UberLogger

Un observatoire noyau pour la défense informatique

Plan

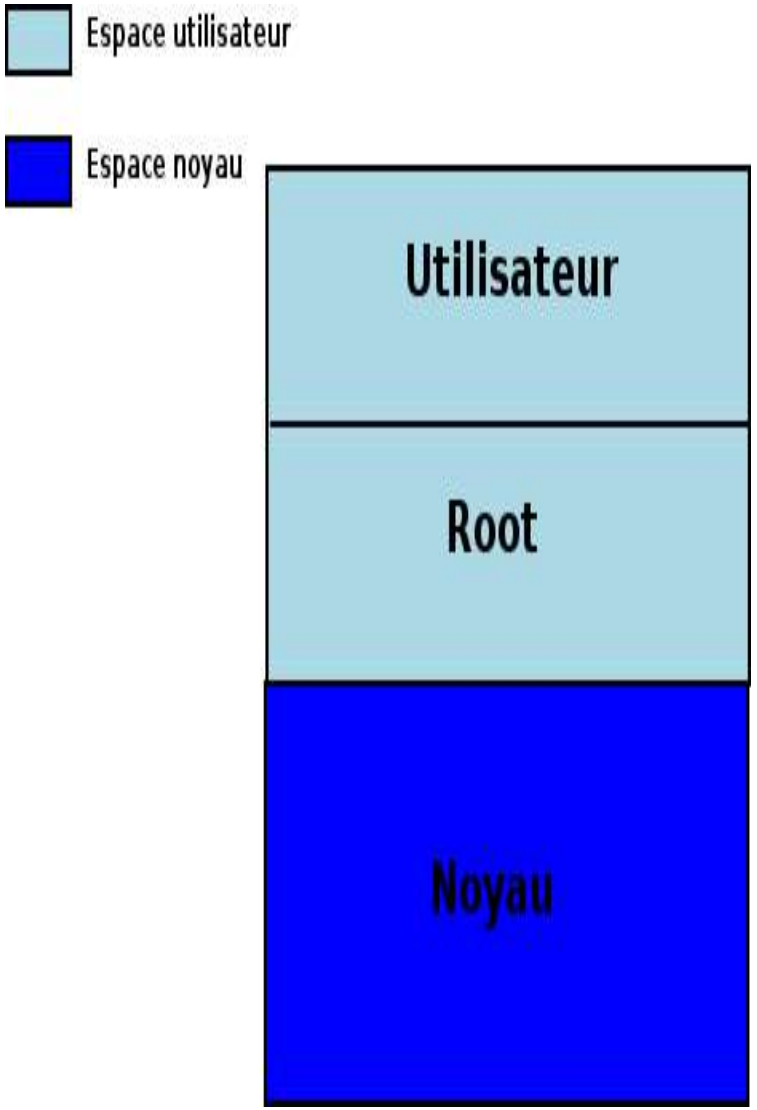
I. Capture des informations

II. Architectures et communications

III. Analyse des données

Où se placer?

- Utilisateur & Root: le pirate a les moyens d'acquérir ces privilèges (c'est ce qui nous intéresse ;-D)
- Noyau: privilège plus difficile à acquérir pour le pirate mais néanmoins possible (à éviter à tout prix)



Où se placer dans le noyau ?

- Appels systèmes: API permettant à l'espace utilisateur d'accèder aux fonctionnalités offertes par le noyau:
  - Plus haut niveau du noyau -> Intéraction directe avec l'espace utilisateur
  - Portabilité
  - Technique ayant fait ses preuves: utilisée par certains rootkit

Appels systèmes choisis

- open, read: quels fichiers et quelles informations dans ces fichiers sont lus
- mmap, readv, pread: permettent de se passer de read
- fork, execve: programmes executés + parenté
- chmod, chown: modification de privilèges
- init_module: module inseré par le pirate
- (setuid, ioctl, getdents, socketcall...)(voir paper)

Détournement des appels

- Méthode statique (modification du code des appels):
  - Ne marche pas dans le cadre du forensics
  - Modification détectable en désassemblant l'image du noyau.
- Méthode dynamique (module modifiant la table des appels systèmes):
  - Marche pour le forensics
  - Modification détectable:
    - présence du module
    - modifications de la table

