Durcissement système à l'aide de systemd : Réduction de l'impact de plusieurs vulnérabilités majeures

Timothée Ravier
timothee.ravier@ssi.gouv.fr

ANSSI

Résumé. Cet article présente trois mécanismes de sécurité du noyau Linux et détaille ensuite leur mise en action à l'aide des fonctionnalités proposées par *systemd*. Plusieurs cas pratiques sont alors étudiés pour valider l'efficacité de ces protections face à trois vulnérabilités majeures.

1 Introduction

Comme tous les logiciels, les projets *open source* ne sont pas exempts de vulnérabilités. Celles-ci sont généralement résolues rapidement et les correctifs intégrés dans les mises à jour des différentes distributions Linux.

Malheureusement, le temps séparant l'annonce de la résolution d'une vulnérabilité n'est pas prévisible. Les administrateurs des systèmes affectés par ces vulnérabilités ne disposent pas non plus toujours du champ libre pour réaliser la mise à jour de systèmes en production. Comme certaines vulnérabilités sont exploitées avant même d'être annoncées publiquement, la sécurité d'un système ne peut pas reposer uniquement sur l'application diligente des mises à jour. Il est donc nécessaire de durcir l'environnement dans lequel évoluent les programmes d'un système.

Cet article présente trois mécanismes de sécurité du noyau Linux. Il détaille ensuite leur mise en action à l'aide de *systemd*. Finalement, il étudie trois cas pratiques de vulnérabilités dont l'impact peut être atténué ou annulé à l'aide des éléments présentés.

2 Mécanismes de sécurité proposés par le noyau Linux

La montée en puissance des technologies à base de conteneurs a contribué à l'émergence et la maturation de plusieurs mécanismes de sécurité dans le noyau Linux. *systemd* propose une interface simple pour les utiliser.

En tant que remplaçant de *SysVinit*, *systemd* gère le lancement et le suivit des services d'un système. Tous les services sont exécutés à partir

d'un fils du processus *init* (PID 1). L'environnement dans lequel s'exécute chaque service est donc contrôlé et de nombreuses restrictions peuvent être mises en place sans nécessiter de modification au niveau des applications.

Cette section présente d'abord brièvement les commandes pour éditer les options mises à disposition par *systemd* pour changer l'utilisateur et le groupe sous lequel s'exécute un service. Elle détaille ensuite le fonctionnement de *seccomp-bpf*, des espaces de nom de point de montage et la gestion des capacités Linux.

2.1 Utilisateurs et groupes non privilégiés

Chaque service géré par *systemd* est configuré à l'aide d'un fichier nommé *unit*. La configuration actuelle d'un service est affichée avec la commande :

```
# systemctl cat mon-service.service
```

La commande suivante permet d'ajouter des options supplémentaires à l'*unit* correspondant à ce service :

```
# systemctl edit mon-service.service
```

Si aucune modification n'a été faite par l'administrateur auparavant, ce fichier sera vide. Ces modifications s'ajoutent et supplantent la configuration fournie par la distribution. Elles sont stockées indépendamment et ne sont pas affectées par les mises à jour du système.

Voici les options pour lancer un service sous un utilisateur et un groupe non privilégié :

```
[Service]
User=http
Group=www
```

Pour que ce changement soit pris en compte, il faut demander à *systemd* de recharger sa configuration (1) puis redémarrer le service concerné (2) :

```
1 # systemctl daemon-reload
2 # systemctl restart mon-service.service
```

2.2 Filtrage d'appels systèmes *(seccomp-bpf)*

L'une des méthodes d'exploitation du noyau Linux consiste à détourner le comportement d'un appel système pour obtenir plus de privilèges en exploitant une vulnérabilité dans le code du noyau [4]. Or, les noyaux

fournis par les distributions Linux sont généralement configurés pour offrir le maximum de possibilités aux développeurs d'applications. Toutes ces options augmentent significativement la surface d'attaque du noyau.

