Systèmes d'exploitation

Virtualisation, Sécurité et Gestion des périphériques

Gérard Padiou

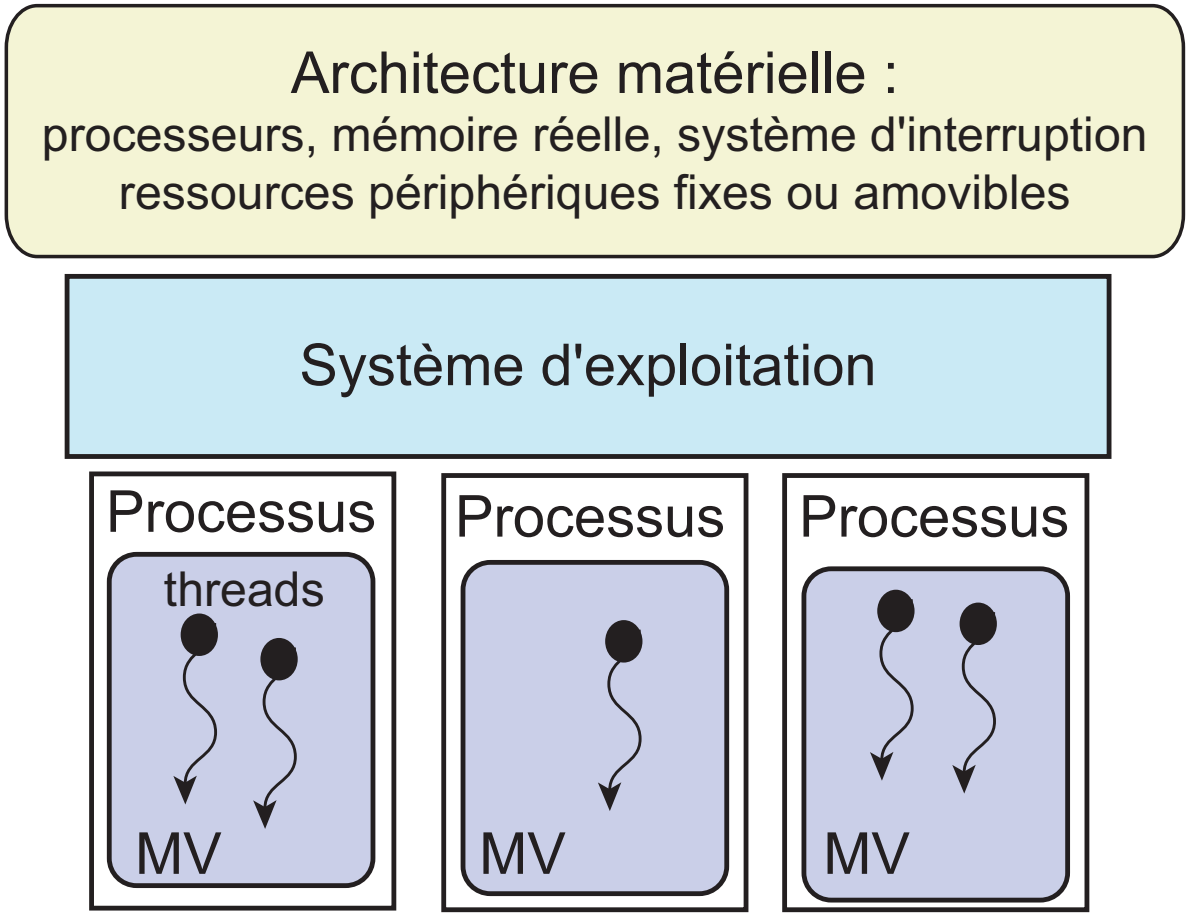
Département Informatique et Mathématiques appliquées
ENSEEIHT

Novembre 2009

plan

L'idée

La vision habituelle



L'idée

Plusieurs machines sur une seule !