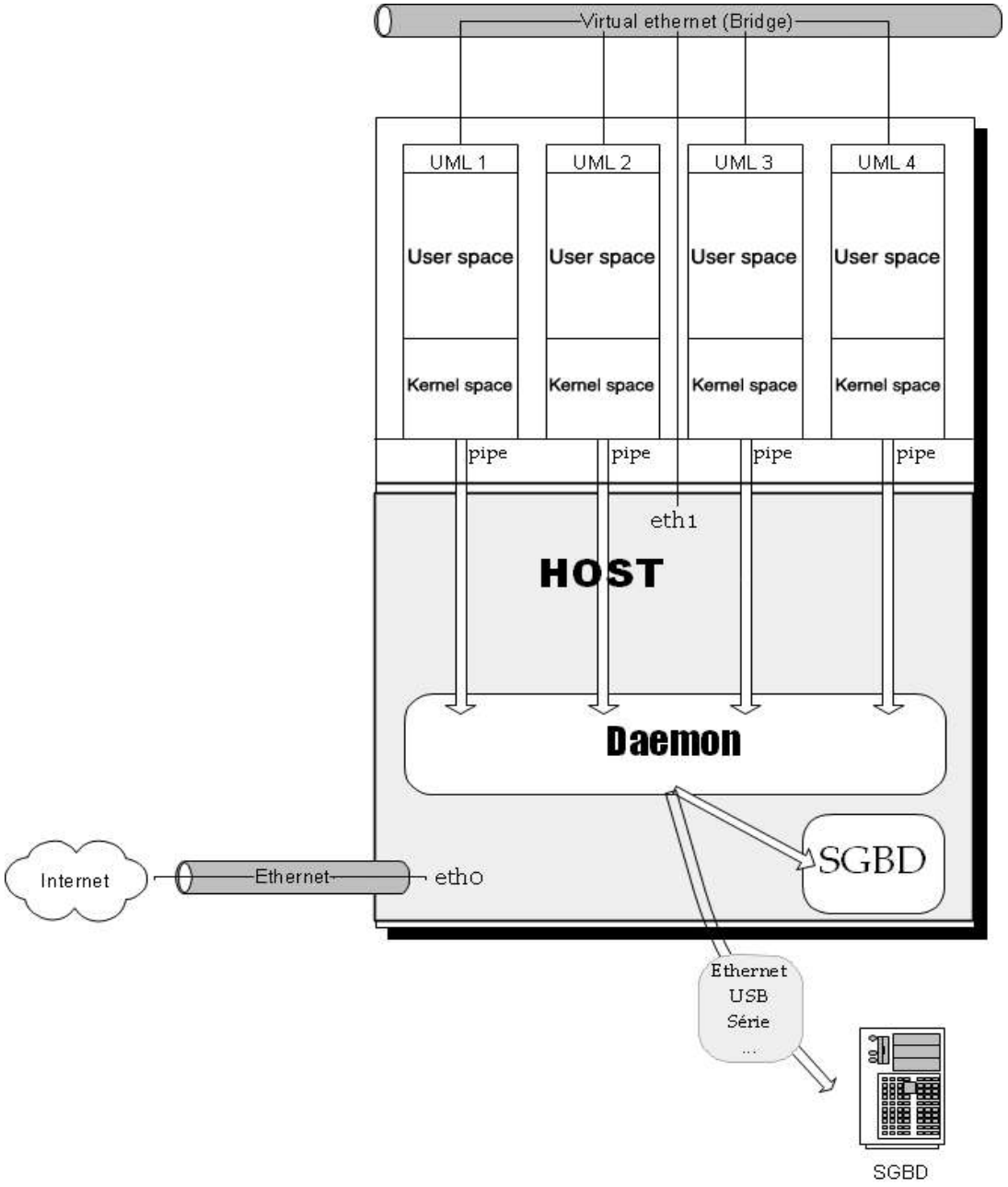
Masquage du dispositif

- Modification table appel système: modification de read, pread, readv, mmap sur lecture de:
  - /proc/kcore
  - /dev/kmem
- Présence du module:
  - Fusion de modules

