# StemJail : cloisonnement dynamique d'activités pour la protection des données utilisateur

Mickaël Salaün

ANSSI

5 juin 2015

# Objectif : protection des données utilisateur

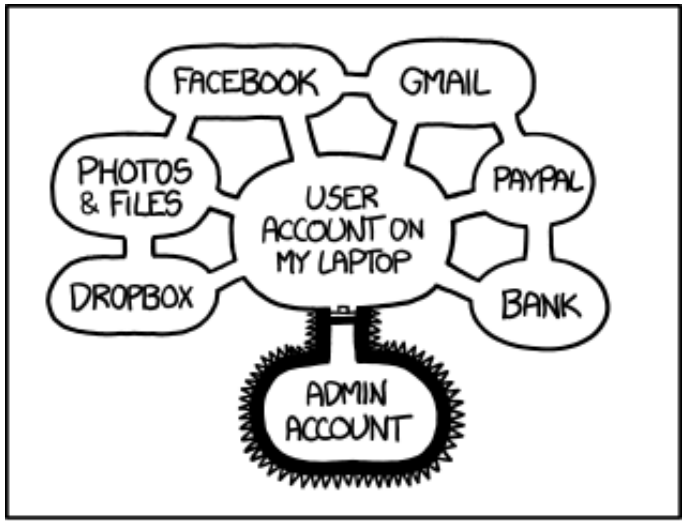
## Menace

Application compromise ou malveillante

## Domaines

Identification des cercles de diffusion :

- personnel/professionnel (emails)
- privé/public (photos)
- secrets (clefs SSH)
- clients X/Y/Z (documents)...




IF SOMEONE STEALS MY LAPTOP WHILE I'M LOGGED IN, THEY CAN READ MY EMAIL, TAKE MY MONEY, AND IMPERSONATE ME TO MY FRIENDS, BUT AT LEAST THEY CAN'T INSTALL DRIVERS WITHOUT MY PERMISSION.

`https://xkcd.com/1200/`

# StemJail : **cloisonnement** évolutif de processus

## Circonscrire une compromission

- complément au contrôle d'accès déjà présent sur le système
- dédié à l'utilisateur final (connait ses cercles de diffusion)
- cloisonnement d'instance d'application dans des cages dédiées
- accessible à tout utilisateur non privilégié

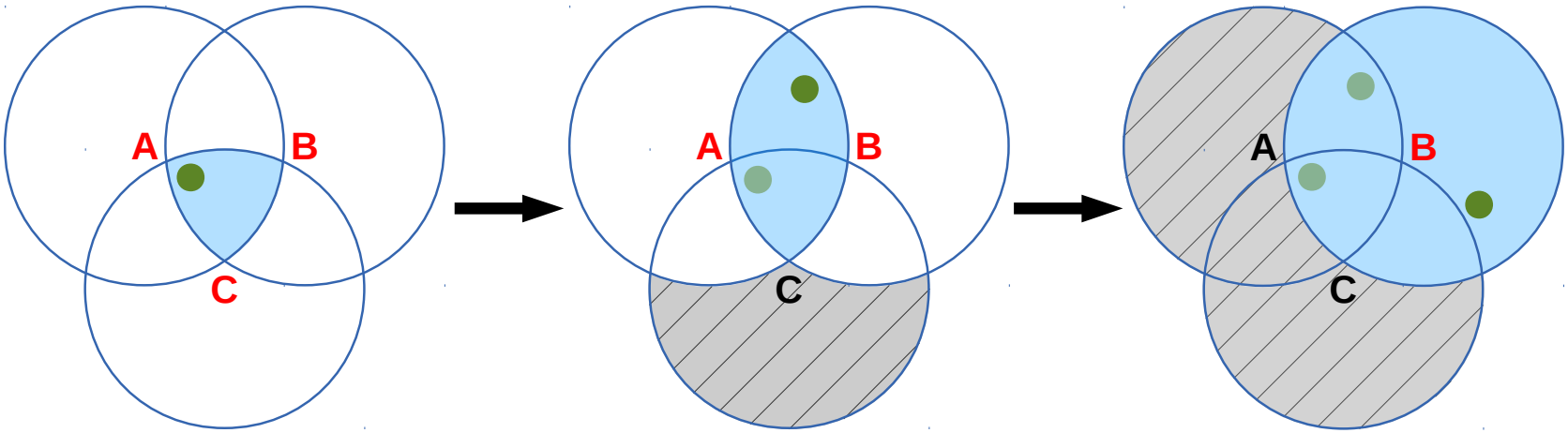
# StemJail : **cloisonnement** évolutif de processus

