Noyaux durcis

pour tout le monde

Yves-Alexis Perez

SSTIC 2016

Qui suis-je ?

Yves-Alexis Perez

ANSSI chef du labo « architectures matérielles et logicielles »

- sécurité des plate-formes
  x86 principalement, ARM
- OS et couches inférieures
  CPU/chipset/SOC, PCI Express, périphériques
- sécurité physique

Debian Dévelopeur

- mainteneur
  - environnement Xfce
  - démon IKE strongSwan
  - **linux-grsec**
- membre de l'équipe sécurité

Contexte

Cette présentation n'est **pas**

- une description détaillée de grsecurity

Cette présentation parle du noyau Linux

- sécurité
- durcissement

Elle vise à faciliter

- l'installation d'un noyau durci
- son utilisation au quotidien

Elle est destinée en particulier à des sysadmins et utilisateurs finaux

Quel besoin de sécurité dans le noyau ?

- *kernel knows everything* (Michaël Kerrisk)
- le noyau tourne en ring0 (mode superviseur)
- le noyau gère la sécurité pour l'espace utilisateur :

DAC	*Discretionary access control*
MAC	*Mandatory access control*
Namespace	isolation
IPsec	chiffrement réseau
dm-crypt	chiffrement de disque
...	

Sécurité noyau souvent limité à la correction de bugs

Habituellement :

vulnérabilité ⟹ élévation de privilège (locale)

- *quasi-systématique* (par exemple *pour rooter Android*)
- mais pas toujours le cas

Correction des vulnerabilités

- de préférence *avant* qu'elles soient déployées dans la nature
- mais parfois après

Ce n'est clairement pas suffisant !

mais nécessaire, bien évidemment

Durcissement noyau

Protection *active* contre les attaques

- nouveau terme marketing : *kernel self protection*
- protéger le noyau des attaques externes (espace utilisateur)
- réduire la surface d'attaque

kernel *airbag*

Fonctionnalité de sécurité :

Linux possède déjà des fonctionnalités de « sécurité »

- DAC
- LSM
- Namespaces

Principalement utiles à l'espace utilisateur

- isolation des processus
- isolation des utilisateurs
- isolation des ressources
- politique de sécurité

grsecurity

En bref

- un (gros) patch pour le noyau Linux
- orienté sécurité (évidemment)
- plus de quinze ans d'existence
- innovations régulières, encore maintenant
- plusieurs grands composants :

PaX	protection contre les corruptions mémoires parfois désigné comme un HIPS[1]
RBAC	Role-based access control
Divers	Durcissement variés protections mémoire, systèmes de fichier, réseau…

1. Host Intrusion Prevention System

PaX

Composants majeurs

MPROTECT respect du W⊕X
pour les allocations en espace utilisateur

KERNEXEC équivalent noyau de NX+MPROTECT ;
empêche tout injection/exécution de code extérieur au noyau

ASLR avant son apparition dans le noyau Linux
(et partout ailleurs)

UDEREF empêche le déréférencement de pointeurs *userland*
SMEP/SMAP ou PXN/PAN, en avance

PaX

plugins GCC

CONSTIFY rend *const* les structures ne contenant que des pointeurs de fonction

SIZE_OVERFLOW détection/prévention des débordement d'entiers

REFCOUNT détection/prévention des débordement de compteurs de références
empêche en particulier l'exploitation de *use-after-free*

RAP[1] *Reuse Attack Protection*
protège des attaques par réutilisation de code (ROP etc.)

Exemple en situation

CVE-2016-0728[2]

- mauvaise gestion du compteur de référence, composant *keyring*
- exploitable par un utilisateur non privilégié
- libération d'une zone mémoire tout en gardant un pointeur
- allocation mémoire au même endroit, contenu quasi-arbitraire
- déréférencement d'un pointeur de fonction
- exécution de code en mode noyau

Protection par PaX

REFCOUNT empêche le débordement du compteur de référence

USERCOPY empêche l'allocation dans au même endroit

UDEREF complique très fortement l'exploitation en empéchant l'accès à l'espace utilisateur

Qui utilise grsecurity ?

- qui utilise Linux ?

Qui utilise grsecurity ?

- qui utilise Linux ?
- qui utilise grsecurity ?

Pourquoi pas tout le monde ?

Quelques idées

- pas inclu dans le noyau *mainline*
- problème de compatiblité ou de support
- diminution de performances
- charge de maintenance (configuration, compilation etc.)
  en général c'est la distribution qui gère

Comment aider les sysadmins

Idées

- faciliter la compilation et les mises à jour : `grsec-config`
- intégrer grsecurity dans les distributions :
  - Gentoo
  - Arch
  - **Debian**
- intégrer les composants *mainline*

Mot clé : pragmatisme

grsec-config [3]

Qu'est-ce donc ?

