Préface

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Bon symposium,
Gérard Menvussa, pour le comité d'Organisation.

Comité d'organisation

Nicolas Bareil	Airbus Group
Mathieu Blanc	CEA DAM
Pierre Capillon	ANSSI
Olivier Courtay	DGA-MI
Olivier Levillain	ANSSI
Benjamin Morin	ANSSI
Nicolas Prigent	CentraleSupélec
Frédéric Tronel	CentraleSupélec

Comité de programme

Erwan Abgrall	DGA MI
Jean-Philippe Aumasson	Kudelski Security
Nicolas Bareil	Airbus Group
Mathieu Blanc	CEA DAM
Aurélien Bordes	ANSSI
Pierre Capillon	ANSSI
Pierre Chifflier	ANSSI
Olivier Courtay	DGA-MI
Guillaume Delugré	Quarkslab
Fabrice Desclaux	CEA DAM
Aurélien Francillon	EURECOM
Isabelle Kraemer	Orange
Éric Leblond	Stamus Networks
Olivier Levillain	ANSSI
Florent Marceau	CERT-LEXSI
Benjamin Morin	ANSSI
Nicolas Prigent	CentraleSupélec
Raphaël Rigo	Airbus Group
Tiphaine Romand	Orange
Axel Tillequin	Airbus Group
Frédéric Tronel	CentraleSupélec
Marion Videau	Quarkslab et LORIA
Sarah Zennou	Airbus Group

Partenaires

Airbus Group - ANSSI - CEA - CentraleSupélec - DGA - MISC/Le magazine 100% Sécurité informatique - Université de Rennes 1

Table des matières

Conférences

Conférences

Noyaux Linux durcis

Yves-Alexis Perez
corsac@corsac.net

Résumé Grsecurity[5] est un patch de durcissement Linux bien connu. Il inclut entre autres le patch PaX[1], qui a introduit certaines techniques de durcissement système telles que la randomisation de l'espace mémoire (ASLR), intégrées plus tard dans la plupart des OS. Le patch grsecurity est toujours externe (il ne fait pas partie du noyau Linux), la plupart des fonctionnalités et du durcissement qu'il apporte ne bénéficient donc pas aux utilisateurs finaux qui, pour la plupart, utilisent les noyaux binaires de leur distribution.
Même si des efforts récents cherchent à améliorer la situation du noyau Linux, c'est toujours le patch grsecurity qui représente l'état de l'art de la sécurité kernel dans le monde Linux, et simplifier son adoption est une étape importante pour en améliorer la sécurité globale.

1 *Durcissement* Linux ?

Dans un contexte informatique, on entend habituellement par *durcissement* un ensemble de mesures permettant d'empêcher des attaques de réussir. La méthode standard pour empêcher l'exploitation d'une vulnérabilité est normalement de corriger celle-ci. C'est évidemment la meilleure solution, mais ce n'est appliquable qu'à cette vulnérabilité particulière. Il est possible qu'une vulnérabilité du même type existe à un autre endroit du code, et soit donc exploitable de la même manière.

Il existe déjà un certain nombre d'outils permettant de détecter et corriger des vulnérabilités. On peut penser aux outils d'analyse statique (coccinelle[2] ou coverity[3]) ou dynamiques, comme le *fuzzing* d'appels systèmes (trinity[4] ou syzkaller[5]), l'instrumentation de l'allocation mémoire par KASAN[6] etc. Il s'agit ici d'identifier des vulnérabilités, de façon automatique ou semi-automatique, de préférence avant qu'elles se retrouvent exploitables en production.

[1] dont l'auteur, The PaX Team, a fait la conférence d'ouverture du SSTIC 2012[7]
[2] http://coccinelle.lip6.fr/
[3] https://scan.coverity.com/projects/linux?tab=overview
[4] http://codemonkey.org.uk/projects/trinity/
[5] https://github.com/google/syzkaller
[6] https://www.kernel.org/doc/Documentation/kasan.txt

Le durcissement du noyau est une façon d'empêcher (ou rendre très difficile) non pas simplement l'exploitation d'**une** vulnérabilité, mais de **classes** entière de vulnérabilité, lors de l'exécution du système, et donc de protéger un noyau déjà en production.

Le noyau actuel possède déjà certaines fonctionnalités qui permettent de rendre un système plus robuste aux attaques. En espace utilisateur, on peut penser en particulier à la répartition aléaloire de l'espace d'adressage (*ASLR*), qui permet, sans l'empêcher complêtement, de compliquer nettement la tâche d'un attaquant souhaitant exploiter une vulnérabilité, que ce soit en important son propre code ou en essayant de d'utiliser celui existant.