Idée fondamentale d'objet virtuel

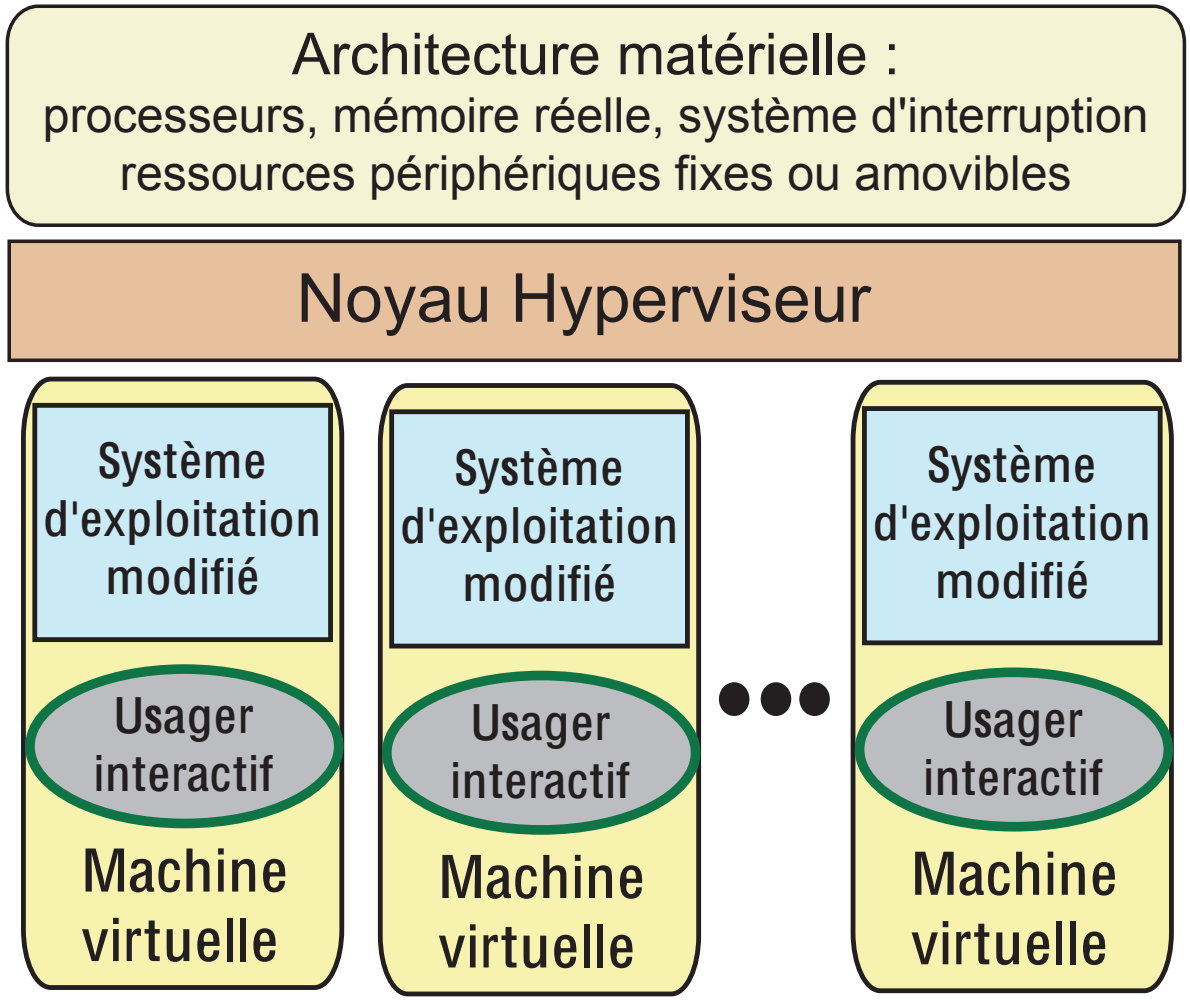
- Une vieille idée d'IBM (Cambridge-Grenoble) en 1967 ! :
  VM/CMS (Virtual Machine/Cambridge Monitor System)
- Remise au goût du jour ;
- Plusieurs implantations possibles ;
- Mecanismes matériels facilitant la mise en œuvre (mode d'exécution).

