Backdoors et rootkits avancés

© 2002 Nicolas Dubée, ndubee@secway.com

Plan de la présentation

- Introduction
- Aperçu du principe des backdoors kernel
- Démonstration sous Solaris
- Prévention et détection

© 2002 Nicolas Dubée, ndubee@secway.com

Introduction

© 2002 Nicolas Dubée, ndubee@secway.com

Définitions

- Backdoor
  - Porte dérobée plantée par un intrus et lui permettant de revenir ou d'élever ses privilèges plus facilement
- Rootkit
  - Modification par un intrus de composants légitimes du système, par exemple dans un but de dissimulation, de backdoor.

© 2002 Nicolas Dubée, ndubee@secway.com

Pourquoi cet exposé ?

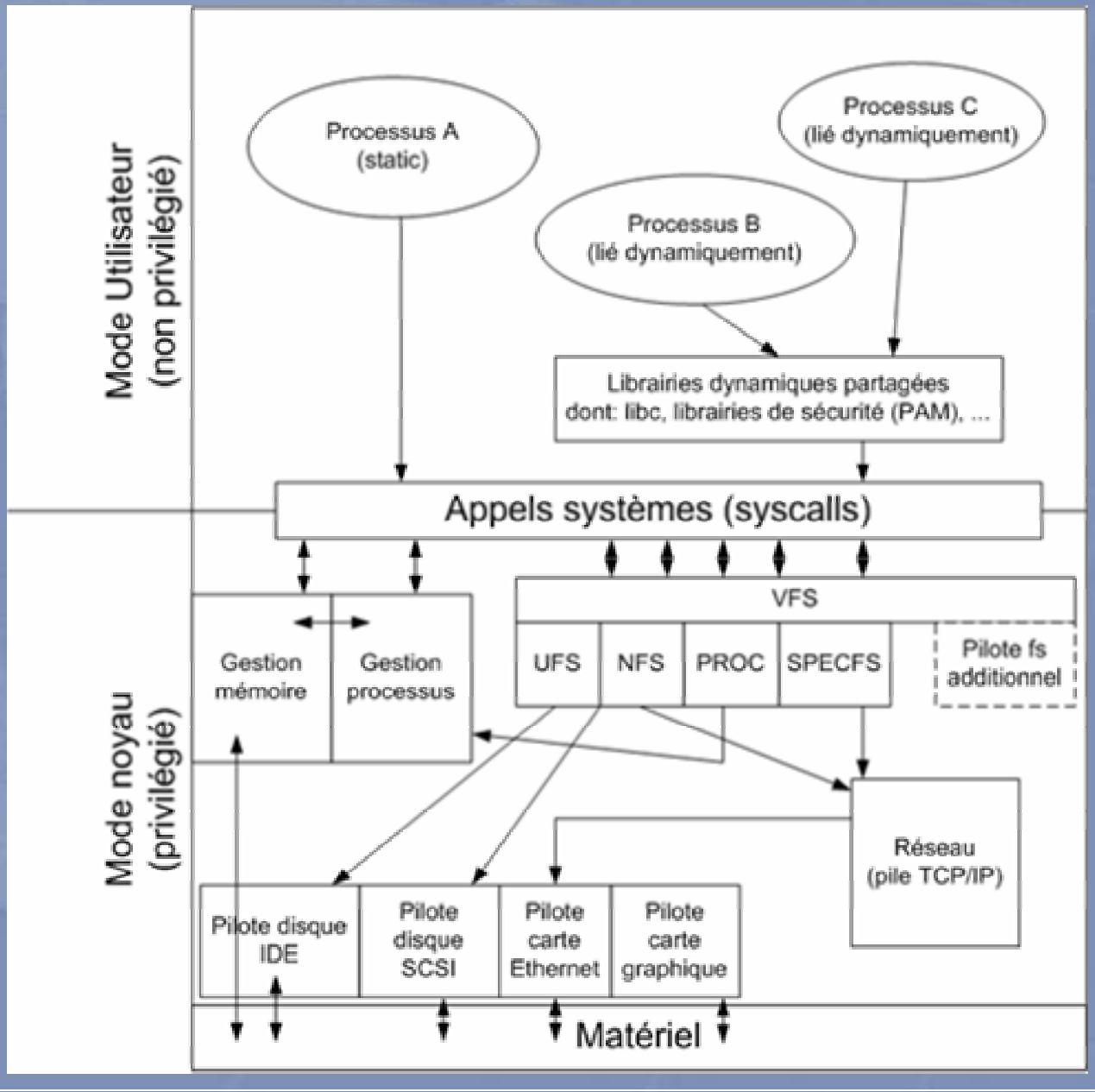
- Exemple d'une application d'attaque « à la mode »
- Démontre les conséquences en cas de compromission de la base de confiance
- Démontre le manque de fonctionnalités avancées de sécurité des OS

© 2002 Nicolas Dubée, ndubee@secway.com

Rappels sur les OS

- OS = système d'exploitation = interface matériels / logiciels
- L'OS propose un ensemble de services aux applicatifs
- Partie de l'OS fonctionnant en mode privilégié : kernel
- Tous les logiciels utilisateurs passent par les fonctionnalités de l'OS (syscalls)