Depuis le noyaux 3.17, le mécanisme nommé *seccomp-bpf* permet à un processus de se retirer l'accès aux appels système dont il n'a pas l'usage. Cette restriction s'appliquera aussi à tous ses processus fils.

systemd propose une série d'options permettant de mettre en place des filtres *seccomp-bpf* sur les services dont il a la charge. L'option nommée `SystemCallFilter` spécifie une liste blanche d'appels système laissés disponibles pour une application. Le fonctionnement en mode liste noire est obtenu en préfixant la liste par un « `~` ». L'option suivante permet par exemple de bloquer l'utilisation de l'appel système ptrace :

```
[Service]
SystemCallFilter=~ptrace
```

Depuis la version 231 de *systemd*, les appels système d'une même famille (gestion de l'horloge, accès au réseau, contrôle des processus) peuvent être bloqués à l'aide de groupes[1]. On peut noter que le sous système *perf* du noyau Linux est, de par sa complexité, une source de vulnérabilités [3]. L'appel système correspondant est inclus avec *ptrace* dans le groupe `@debug`, que l'on peut bloquer avec l'option :

```
[Service]
SystemCallFilter=~@debug
```

Enfin, remarquons qu'il est possible de contourner, par l'intermédiaire de l'appel système *ptrace*, l'application de ces filtres sur les noyaux inférieurs à la version 4.8 [5]. Il faut donc dans ce cas veiller à inclure systématiquement cet appel système dans la liste noire (ou l'exclure de la liste blanche).

2.3 Isolation avec les espaces de nom de point de montage *(mount namespace)*

Cette fonctionnalité, disponible depuis le noyau Linux 2.6.15, permet de donner à un processus et ses fils une vue distincte de l'arborescence du système de fichier. Les modifications de points de montage dans cet espace de nom n'affectent pas les autres processus du système. Il est alors possible

[1] Recopier manuellement la liste d'appels système filtrés par un groupe d'une version plus récente de *systemd* permet d'en reproduire le comportement sur une version précédente ne supportant pas l'usage des groupes.

de monter un dossier vide par dessus un dossier arbitraire pour un service en particulier sans impact sur le reste du système[2] :

```
[Service]
InaccessiblePaths=/etc/secrets
```

Plusieurs options sont disponibles pour rendre des portions du système accessibles en lecture seule. Les répertoires `/usr`, `/boot` et `/etc` sont protégés par l'option `ProtectSystem=full`, `/home` et `/root` par `ProtectHome=yes`, certains répertoires de `/proc/sys` et `/sys` par `ProtectKernelTunables=yes`[3].

Il faut en revanche s'assurer que ces opérations de montage ne seront pas réversibles par le service en cas de compromission. Ce point est l'objet de la section suivante.

2.4 Plafond appliqué aux capacités *(capability bounding set)*

Certaines interactions avec le système nécessitent qu'un processus soit privilégié : configuration du réseau, chargement de module noyau, changement des limites mémoire ou CPU, etc. Mais rares sont les programmes faisant usage de l'ensemble de ces privilèges simultanément.

Ainsi, pour réduire les capacités à disposition d'un processus privilégié, on peut utiliser le *capability bounding set*, qui se comporte comme un plafond limitant l'acquisition et l'usage des capacités. La suppression de la capacité `CAP_SYS_ADMIN` combinée au filtrage des appels système liés à *mount* avec *seccomp-bpf* interdit ainsi toute manipulation des points de montage[4] :

```
[Service]
CapabilityBoundingSet=~CAP_SYS_ADMIN
SystemCallFilter=~@mount
```

Mais il est aussi possible d'accorder à un processus non privilégié un sous ensemble des privilèges généralement accordés à *root*. On peut par exemple accorder la capacité `CAP_NET_BIND_SERVICE` à un serveur web s'exécutant sous un utilisateur non privilégié pour lui permettre d'écouter les connections entrantes sur les ports inférieurs à 1024 (dont 80 et 443)[5] :

```
[Service]
AmbientCapabilities=CAP_NET_BIND_SERVICE
```

[2] L'option est nommée `InaccessibleDirectories` pour les versions inférieures à 231.
[3] L'option `ProtectKernelTunables` n'est disponible qu'à partir de la version 232.
[4] Supprimer `CAP_SYS_ADMIN` n'est pas toujours suffisant. Pour plus de détails, voir [1].
[5] Cette fonctionnalité (capacités *ambient*) n'est disponible qu'à partir du noyau 4.3 et de la version 230 de *systemd*.

3 Étude de plusieurs vulnérabilités et de contre-mesures

Cette section présente trois vulnérabilités majeures et les contre-mesures permettant de limiter voire d'annuler leur impact, illustrant ainsi l'usage des mécanismes de sécurité présentés dans la section précédente.