Objectifs

- Cohabitation de plusieurs systèmes différents ;
- Isolation : L'arrêt de l'un n'arrête pas les autres ;
- Facilite le test : cohabitation entre serveur en exploitation et test d'une nouvelle version.

L'idée

La vision des précurseurs



Types de solutions

Problème

Peut-on obtenir une solution performante sans modifier le système d'exploitation ?

Types de solutions

- La virtualisation par simulation complète avec plusieurs variantes :
  - Emulation de chaque instruction ;
  - Compilation de blocs d'instructions à la volée.
- La paravirtualisation ;
- La virtualisation à hyperviseur ;

Les problèmes à résoudre

Principe de base

Notion de machine virtuelle

Un système d'exploitation « invité » ne s'exécute plus sur la machine réelle mais sur une machine « virtuelle ».

☞ Un logiciel de virtualisation est donc nécessaire.

Virtualisation des ressources allouées à une machine

- Partage du temps processeur entre machines virtuelles ;
- Une unité de disque virtuel : réalisé par un « gros » fichier ;
- Une carte graphique de base : réalisée par une fenêtre ;

La virtualisation par simulation complète

Avantages et inconvénients

- Avantage : pas de modification du SE « invité »
- Inconvénient : performances faibles : il faut tout émuler.

Exemple

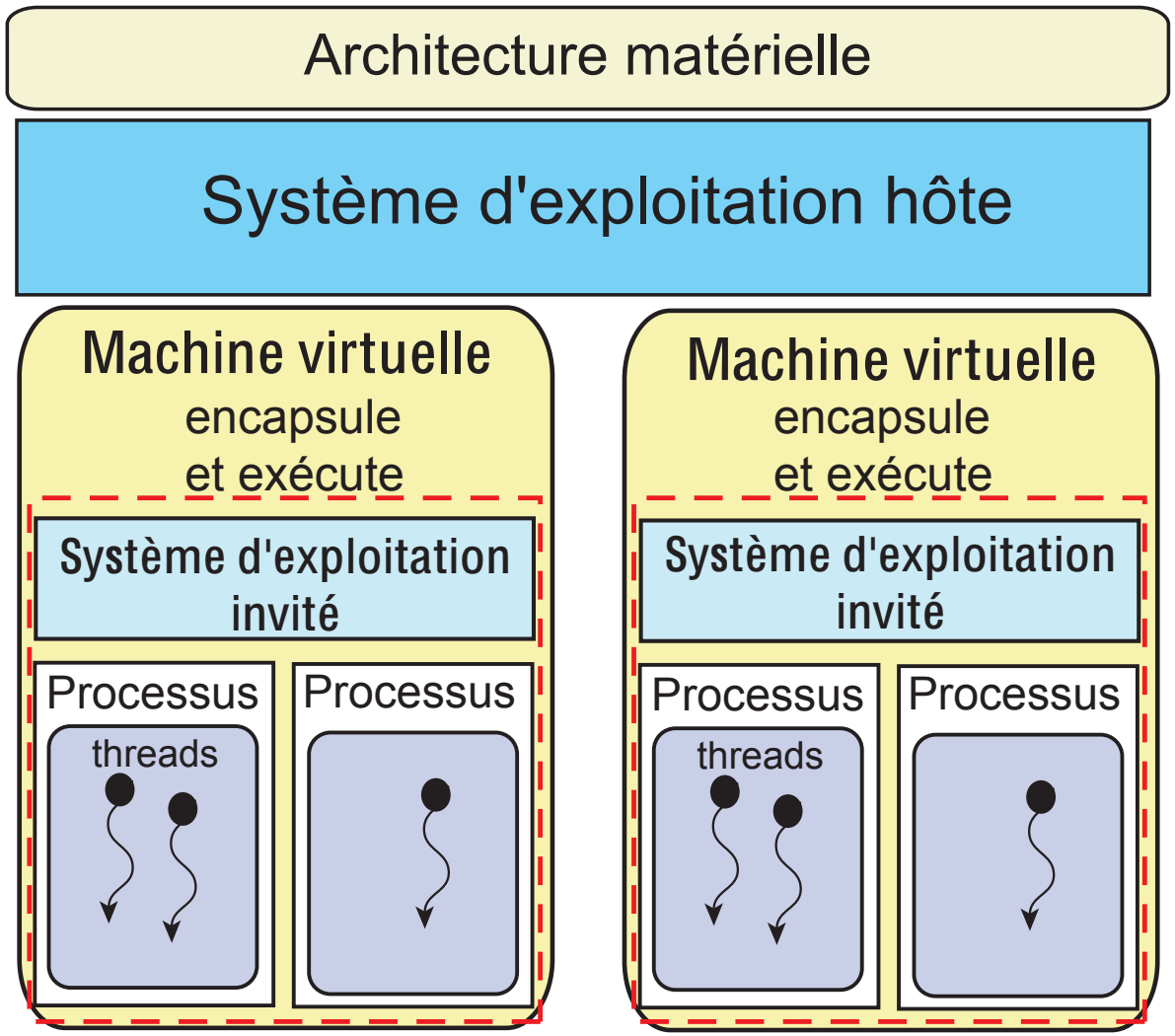
Projet Bochs

Site Web : http ://bochs.sourceforge.net/

- Emulateur de la famille de PC à base de IA-32(x86) ;
- Développé en C++ ;
- Emulation du processeur, d'un BIOS et des périphériques de base (console, disque, . . .) ;
- Supporte plusieurs SE : Linux, DOS, Windows NT/XT,. . .
- Problème : performances faibles mais bon support de test.

L'idée

La vision actuelle : par simulation (émulation)