Le durcissement appliqué au noyau lui-même est encore assez faible. Le noyau utilise les fonctionnalités offertes par le matériel, en particulier concernant la non exécutabilité de la mémoire (*NX*/*XN*, voire *SMEP* et *PXN*[7]), mais cherche peu à se protéger lui-même des attaques.

2 Grsecurity

Grsecurity est un ensemble de plusieurs fonctionnalités rassemblées au sein d'un patch pour le noyau Linux. Il contient en particulier les éléments suivants :

- PaX
- RBAC (*role-based access control*), un mécanisme de contrôle d'accès obligatoire
- Des durcissements divers (mémoire, système de fichier, réseau, USB)
- Des améliorations dans le système de journalisation et d'audit
- Des *backports* de patches de sécurité spécifiques
- Des *backports* de patches fonctionnels, dans certaines branches stables

Les parties les plus intéressantes ici sont PaX et les « durcissements divers ». S'il n'est pas question ici de rentrer dans les détails des différentes fonctionnalités de PaX, on peut tout de même citer les plus grandes :

`NOEXEC` implémentation logicielle du bit NX

`MPROTECT` W⊕X au niveau des pages mémoires, interdit les projections mémoires accessibles à la fois en écriture et exécution, complète `NOEXEC`

`KERNEXEC` version noyau de `NOEXEC` + `MPROTECT` ; empêche l'injection et l'exécution de code arbitraire dans le noyau courant

[7] clairement inspirés de la fonctionnalité `UDEREF`, fournie par PaX

`ASLR` précède l'implémentation du noyau Linux[8]) et en est toujours une version améliorée

`UDEREF` empêche le déréférencement de pointeurs *userland* par le noyau[9]

`CONSTIFY` *constification* des structures ne contenant que des pointeurs de fonctions, implémenté à l'aide d'un plugin gcc

`SIZE_OVERFLOW` plugin gcc permettant de détecter des *integer overflows*, en particulier quand il s'agit de taille d'allocations mémoire

Les fonctionnalités les plus intéressantes pour la protection du noyau sont `KERNEXEC`, `UDEREF`, `CONSTIFY` et `SIZE_OVERFLOW`. Celles-ci n'ont pas leur équivalent dans *mainline*, même si certains efforts sont en cours pour en intégrer une partie.

3 Cas concret de vulnérabilité évitée

Parmi les vulnérabilités récentes[10] ayant touché le noyau, on peut s'arrêter sur le CVE-2016-0728[6], élévation de privilège[11] dans le composant *keyring*. La vulnérabilité elle-même est un problème de gestion de référence dans les *keyrings*. Sous certaines conditions (contrôlables par un attaquant local), un compteur de référence n'est pas décrémenté lorsqu'une référence est libérée. En exploitant ce problème de façon répétée, un attaquant peut finir par faire boucler le compteur de référence et le remettre à zéro, provoquant la libération de la zone mémoire concernée alors qu'un composant du système possède toujours un pointeur vers cette zone mémoire. Il devient alors possible d'exploiter ce *use-after-free* en allouant de la mémoire à cet endroit et en remplaçant un pointeur de fonction afin de permettre de l'exécution de code arbitraire.

Dans un noyau typique et une architecture très récente, SMEP et SMAP (ou PXN/PAN sur arm) permettent de rendre l'exploitation plus délicate, mais pas impossible : en effet, l'attaquant peut essayer une chaîne de ROP à l'intérieur du noyau pour éviter d'accéder directement à l'espace utilisateur.

Sur un noyau *grsec*, plusieurs barrières permettent d'empêcher l'exploitation de cette vulnérabilité. La principale bloque totalement la vulnérabilité, l'autre permettrait de compliquer fortement son exploitation :

[8] ainsi d'ailleurs que de toutes les autres
[9] à rapprocher de SMEP/SMAP ou PXN/PAN
[10] `http://www.openwall.com/lists/kernel-hardening/2016/01/19/1`
[11] ou plutôt une exécution de code arbitraire en mode noyau

REFCOUNT *PaX Reference counter protection*[1] empêche le compteur de référence du *keyring* de boucler (il reste bloqué à `INT_MAX`) ; la vulnérabilité est donc complètement empêchée ;

UDEREF de la même façon que SMEP et SMAP, permet d'empêcher d'accéder à l'espace mémoire utilisateur, et complique donc nettement la tâche d'un attaquant, le forçant à utiliser uniquement de la mémoire kernel ; contrairement à SMEP, il n'est pas possible de désactiver `UDEREF` à l'exécution.