## Politique de sécurité

- création de domaines cohérents vis-à-vis des activités de l'utilisateur :
  - organisation initiale des fichiers par usage
  - respect du sens de cette organisation dans le temps
- simplicité d'utilisation :
  - définition d'accès en lecture-seule ou lecture-écriture par domaine
  - intégration dans le *workflow* de l'utilisateur
- découverte d'activité :
  - évolution des accès autorisés de la cage en fonction des accès demandés
  - accès accordé comme si on avait toujours été dans le domaine correspondant à l'activité

# StemJail : cloisonnement **évolutif** de processus

## Construction dynamique de domaine par activité




**1er domaine : A ∩ B ∩ C**

Transitions possibles :
- A
- B
- C
- **A ∩ B**
- A ∩ C
- C ∩ B

**2e domaine : A ∩ B**

Transitions possibles :
- A
- **B**

**3e domaine : B**

Aucune transition

: ressource
: domaine courant
: domaine accessible
: domaine inaccessible

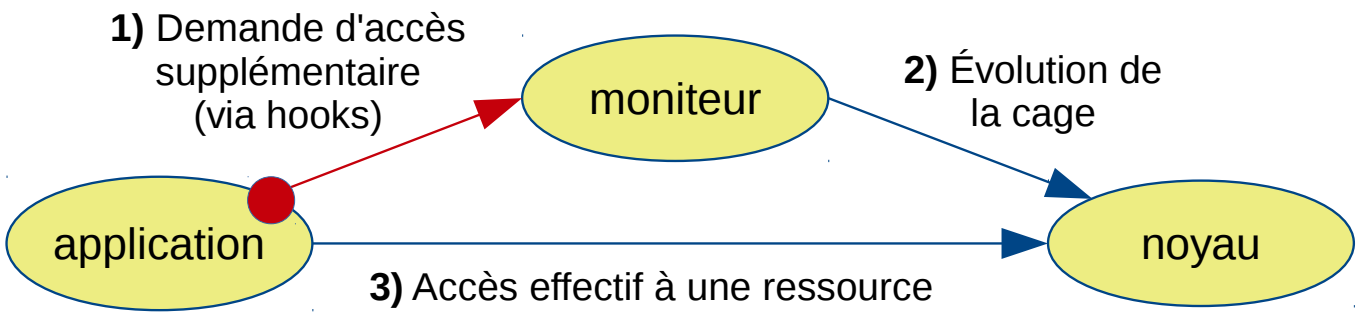
# StemJail : cloisonnement évolutif de **processus**

## Caractéristiques

- en espace utilisateur : cloisonnement via *user namespaces* Linux
- développé en Rust : typage fort, FFI, gestion mémoire, concurrence...

## Fonctionnement

- création de cage par le lanceur d'application (shell)
- architecture client/serveur (locale) :
  - application : interopérabilité via surcharge de la *libc* (dans les cages)
  - moniteur : validation des demandes d'accès et évolution de la cage



# **StemJail** : cloisonnement évolutif de processus

## Preuve de concept

- projet expérimental en cours de développement
- license LGPL v3
- disponible sur `https://github.com/stemjail`

# **StemJail** : cloisonnement évolutif de processus

## Preuve de concept

- projet expérimental en cours de développement
- license LGPL v3
- disponible sur `https://github.com/stemjail`

Démo