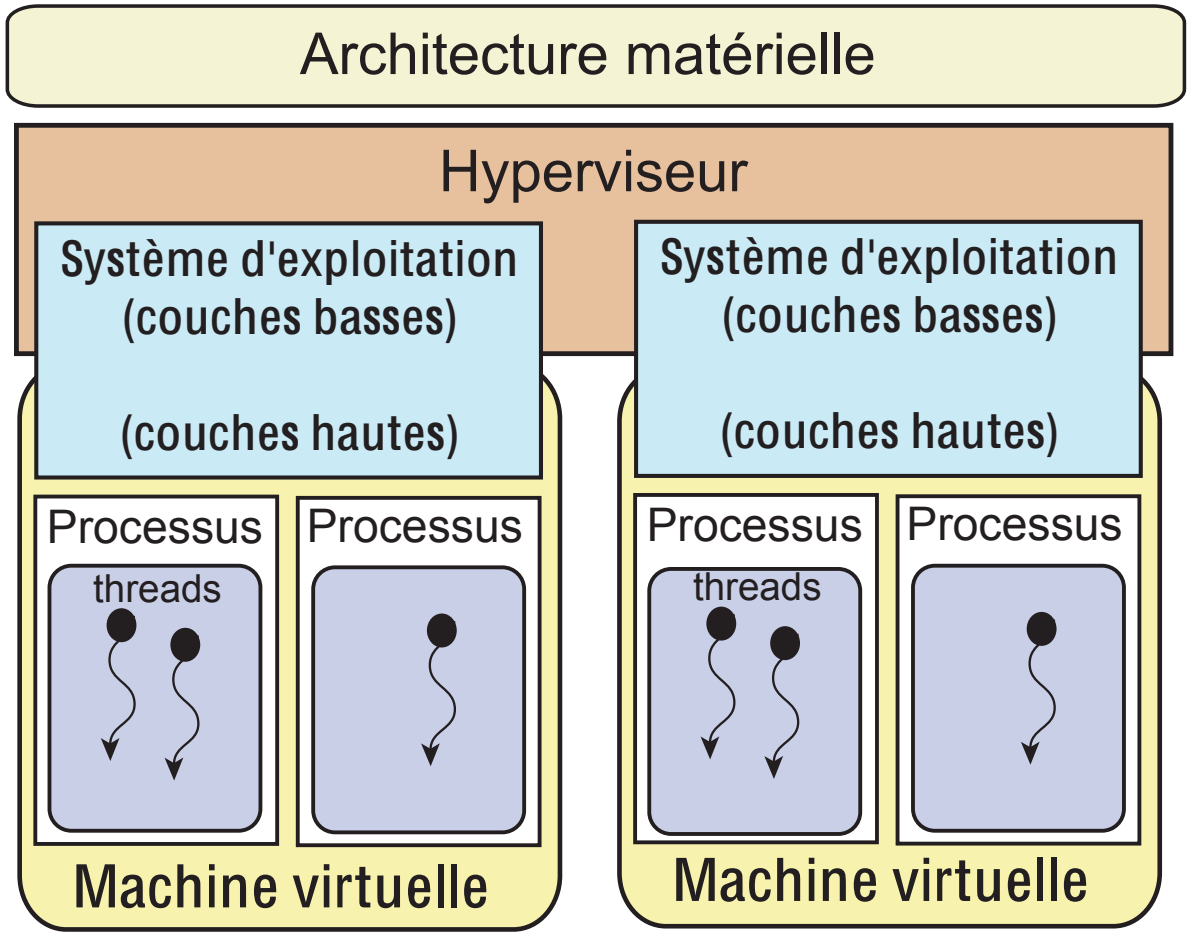


Exemple de configuration PC émulée par Parallels Desktop

- Un processeur Intel Pentium.
- Jusqu'à 1500 MB de mémoire.
- une carte graphique VGA.
- Jusqu'à 4 disques d'interface IDE (de 20 à 128 GB représenté par un fichie dont un de démarrage.
- Jusqu'à 5 interfaces réseaux , dont une carte virtuelle Ethernet.
- 4 ports séries (COM) ports.
- 3 ports parallèles (LPT)
- un controleur USB 2.0 controller.
- une carte son.
- un clavier PC générique.
- une souris PS/2.

L'idée

La vision actuelle : par hyperviseur



Des produits commerciaux

Exemple

Famille VMware

Site Web : http ://www.vmware.com/fr/

- Famille de logiciels de virtualisation.
- Versions serveurs ;
- Approche par hyperviseur ;
- Limite : processeur Intel (Pas d'émulation)

La sécurité d'un système

Problème fondamental

- Comment empêcher les accès frauduleux et/ou la destruction d'information ?
- Comment garantir qu'un système d'exploitation «résistera » à ses usagers (locaux ou distants)

☞ Beaucoup de possibilités à contrer . . .

La sécurité d'un système

Une règle qui éviterait bien des ennuis

☞ N'exécuter que les programmes que l'on a écrit . . .