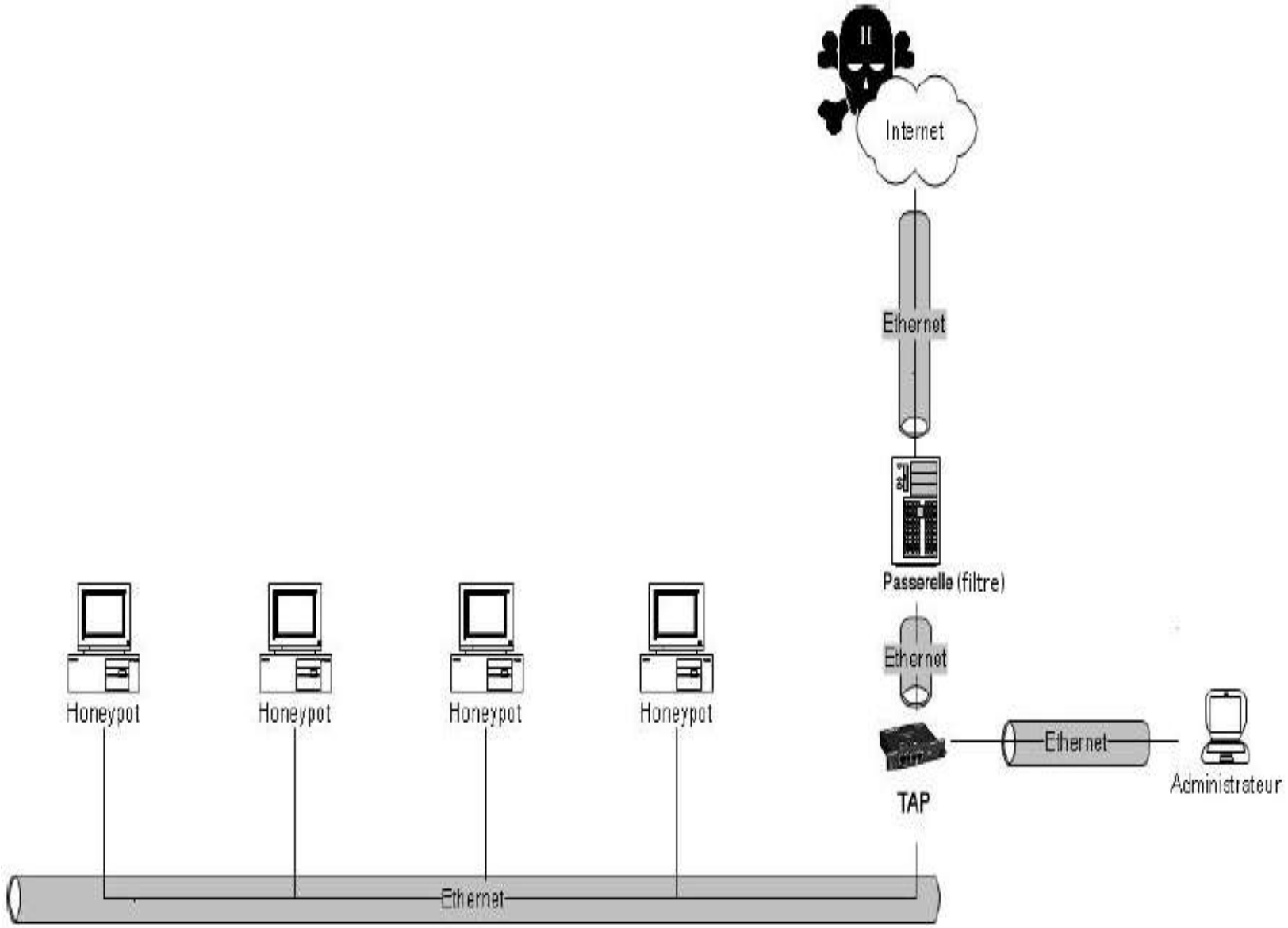
Protection contre la désactivation

- Utilisation des capabilities, on enlève:
  - CAP_SYS_MODULE
  - CAP_SYS_BOOT
  - CAP_SYS_RAWIO
- Faille permettant au pirate de récupérer ces capabilities: STHITH!! (Shoot the honeypot in the head)

Architecture UML



Architecture classique



Furtivité

- UML :
  - Coût faible
  - Communications furtives
  - Dispostif moins furtif
- Classique :
  - Dispositif furtif
  - Communications moins furtives