- deux scripts pour faciliter la vie des sysadmins
- pour compiler des noyaux grsec

Cible : sysadmins et packaging d'entreprise

- facilité d'intégration dans un environnement Debian (ou RPM)
- upload possible dans un miroir local

Comment on s'en sert ?

get-grsec à lancer depuis un clone de linux.git

1. télécharge le dernier patch grsecurity
2. créé une nouvelle branche locale
3. applique le patch

kconfig

- fusion de morceaux de .config
- utile pour garder séparées
  - configuration matérielle
  - configuration logicielle
  - configuration grsecurity
- vient du paquet source Debian

Qu'est-ce que ça donne ?

Simple comme bonjour

1. récupération du patch et application

   `get-grsec.sh`
2. (régénération de la configuration)

   `kconfig.py .config /boot/config-4.4.0-1-amd64 ../grsec`
3. compilation

   `make deb-pkg [ou rpm-pkg, ou autre]`
4. (envoi du résultat sur le miroir local)

Rinse, repeat

`https://deb.li/grseccfg`

Distributions

Plusieurs distributions proposent un kernel grsec

Gentoo il faut évidemment le compiler soit même

Arch *rolling release*

Debian unstable + stable-backports

Autres ? (*aucune idée*)

problématique de la disponibilité de grsecurity

- plus de patches *stables* en libre accès
- uniquement pour le noyau courant
- les patches pour les noyaux LTS sont réservés aux sponsors

Debian

linux-grsec

- fork/rebase du paquet src :linux
- inclus le patch grsecurity
- inclus les patches Debian quand il n'y a pas de conflit
- utilisation de la configuration Debian
- amd64 et i386 pour l'instant (aide bienvenue pour ARM)
- suivi du cycle de grsecurity
- disponible dans sid et jessie-backports

Demo

```
root@jessie: echo "deb http://ftp.fr.debian.org/debian jessie-
    backports main" >> /etc/apt/sources.list.d/backports.list
root@jessie: apt update
root@jessie: apt install linux-image-grsec-amd64 paxctld
```

Spécificités d'un noyau grsecurity

- Comportements parfois surprenants
- Restrictions parfois trop fortes

Exemple : MPROTECT

Tout ce qui utilise du JIT a besoin de mémoire RWX :

- JVM
- moteur javascript (navigateurs web)
- autres exemples divers (python/libffi)

```
grsec: denied RWX mmap of <anonymous mapping> by /usr/lib/chromium/
chromium
```

Possibilité d'adapter le comportement du noyau

Adaptation pour tout le système

Exemple de Debian (noyau pré-compilé)

Utilisation de sysctls :

/etc/sysctl.d/grsec.conf

```
[...]
## Kernel Auditing
kernel.grsecurity.exec_logging = 0
kernel.grsecurity.audit_chdir = 0
[...]
# Once you're satisfied, set grsec_lock to 1 so noone can change
# grsec sysctl on a running system
kernel.grsecurity.grsec_lock = 1
```

Adaptation pour un exécutable

Marquage du binaire

- paxctld

/etc/paxctld.conf

```
/usr/lib/chromium/chromium  m
```

- édition de l'ELF

```
paxctl -cm /usr/lib/chromium/chromium
```

- ajout d'un attribut étendu

```
setfattr -n user.pax.flags -v m /usr/lib/chromium/chromium
```

- utilisation de RBAC

/etc/grsec/policy

```
subject: /usr/lib/chromium/chromium
  -PAX_MPROTECT
```

Kernel self protection project [4]

Inclus dans le noyau 4.6

- *page poisoning* n'est plus une option de debug
- introduction de l'attribut *__ro_after_init*
  zone mémoire en lecture seule après l'initialisation
- kASLR pour l'architecture arm64
- protections mémoires activées par défaut sur ARMv7+ et arm64 (`CONFIG_DEBUG_RODATA`), obligatoires sur x86

Attention à l'aspect marketing

- __ro_after_init uniquement positionné sur vDSO pour l'instant
- `CONFIG_DEBUG_RODATA` déjà activé par les distributions

Kernel self protection project [4]

Prochains noyaux

4.7
- nouvelles améliorations pour kASLR x86
- MIPS kASLR
- LoadPin LSM
  vérification de l'origine des firmware et modules

4.8
- autres améliorations pour kASLR x86
- plugins GCC
- randomisation des structures par build (RANDKSTRUCT)

Conclusion

- pensez à durcir votre noyau
- grsecurity n’est pas si compliqué (et peut vous sauver la vie)
- certains outils peuvent vous faciliter la vie
- du support est disponible (forum ou dédié sponsors)

Questions ?

References

B. Spengler and the Pax Team, “Frequently asked questions about rap.”
https://grsecurity.net/rap_faq.php.

P. P. R. Team, “Analysis and exploitation of a linux kernel vulnerability.”
http://perception-point.io/2016/01/14/analysis-and-exploitation-of-a-linux-kernel-vulnerability-cve-2016-0728/.

Y.-A. Perez, “Tools to build a linux kernel with grsecurity patch for debian.”
https://anonscm.debian.org/cgit/users/corsac/grsec/debian-grsec-config.git/.

K. Cook, “Kernel self protection project.”
http://www.openwall.com/lists/kernel-hardening/2015/11/05/1.

Other stuff to look at on a Linux system

Optional / time permitting

More hardening : userland

- systemd (namespaces, capabilities)
- LSM
- RBAC