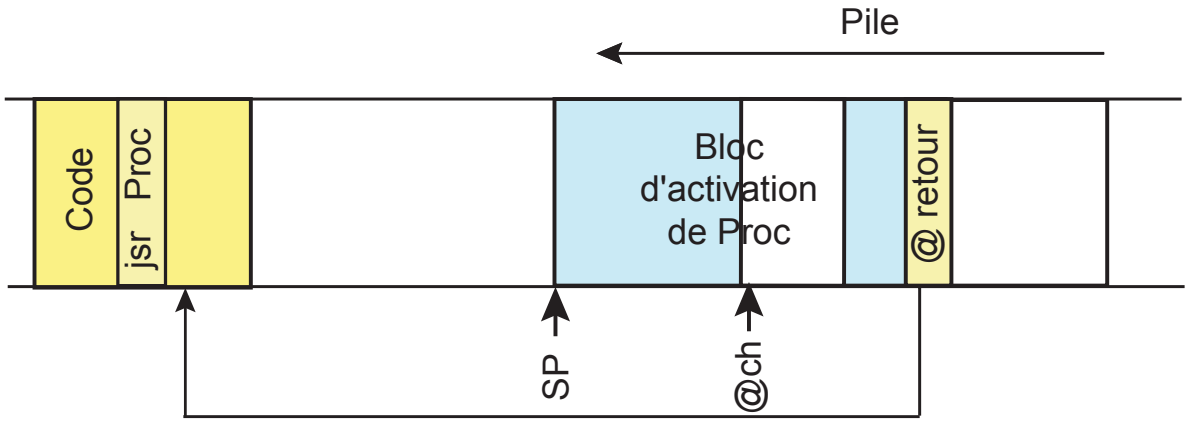
Le problème de base

- Un programme est toujours exécuté avec les droits de l'usager pour lequel « roule » le processus ;
- Les droits de l'usager qui a écrit le programme n'entrent donc pas en jeu ;
- Conclusion : celui qui a écrit le programme a pu prévoir qu'il aurait les droits d'accès d'un autre usager ;
- La grande quête du programmeur malveillant : arriver à faire exécuter ses programmes par un usager disposant du plus de droits possibles ⇒ par exemple, l'administrateur *root*

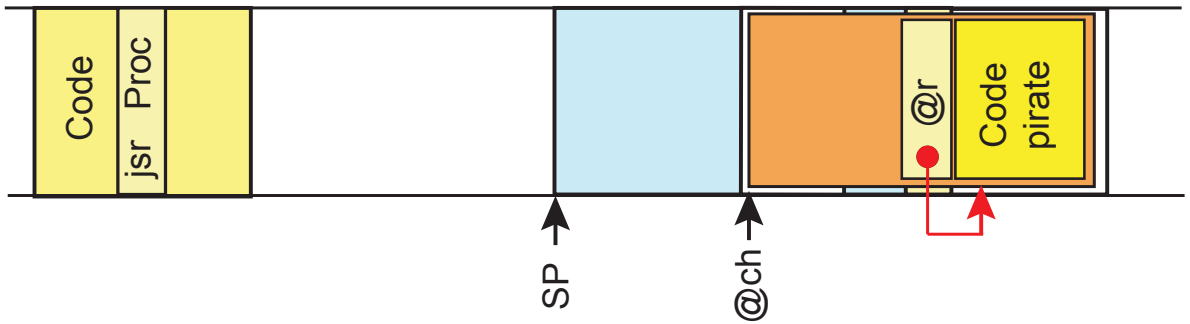
L'exploitation des points faibles

Les débordements de tableaux

- Avant la lecture de la chaîne rangées à l'adresse @ch :



- Après la lecture de la chaîne trop longue :