Analyse des données

- Objectif
  - Avoir accès facilement au maximum d'informations
  - Mettre à disposition des outils simples et efficaces
  - Analyse des appels système

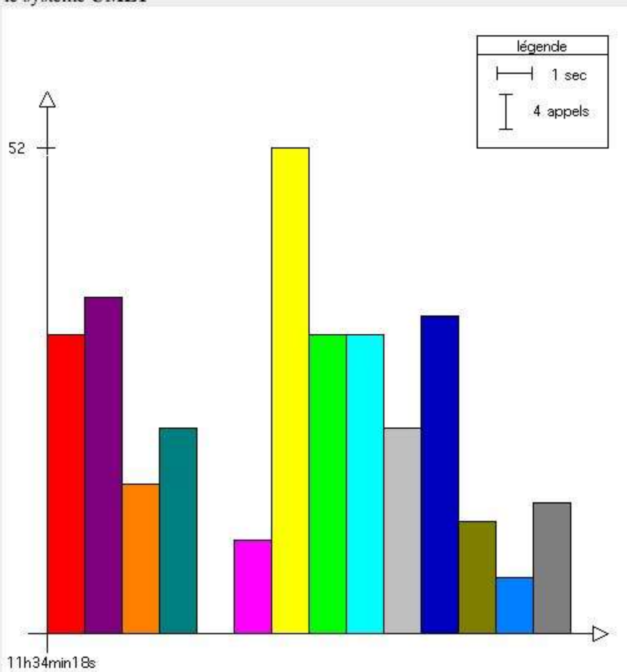
Analyse des données

- Récupération des informations
  - Communication furtive
  - Stockage dans une base de données
- Traitement des données
  - Interface d'administration (PHP/MySQL)
  - Outils d'analyse :
    - Tri des informations
    - Outil de recherche
    - Analyse temporelle

Screenshots

Analyse temporelle

Graphique temporelle de l'appel système **sysread** entre **11h32min18s** et **11h32min32s**
Pour le système **UML1**



UBERLOGGER

Résultat de la recherche

Voici tous les appels **execve** contenant **/bin/sh** dans les champs **argv**

uml : 0
time_seconds : 1110024699
time_microseconds : 526716
process_ID : 133
user_ID : 1000
enhanced_user_ID : 1000
cap_effective : 0
cap_inheritable : 0
cap_permitted : 0
res : 0
argc : 2
argv1 : vuln1
argv1 (Hexa) : 7675 6c6e 31
argv2 : ë$^□‰^□3Ò‰V□‰V□¸□V4□5□V4□N□‹ÑÍ€3À@Í€è×ÿÿÿ/bin/sh þÿ¿ þÿ¿
argv2 (Hexa) : 9090 eb24 5e8d 1e89 5e0b 33d2 8956 0789 560f b81b 5634 1235 1056 3412 8d4e 0b8b d1cd 8033 c040 cd80 e8d7 ffff ff2f 6269 6e2f 7368 20fe ffbf 20fe ffbf

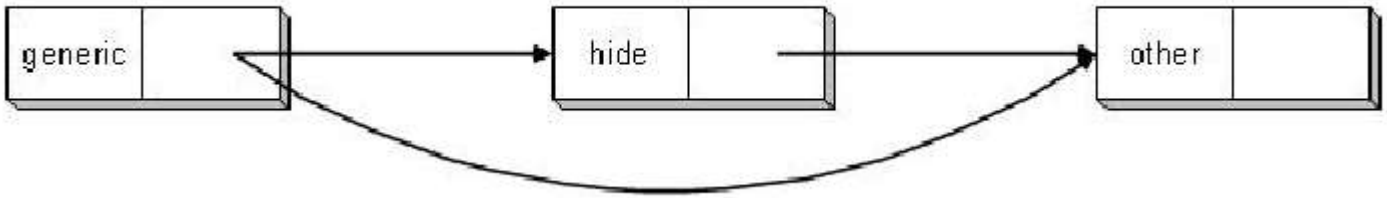
Conclusion

- Contacts :
  - Mailing list : uberlogger@rstack.org
  - Site : uberlogger.rstack.org
- Avenir du projet :
  - Furtivité sur /proc/kcore
  - Portage sur d'autres noyaux
  - Implémentation du log d'autres appels
  - Evolution de l'inteface