3.1 Du bon usage d'un *socket*

La vulnérabilité CVE-2016-8655 [8] est une situation de compétition *(race condition)* dans l'implémentation des *sockets* de type `AF_PACKET` dans le noyau Linux. Elle peut conduire à l'utilisation d'une zone de mémoire n'étant plus allouée *(use after free)*. L'utilisateur souhaitant créer un *socket* de type `AF_PACKET` doit disposer de la capacité Linux `CAP_NET_RAW`. Ces capacités peuvent être obtenues en utilisant la fonctionnalité d'espace de nom utilisateur non privilégié introduite dans le noyau 3.8.

Options possibles pour atténuer l'impact : Plusieurs options se présentent pour empêcher l'exploitation de cette vulnérabilité [7] :

- désactiver ou restreindre l'usage des espaces de nom utilisateur ;
- bloquer l'acquisition de la capacité `CAP_NET_RAW` ;
- bloquer l'utilisation de sockets de type `AF_PACKET`.

Pour désactiver les espaces de nom utilisateur, il est nécessaire de recompiler le noyau Linux. Certaines distributions proposent d'en limiter l'usage aux seuls utilisateurs privilégiés à l'aide d'un paramètre de configuration *sysctl*. Cette option s'applique alors à l'ensemble du système. Un redémarrage est nécessaire pour que tous les processus soient affectés.

Configuration avec *systemd* : Un filtre *seccomp-bpf* est capable de filtrer certains appels système en fonction des valeurs qui leurs sont passées en paramètres. L'option `RestrictAddressFamilies` permet ainsi de restreindre l'utilisation des *sockets* de type `AF_PACKET` :

```
[Service]
RestrictAddressFamilies=~AF_PACKET
```

Cette solution minimale peut être étendue pour n'autoriser, à l'inverse, que les types les plus couramment utilisés :

```
[Service]
RestrictAddressFamilies=AF_INET AF_INET6 AF_UNIX
```

L'option suivante bloque l'obtention de la capacité `CAP_NET_RAW` :

```
[Service]
CapabilityBoundingSet=~CAP_NET_RAW
```

Enfin, les versions récentes de *systemd*[6] proposent une option pour restreindre directement l'utilisation des espaces de noms utilisateur :

```
[Service]
RestrictNamespaces=~user
```

3.2 Confinement de *Bovinae* : *« Dirty CoW »*

La vulnérabilité CVE-2016-5195 affecte tous les noyaux Linux depuis la version 2.6.22, publiée en 2007. Le noyau Linux utilise le principe du *copy-on-write* pour limiter la consommation en mémoire des processus d'un système. Cette vulnérabilité a pour origine une situation de compétition dans la réalisation de la duplication des zones de mémoire partagées. Sous certaines conditions, un processus malveillant peut forcer le noyau à écrire dans la mémoire avant que l'opération de copie n'aie pu être réalisée. Il est alors possible d'effectuer plusieurs actions normalement impossibles pour un utilisateur non privilégié :

- écrire dans un fichier accessible en lecture seule ;
- modifier les zones de mémoires communes à tous les processus d'un système (par exemple le vDSO).

La situation de compétition est déclenchée par l'appel système `madvise`. L'exploitation nécessite l'usage d'une méthode d'accès direct *(raw)* à la mémoire d'un processus, comme le fichier virtuel `/proc/self/mem` ou la fonction `PTRACE_POKEDATA` de l'appel système `ptrace`[7].

Options possibles pour atténuer l'impact : Plusieurs options se présentent pour complexifier l'exploitation de cette vulnérabilité [6] :

- bloquer l'appel système `madvise` ;
- bloquer l'appel système `ptrace` ;
- supprimer l'accès au système de fichier virtuel `/proc` ;
- supprimer l'accès au périphériques matériels exposés dans `/dev`.

[6] À partir de la version 233.

[7] Certain opérations sur des périphériques accessibles dans `/dev` peuvent aussi potentiellement être source de telles opérations.

Bien qu'il soit possible de bloquer l'appel système `madvise` [2], de nombreux programmes en font un usage légitime (notamment par l'intermédiaire de la *glibc*). Les autres solutions seront donc privilégiées.