© 2002 Nicolas Dubée, ndubee@secway.com



© 2002 Nicolas Dubée, ndubee@secway.com

Place de la sécurité dans un OS

- Des fonctionnalités en mode kernel
- Mais aussi souvent en mode utilisateur
  - Fonctionnalités (protection mémoire, FS) de base en mode kernel
  - Le reste (gestion passwords, ...) en mode utilisateur
- Tous les systèmes de sécurité se basent sur des fonctionnalités offertes par l'OS
  - pour récupérer l'information de décision
  - Pour appliquer les actions en conséquence

© 2002 Nicolas Dubée, ndubee@secway.com

Modularité

- La modularité est une contrainte
- Les kernels des OS sont hautement configurables
- Les kernels ne sont pas fermés
- Du code peut y être ajouté (pilotes périphériques)

© 2002 Nicolas Dubée, ndubee@secway.com

Le paradigme « root »

- Sous Unix, les seules permissions reconnues au niveau kernel
  - L'UID 0 (root) dispose de toutes les permissions
  - Les autres (!= 0) sont restreintes à leurs propres objets
- Le root dispose entre autres de la possibilité de modifier l'OS

© 2002 Nicolas Dubée, ndubee@secway.com

Rootkits kernel

© 2002 Nicolas Dubée, ndubee@secway.com

Idée de base

- Sur une machine compromise
- Modifier le kernel
- Pour faire réagir l'OS comme souhaité
- Et ainsi biaiser tous les applicatifs
- Dont ceux de sécurité

© 2002 Nicolas Dubée, ndubee@secway.com

Résultats

- La base de confiance est compromise
- Tous les applicatifs de sécurité (H/N-IDS, journalisation) sont inutiles
- Car ne peuvent avoir comme vue du système que celle fournie par le kernel

© 2002 Nicolas Dubée, ndubee@secway.com

Démonstration sous Solaris

© 2002 Nicolas Dubée, ndubee@secway.com

Modification du kernel

- Une fois que l'intrus a accès au compte « root »
- Il utilise des fonctionnalités standard de l'OS pour modifier le kernel
  - Modules dynamiques (LKM)
  - Modification des fichiers kernel (/kernel/genunix)
  - Accès mémoire direct (/dev/kmem, /dev/mem)
- Et y injecte son propre code

© 2002 Nicolas Dubée, ndubee@secway.com

Réalité du problème

- **Adore** v0.34 (Linux), Stealth
  - Avec programme d'installation et d'administration aisée permettant de gagner root par une fonction cachée, dissimuler des processus, des fichiers des connexions TCP (contre netstat) et du mode Promisc.
- **Kernmod** 0.2 (Solaris), Job de Haas - ITSX
  - Démonstration d'un module kernel Solaris réalisant toutes les opérations de base (dissimulation de fichiers, processus, connexions TCP, …).
- **Rootkit** 2000 (Windows NT 4.0, Windows 2000), Greg Hoglund & collectif
  - Rootkit kernel sous Microsoft Windows.

© 2002 Nicolas Dubée, ndubee@secway.com

Techniques usuelles

- Cacher les fichiers
  - Modification des routines de gestion FS (VFS) ou même devices
- Cacher des processus
  - Modification des listes chaînées d'ordonnancement
- Cacher des connexions réseaux
  - Modification des listes chaînées des connexions actives
- Ajouter des fonctionnalités cachées
  - « Covert channels »
  - Backdoors avec déclenchement par réseau

© 2002 Nicolas Dubée, ndubee@secway.com

Détection et prévention

© 2002 Nicolas Dubée, ndubee@secway.com

Le problème

- Il n'y a pas vraiment de moyen de vérifier l'OS « à chaud »
- Car c'est lui qui définit la vue du système qu'ont les applications
- Le rootkit peut donc leur présenter un état normal

© 2002 Nicolas Dubée, ndubee@secway.com

Les méthodes de détection

- La plus fiable : redémarrer sur un kernel sûr et vérifier manuellement
- Utiliser les signatures des rootkits traditionnels
- Utiliser des bugs dans les rootkits pour les détecter
- Si bien conçus, aucun moyen de les détecter !!!

© 2002 Nicolas Dubée, ndubee@secway.com

La prévention (1)

- Idéalement, empêcher l'intrus de devenir root
- Cependant, c'est difficile !!!

© 2002 Nicolas Dubée, ndubee@secway.com

La prévention (2)

Dans le monde réel :

- Vérifier le code à injecter dans le kernel par signature (du constructeur)
- Interdire toute modification du kernel
  - Pas toujours possible
  - Pas toujours pratique
  - Securelevel BSD
- Limiter les privilèges de l'utilisateur « root »
  - Compartimentation des droits
  - Patchs de sécurité (Argus, …)

© 2002 Nicolas Dubée, ndubee@secway.com

Conclusions

- Les pirates déploient des techniques évoluées
- Compromettant la base de confiance qu'est l'OS
- Celui-ci ne dispose pas nativement de fonctionnalités de sécurité suffisantes
- Modularité au détriment de sécurité

© 2002 Nicolas Dubée, ndubee@secway.com

Merci et bonne journée !

Cette présentation et d'autres disponibles sur
http://www.secway.com/

© 2002 Nicolas Dubée, ndubee@secway.com