Démocratisation de grsecurity

Si grsecurity permet d'éviter des vulnérabilités, on peut se demander pourquoi tout le monde ne l'utilise pas. S'il est délicat d'avoir des certitudes sur toutes les situations, on peut avancer plusieurs hypothèses :

- il s'agit d'un patch externe (non intégré au noyau Linux mainline) ;
- certaines fonctionnalités sont non triviales à utiliser et impliquent donc une montée en compétence et une prise en compte globale (en particulier concernant la compatibilité de certains logiciels) ;
- certaines fonctionnalités ont un impact sur les performances.

Le fait qu'il s'agisse d'un patch externe pose sans doute un certain nombre de problèmes aux administrateurs système :

- il est absent de la plupart des noyaux de distributions ;
- utiliser un noyau externe peut faire perdre les éventuelles certifications de compatibilité entre une distribution et un matériel ;
- utiliser un noyau externe peut faire perdre l'accès au support technique de la distribution ;
- configurer et compiler un noyau est non trivial, en particulier le maintien dans le temps et sur un parc potentiellement hétérogène.

Afin d'abaisser les barrières à l'utilisation de grsecurity, que ce soit de façon personnelle ou professionnelle, y compris sur des grands parcs, l'auteur a mené deux initiatives.

3.1 grsec-config[4]

Il s'agit d'un projet dont le but est de faciliter la récupération, la configuration, la compilation et la distribution locale d'un noyau grsecurity.

Un script (`get-grsec.sh`) permet de récupérer le dernier patch grsecurity et l'appliquer à la version correspondante du noyau (dans un clone local du git du noyau Linux). Un deuxième script (`kconfig.py`) permet

de générer éventuellement une configuration adaptée en fusionnant divers *snippets* de configuration (par exemple un pour grsecurity, un pour la cible matérielle, un pour la cible logicielle). Il est enfin possible de générer une image du noyau Linux sous différent format afin de le déployer directement ou via un système de miroir locaux, adaptés à la distribution.

3.2 linux-grsec

Il s'agit ici d'intégrer directement dans une distribution une version packagée du noyau grsecurity. Peu de distributions fournissent ceci, on peut citer en particulier *Gentoo hardened* mais il ne s'agit pas d'une version compilée, ainsi que Arch[12].

La distribution Debian fait depuis peu partie des distributions fournissant grsecurity[13] , même s'il ne s'agit pour l'instant que de la version *unstable*. L'inclusion dans la prochaine version stable (*stretch*) est rendue délicate par le fait qu'il n'est plus possible d'accéder librement au patch *grsecurity* pour les version stables du noyau Linux.

Il devient possible d'installer un noyau patché avec :

```
~# apt install linux-image-grsec-amd64
```

Listing 1. Installation d'un noyau grsec sous Debian *sid*

4 Mise en œuvre d'un noyau grsecurity

Une fois un noyau grsecurity installé, son utilisation peut poser quelques challenges. En se plaçant dans cas d'une distribution Debian, on peut indiquer quelques pistes à suivre pour faciliter l'intégration. Ces conseils sont dans l'ensemble applicables quelle que soit la distribution, mais peuvent nécessiter des adaptations. Ils ne doivent pas être pris au pied de la lettre, comme une référence immuable, mais évidemment adaptés au contexte.

4.1 Utilisation des `sysctl`

Le patch grsecurity supporte la configuration dynamique d'un certain nombre de paramètres, via l'utilitaire `sysctl`. Un fichier de configuration[14] est fourni lors de l'installation du noyau, et permet d'ajuster son

[12] Voir en particulier `https://wiki.archlinux.org/index.php/Grsecurity`
[13] `https://tracker.debian.org/pkg/linux-grsec`
[14] `/etc/sysctl.d/grsec.conf`

comportement suivant l'utilisation du système. La plupart des paramètres sont activés de base sous Debian (*secure by defaul*), parmi ceux qu'il est souhaitable de regarder on peut noter :

```
# Disable logging of all execve() and chdir()
kernel.grsecurity.exec_logging = 0
kernel.grsecurity.audit_chdir = 0
# Lock dynamic configuration
kernel.grsecurity.grsec_lock = 1
```

Listing 2. Configuration dynamique de grsec

4.2 Marquage d'exécutables

L'utilisation de `MPROTECT` permet d'appliquer à l'espace utilisateur le principe du W⊕X et de fortement limiter la possibilité pour un attaquant d'importer du code à lui pour exploiter une vulnérabilité. Malheureusement, cette protection empêche totalement de fonctionner les outils utilisant de la compilation *just-in-time* (JIT), puisque par principe le programme compile du code pendant l'exécution, et doit donc pouvoir exécuter du code qu'il vient de générer. On trouve ce type de fonctionnement dans la plupart des environnement d'exécution pour langage interprété ou à *bytecode* (python, java, javascript etc.) car il permet d'augmenter fortement les performances.