Configuration avec *systemd* : Il est possible de masquer le dossier `/proc` en utilisant les espaces de nom de point de montage[8] :

```
[Service]
InaccessiblePaths=-/proc
```

L'option `PrivateDevices` permet de limiter au strict minimum les périphériques exposés dans le répertoire `/dev` pour un service. Les fichiers spéciaux `/dev/null`, `/dev/urandom` et `/dev/zero` restent disponibles :

```
[Service]
PrivateDevices=yes
```

Enfin il est possible de bloquer l'appel système `ptrace` :

```
[Service]
SystemCallFilter=~ptrace
```

3.3 *systemd vs the crashing tweet*

La vulnérabilité CVE-2016-7795 [9] affecte le démon principal du projet *systemd*.

systemd met à disposition des services du système un *socket* de notification qu'ils peuvent utiliser comme *watchdog* logiciel : si un service n'envoie plus de notifications à intervalles réguliers alors il est redémarré. Cette fonctionnalité nécessite d'être activée explicitement pour chaque service mais ce *socket* est accessible à l'ensemble des utilisateurs du système.

Une erreur de logique permet de lever une *assertion* et de forcer ainsi la mise en pause de l'exécution du démon PID 1. De nombreuses fonctionnalités ne sont alors plus disponibles sur le système (gestion des services, redémarrage du système) et les opérations de *login* ne se complètent qu'après l'expiration d'un *timeout*.

Options possibles pour atténuer l'impact : La solution la plus simple est de supprimer l'accès à ce *socket* pour les services qui n'en ont pas l'usage. Le *socket* `notify` est situé dans le dossier `/run/systemd`.

[8] Pour plus de détails : `https://lists.freedesktop.org/archives/systemd-devel/2019-April/038634.html`

Configuration avec *systemd* : L'option `InaccessiblePaths`, qui rend un dossier inaccessible pour un service en particulier, est adaptée :

```
[Service]
InaccessiblePaths=/run/systemd
```

4 Conclusion

Nous avons vu qu'un système peut être durci avec quelques mesures simples qui limitent sérieusement l'impact de vulnérabilités pourtant critiques. La mise en place progressive de ces méthodes de durcissement système est facilitée par l'interface proposée par le projet *systemd*. Ces techniques de protection ne sont désormais plus limitées aux systèmes spécialisés et peuvent être déployées sur la majeure partie des distributions Linux.

5 Remerciements

Je remercie mes relecteurs (Philippe Trébuchet, Mickaël Salaün, Yves-Alexis Perez, Thomas Letan, Nicolas Vivet) pour leurs remarques pertinentes lors de la rédaction de cet article.

Références

1. Brad Spengler. False Boundaries and Arbitrary Code Execution. `https://forums.grsecurity.net/viewtopic.php?f=7&t=2522`, 2 janvier 2011.
2. David Timothy Strauss. Mitigating dirtyc0w with systemd. `https://medium.com/@davidtstrauss/mitigating-dirtyc0w-with-systemd-5f3bad6f376a#.h5fwjhn2r`, 21 octobre 2016.
3. Jeff Vander Stoep. Android : protecting the kernel. `http://events.linuxfoundation.org/sites/events/files/slides/Android-%20protecting%20the%20kernel.pdf`, 26 août 2016.
4. Jon Oberheide. Linux Kernel Exploitation : Earning Its Pwnie a Vuln at a Time. SOURCE Boston 2010, `https://jon.oberheide.org/files/source10-linuxkernel-jonoberheide.pdf`, 25 avril 2010.
5. Kees Cook. Security things in Linux v4.8. `https://outflux.net/blog/archives/2016/10/04/security-things-in-linux-v4-8/`, 4 octobre 2016.
6. Kenton Varda. Linux kernel CVE-2016-5195 "Dirty COW" mitigated by Sandstorm. `https://sandstorm.io/news/2016-10-25-cve-2016-5195-dirtycow-mitigated`, 25 octobre 2016.
7. Lennart Poettering. Avoiding CVE-2016-8655 with systemd. `http://0pointer.net/blog/avoiding-cve-2016-8655-with-systemd.html`, 07 décembre 2016.
8. Philip Pettersson. CVE-2016-8655 Linux af_packet.c race condition (local root). `http://seclists.org/oss-sec/2016/q4/607`, 6 décembre 2016.
9. Red Hat Buzilla. CVE-2016-7795 systemd : Assertion failure when PID 1 receives a zero-length message over notify socket. `https://bugzilla.redhat.com/show_bug.cgi?id=1380286`, 29 septembre 2016.