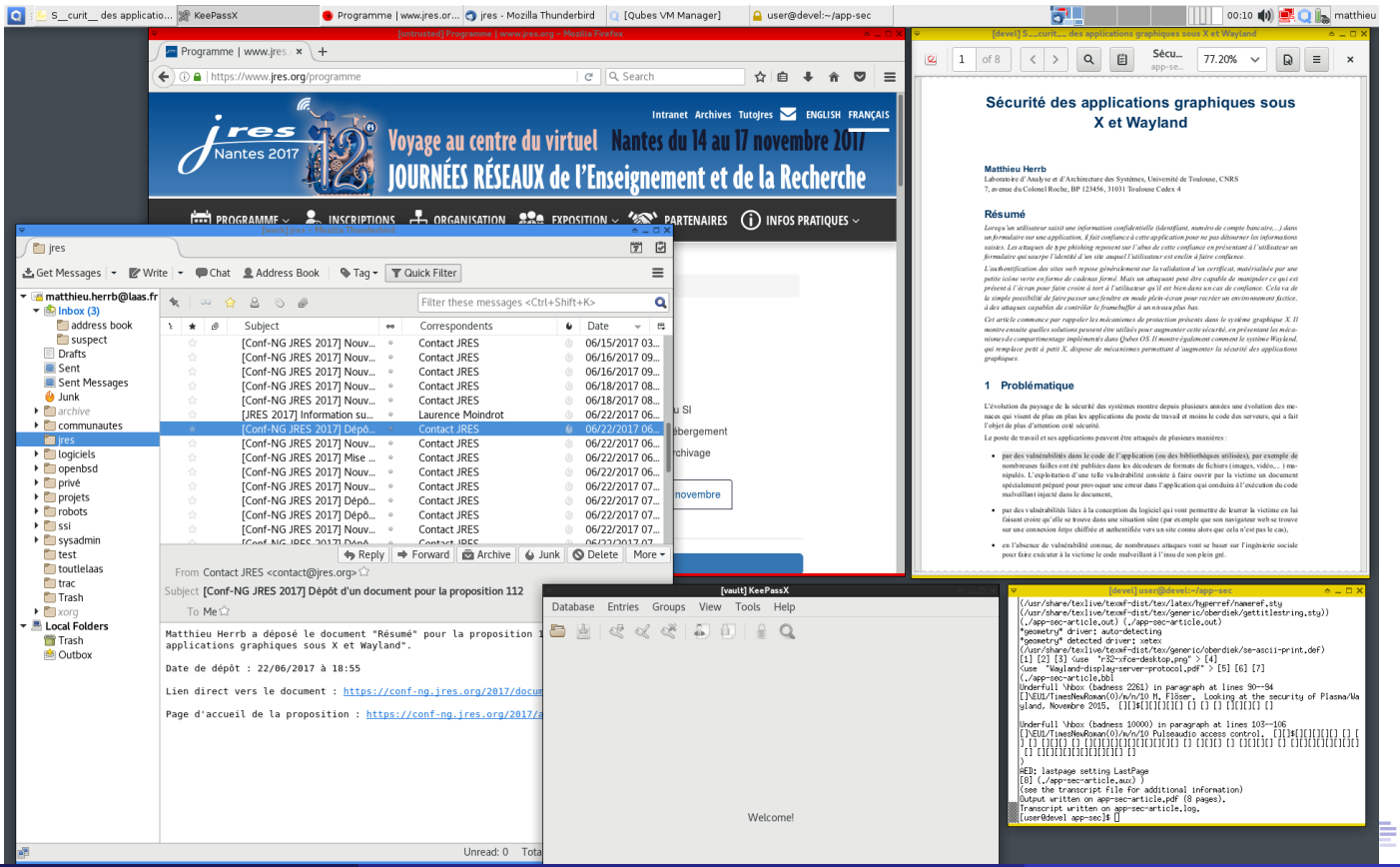
Wayland

- Pas de communication directe inter-clients
- Toutes les communications passent par le *Compositeur*