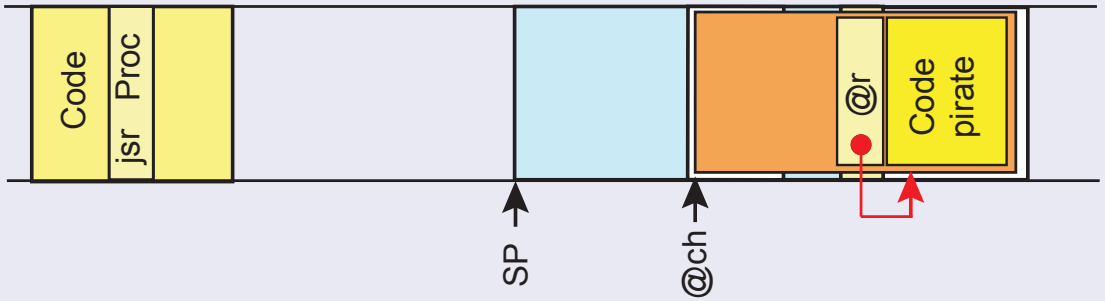


L'exploitation des points faibles

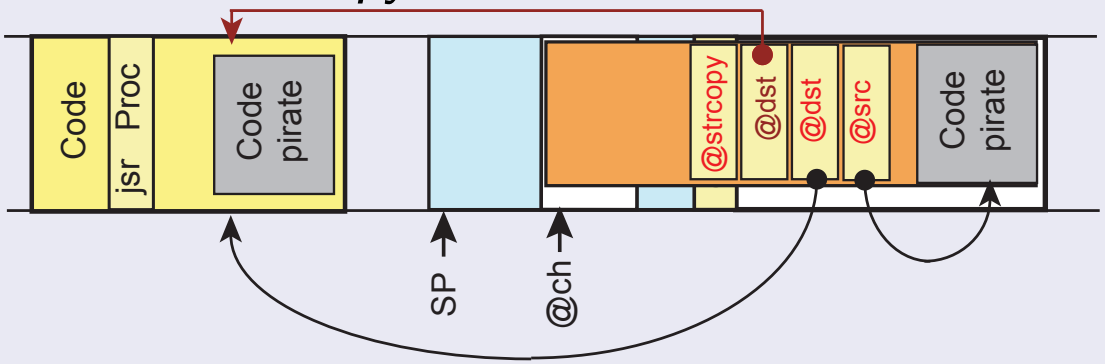
Les résultats de fonctions de la *libc*

Usage malveillant de la fonction *strcopy*

- Après la lecture de la chaîne trop longue :



- Après l'exécution de *strcopy* :



L'exploitation des points faibles

Que faire faire au code pirate ?

Il faut avoir choisi le bon programme à attaquer

Si le programme « au point faible » a son bit SETUID positionné et qu'il a été créé par *root* alors . . . :

- Il peut lancer par une primitive exec un shell avec les droits du super-utilisateur ;
- Il peut télécharger un programme via une connexion Internet et créer un processus qui exécutera ce programme.
- Ce processus peut alors accepter lui aussi des commandes, etc

L'exploitation des points faibles

L'injection de code

Idée

Introduire des commandes dans une donnée d'entrée

- Un programme accepte un nom de fichier en argument : argv[i] ;
- Il appelle la fonction *system* avec une commande en paramètre incluant la valeur argv[i] :

```
strcat(cmd,argv[i]) ;
system(cmd) ;
```

- dans argv[i] il y a : "toto ; rm $*$"

Les logiciels malveillants

Principe

- Utiliser la faille incontournable : un utilisateur exécute un programme qu'il n'a pas écrit.
- Le programme est un freeware qui fait quelque chose de rigolo ou joli . . .
- Mais il fait aussi autre chose. . . de moins visible.
- Deux approches :
  - Approche « cheval de Troie »
  - Approche « virus »

Les logiciels malveillants

Approche « cheval de Troie »

Principe

- Le programme malveillant installe un autre programme caché ;
- Ce programme caché est prévu pour être exécuté soit régulièrement, soit involontairement par un utilisateur . . .

Exemple

- Le programme est caché dans /usr/X11R6/bin et s'appelle lq ;
- L'utilisateur malheureux a dans sa variable PATH :
  /bin :/usr/bin :/usr/local/bin :/usr/X11R6/bin
- L'utilisateur malheureux frappe par erreur > lq (pour > ls)
- Le cheval de Troie est exécuté. . . et remplace /bin/ls par un autre cheval de Troie.

Les logiciels malveillants

Approche « virus »

Principe

- Au départ, principe de base identique au cheval de Troie ;
- Un programme contient un bout de code viral ;
- Lorsque le programme infecté est récupéré par un utilisateur, dès que celui-ci s'exécute, l'infection se propage.
- Un jour ou l'autre les programmes infectés déclenche . . . la maladie appelée charge utile (*payload*)

Les logiciels malveillants

Approche « virus »

Plusieurs types de virus

- Ceux qui se propagent dans les codes binaires par écrasement ou ajout (parasites, bouche-trou)
- Ceux qui se propagent dans du code source ;
- Ceux qui se cachent en mémoire réelle et filtrent les appels de primitives ;
- Ceux qui infectent les pilotes de périphériques ;
- etc

Unix : La gestion des péripériques

Les pilotes (drivers)

Les principes

- Définition d'une interface générique ;
- Indépendance vis-à-vis du matériel ;
- Reconfiguration dynamique ;

Les difficultés

- Grande diversité de ressources matérielles ;
- Périphériques amovibles ;
- Protection de la structure logique des systèmes de fichiers.