`MPROTECT` empêche donc habituellement de fonctionner une partie des programmes python, les différents navigateurs internet, la JVM, etc. Il est possible, dans ce cas, d'ajouter des exceptions à `MPROTECT` pour autoriser certains binaires à avoir des projections mémoires accessibles à la fois en écriture et exécution.

Suivant les fonctionnalités présentes dans le noyau, ces exceptions peuvent être ajoutées via l'utilisation d'attributs étendus du système de fichier ou en marquant le binaire lui-même.

La première méthode nécessite uniquement l'utilisation de l'outil générique `attr`[15]. Par exemple, pour autoriser *chromium* à avoir des projections mémoires W|X, on utilisera la commande suivante :

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

Listing 3. Marquage de chromium

Il existe aussi un certain nombre d'utilitaires spécifiques (`paxctl`, `paxctld`[16]) qui permettent de marquer les binaires. Il est enfin possible

[15] qui permet d'écrire des attributs étendus sur un système de fichier les supportant
[16] démon qui marque les binaires automatiquement, permettant de maintenir les exceptions lorsque les binaires sont mis à jour

d'autoriser ces exceptions au titre d'une politique de sécurité `RBAC`, mais ce n'est pas abordé ici.

5 Kernel self-protection project

L'un des obstacles majeurs à l'utilisation de noyaux durcis, on l'a déjà dit, est le fait que ce durcissement soit quasiment inexistant dans le noyau *mainline*. Une certaine prise de conscience de cet état de fait parmi des développeurs du noyau Linux a mené au lancement en novembre 2015[2] du *Kernel self-protection project*[3] qui vise à intégrer des mesures d'auto-protection (en partie issues de PaX et grsecurity).

Parmi les mesures candidates à l'intégration, on peut trouver l'infrastructure de plugins gcc, afin de pouvoir intégrer à terme `PAX_SIZE_OVERFLOW`, `PAX_CONSTIFY` et `PAX_REFCOUNT`. `KERNEXEC` et `UDEREF` sont aussi intéressantes mais posent un certain nombre de challenges non encore résolus, l'idée étant de commencer petit.

En mars 2016, plusieurs patches ont déjà été postés sur la liste, voire intégrés à l'arbre *linux-next*, mais aucun n'a encore été intégré à la branche de Linus Torvalds. Conclusion

Il existe à l'heure actuelle un certain nombre de moyens de durcir un noyau Linux contre des attaques, que ce soit via des vulnérabilités courantes ou des attaques plus ciblées. Si patcher et reconfigurer soi-même un noyau Linux reste difficile pour une bonne partie des administrateurs et utilisateurs, il existe des moyens de simplifier ces opérations. L'intégration du patch dans plusieurs distributions Linux rend l'installation nettement plus facile, et le projet d'en intégrer une partie des fonctionnalités dans le noyau *mainline* devrait encore améliorer ceci.

Une fois le patch en production, un certain changement de paradigme est nécessaire, afin de prendre en compte la sécurité du système de façon globale : propagation du W⊕X au système de fichier, prise en compte des alertes PaX.

Certaines adaptations du systèmes sont aussi nécessaires[17] mais ces adaptations sont bien documentées et, si elles peuvent affaiblir certaines protections, permettent tout de même de bénéficier d'un noyau durci.

Références

1. Rodrigo Branco. Pax reference counter protection documentation. `https://forums.grsecurity.net/viewtopic.php?f=7&t=4173`.

[17] par exemple pour les programmes utilisant de la compilation *just-in-time* et nécessitant donc des exceptions au W⊕X)

2. Kees Cook. Kernel self protection project. http://www.openwall.com/lists/kernel-hardening/2015/11/05/1.
3. Kees Cook. Kernel self protection project wiki. http://kernsec.org/wiki/index.php/Kernel_Self_Protection_Project.
4. Yves-Alexis Perez. Tools to build a linux kernel with grsecurity patch for debian. https://anonscm.debian.org/cgit/users/corsac/grsec/debian-grsec-config.git/.
5. Bradley Spengler. Grsecurity. https://grsecurity.net.
6. Perception Point Research 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/.
7. The Pax Team. 20 years of pax. https://www.sstic.org/2012/presentation/Keynote_2012/.

Index des auteurs