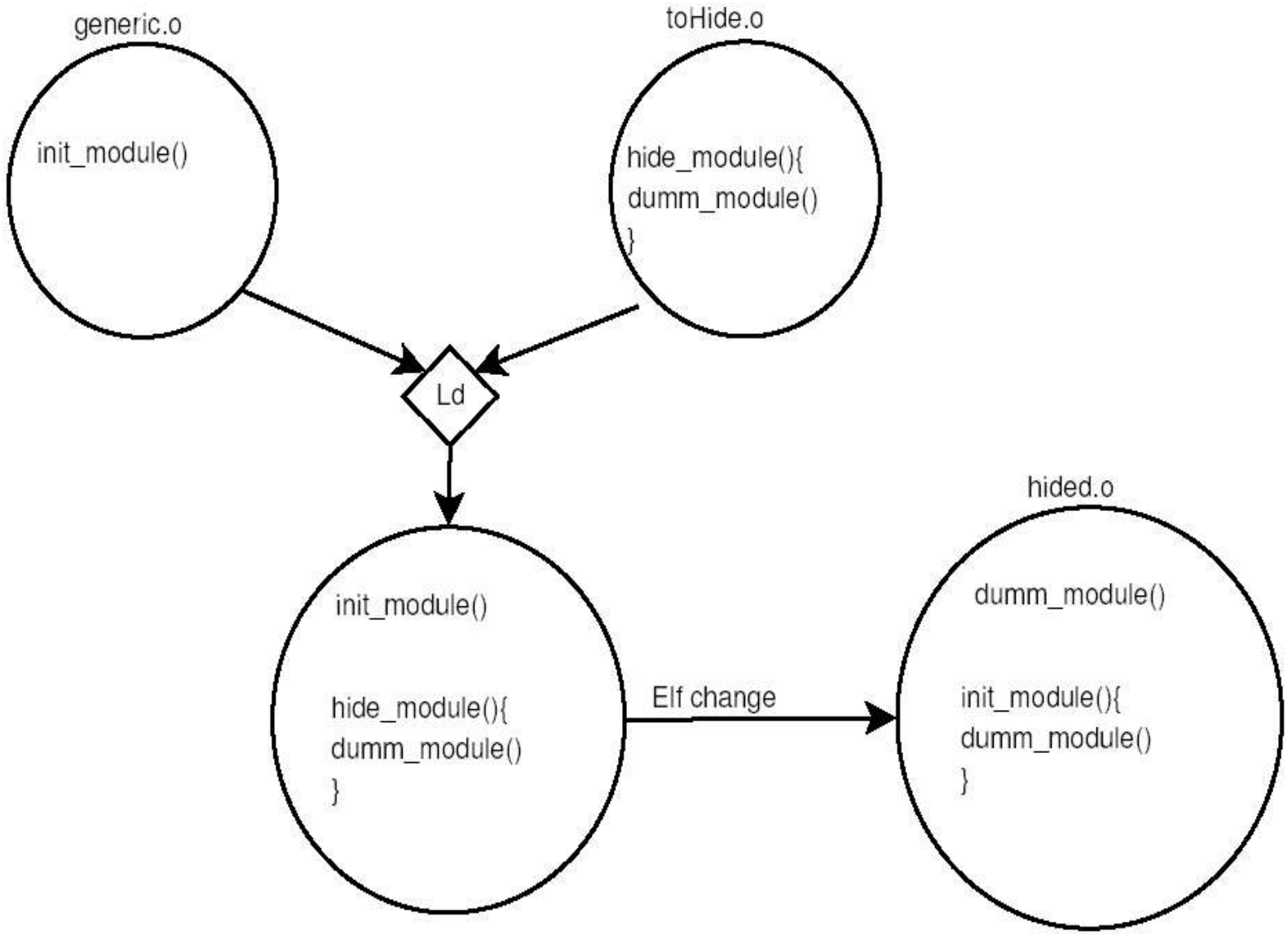
Des questions?

Merci beaucoup de votre attention

Saut de modules



Fusion de modules



Informations choisies

- Informations communes à tous les appels systèmes:
  - date d'execution de l'appel
  - pid, uid, euid du processus l'appelant
  - capabilities (effective, inheritable, permitted) du processus l'appelant
  - valeur de retour de l'appel