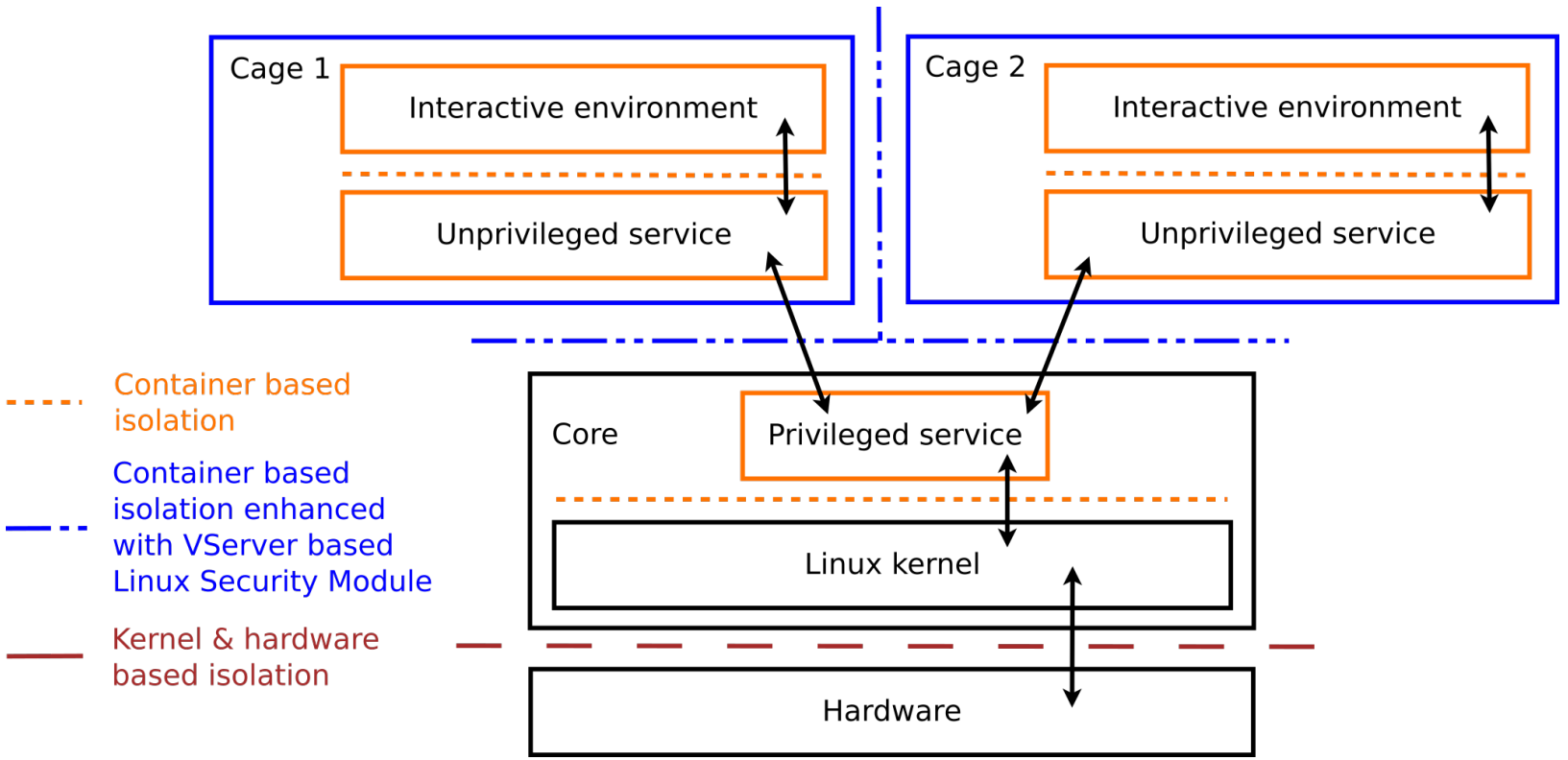
Qubes OS

- Isolation des applications dans des VM de niveaux de sécurité différents.
- Qubes Gui contrôle les échanges entre VMs
- Marquage explicite (code couleur) des applications par niveau de sécurité.



CLIP OS (ANSSI)

- CLIP OS système d'exploitation durci, capable de gérer des informations de plusieurs niveaux de sensibilité.
- se base sur le noyau Linux et un ensemble de logiciels libres.
- Contrôle de l'intégrité du boot et des systèmes de fichier (secure boot)



Flatpak & Snap

Modes de distribution des paquets binaires du futur (?)

- flatpak (redhat)
- snap (Canonical)

Agenda

Conclusions

- Nombreuses solutions possibles
- Appliquer le principe KISS (Keep It Simple and Stupid)
- Configuration par défaut de la distribution bien mise à jour
  $\rightarrow$ souvent suffisant
- Si besoins plus spécifiques $\rightarrow$ paquets/configurations sur mesure
  - penser aux mises à jour
  - valider / tester
  - surveiller le bon fonctionnement / les mises à jour
- Documenter les changements.

Bibliographie

- T. Ravier. Durcissement système à l'aide de systemd, Symposium sur la sécurité des technologies de l'information et des communications (SSTIC), Rennes, 2017.
- S. Dodier-Lazaro et M. Peres. Security in Wayland-based desktop environments : Privileged clients, authorization, authentication and sandboxing ! à *XDC2014*, Bordeaux, Septembre 2014. .
- A. Larsson. The flatpak security model, Janvier 2017. .
- M. Marczykowski-Górecki. Improving client systems security with Qubes OS. aux *RMLL2016*, Paris, Juillet 2016.
- T. Ravier et M. Salaün. CLIP OS : Building a defense-in-depth OS around Linux kernel security improvements à Kernel Recipes 2018, Paris.
- M. Herrb. Sécurité des applications graphiques sous X et Wayland. aux *JRES*, Nantes, 2017.