Unix : La gestion des périphériques

L'interface d'accès aux périphériques

Principes de conception

- Intégration dans l'espace de fichiers, dits *spéciaux* ;
  - La désignation est identique ;
  - L'interface d'usage au niveau commandes et primitives est celle des fichiers ;
  - Les mécanismes de protection sont les mêmes.
- Répertoire dédié : /dev
- Deux catégories de périphériques (méthode d'accès) : modes « blocs » ou « caractères »
- A un fichier spécial correspond un pilote ;
- Désignation interne d'un périphérique : couple de numéros (majeur, mineur)

Interface générique

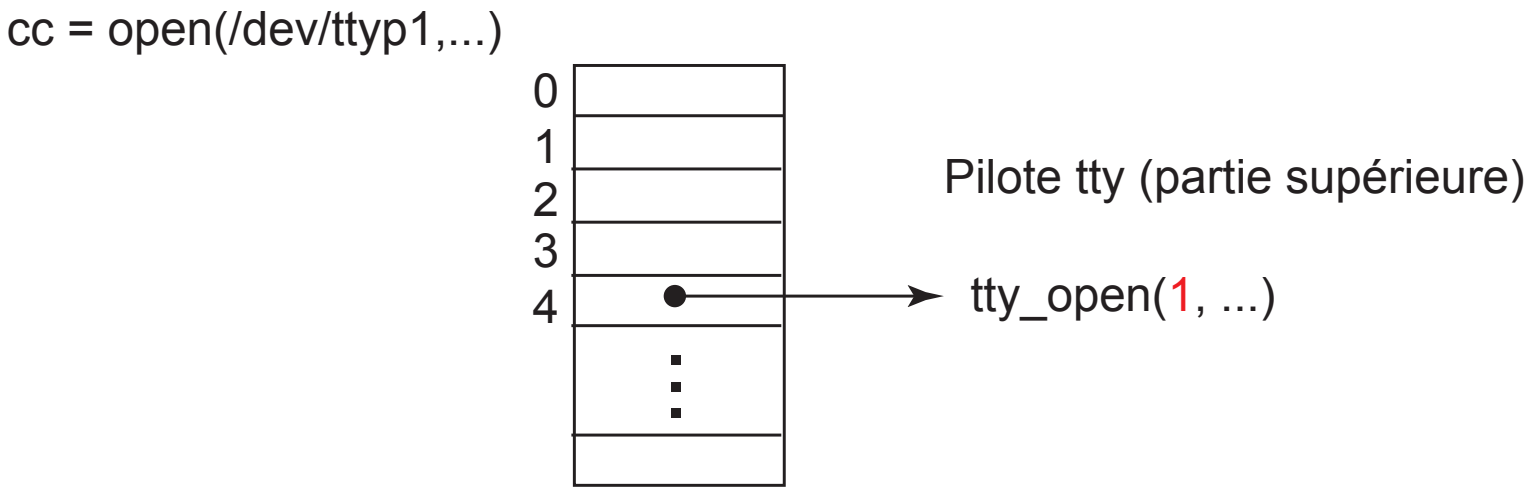
Les primitives d'accès

- open : connexion au périphérique ;
- close : déconnexion ;
- read : lecture (si possible) ;
- write : écriture (si possible) ;
- ioctl : action spécifique.

Remarque : la norme POSIX a introduit des primitives spécifiques pour remplacer la primitive « à tout faire » ioctl pour un terminal.
☞ cfgetispeed, cfsetispeed, cfgetospeed, cfsetospeed, cfsetspeed, cfmakeraw, tcgetattr, tcsetattr

Réalisation
Les pilotes

```
> ls -l /dev/ttyp1
crw-w--- 1 PE tty 4, 1 .../dev/ttyp1
```



Modes d'accès par bloc ou « caractère »