Sécurité du noyau Linux et systèmes de contrôle d'accès

SSTIC – 02/06/05

Julien Tinnès

Le contrôle total

- **Un attaquant souhaite souvent avoir le contrôle total d'un système**
- **Contrôle total d'un système = exécution de code arbitraire en mode noyau (CPL0 sur Intel)**
- **Souvent équivalent à être root (avoir l'uid 0)**
  - Mais capabilities depuis Linux 2.2
  - Systèmes de contrôle d'accès
    - RSBAC
    - LIDS
    - GRSecurity
    - SELinux

Sécurité du noyau Linux

- **En 2004, Brad Spengler trouve 10 failles avec grep en quelques heures**
- **Paul Starzetz publie une faille tous les deux mois**
  - Failles nécessitant très peu de privilèges pour être exploitée (mremap…)
- **Maintainers très peu réactifs**
  - Failles critiques ne déclenchant pas la sortie d'un nouveau noyau
  - Très peu sensibilisés à la sécurité kernel
- **Un peu mieux depuis 2005**
  - 11 révisions du noyau 2.6.11! (Greg KH)
  - 2.4 (Willy Tarreau –hf)
- **Expérience: une vingtaine d'erreurs de programmation trouvées en quelques jours**

Failles kernel

Prévention générique délicate

- Rien au dessus du noyau
- Méthodes classiques inefficaces
  - Pile kernel non exécutable, randomizée ou protégée par SSP peu utile
  - Le moindre bug peut avoir des conséquences de sécurité et ne sont souvent pas des overflows
- Projet de kernel armor, mais peut satisfaisant

Conséquences importantes

- Permet de passer outre un système de contrôle d'accès
- La plupart des failles permettent de passer en mode noyau (même les lost VMAs)
- Déréférence de pointeurs nuls plus facile à exploiter qu'en mode user

Failles kernel (2)

- **Beaucoup de bugs exploitables sous conditions particulières**
  - Avec l'uid 0 (mais sans capabilities particulières)
  - Avec certaines capabilities
  - Permettent de prendre le contrôle du noyau
    - Et en particulier de désactiver tout système d'AC
  - Vraiment facile d'en trouver
- **Vraiment difficile de les faire corriger**
  - Beaucoup de maintainers ne comprennent pas les conséquences
    - « Pourquoi rajouter un BUG_ON(), l'utilisation de la première page produira un stack trace similaire si le pointeur est déréférencé» (null pointer dereference)

Conclusion

- **Etapes**
  - 1. Exécuter du code arbitraire en mode user
  - 2. Exécuter du code arbitraire en mode kernel
- **Sur une machine sécurisée il faut prévenir (1)**
  - Prévention générique classique: Full PaX, SSP…
  - Si utilisateurs locaux, utiliser un système de contrôle d'accès pour faire du TPE
    - Il faut garantir qu'un programme ne permet pas d'exécuter du code arbitraire
      - *CF ld.so, langages de scripts…*
      - *Tout programme devient une cible potentielle! Ex: ls*
- **On ne peut a l'heure actuelle espérer prévenir correctement (2)**
- **Il faut auditer le noyau Linux!**