Red Hat Enterprise Linux 9

Gestion, surveillance et mise à jour du noyau

Guide de gestion du noyau Linux sur Red Hat Enterprise Linux 9

Last Updated: 2024-02-27

Red Hat Enterprise Linux 9 Gestion, surveillance et mise à jour du noyau

Guide de gestion du noyau Linux sur Red Hat Enterprise Linux 9

Notice légale

Résumé

En tant qu'administrateur système, vous pouvez configurer le noyau Linux pour optimiser le système d'exploitation. Les modifications apportées au noyau Linux peuvent améliorer les performances, la sécurité et la stabilité du système, ainsi que votre capacité à auditer le système et à résoudre les problèmes.

Table des matières

RENDRE L'OPEN SOURCE PLUS INCLUSIF

Red Hat s'engage à remplacer les termes problématiques dans son code, sa documentation et ses propriétés Web. Nous commençons par ces quatre termes : master, slave, blacklist et whitelist. En raison de l'ampleur de cette entreprise, ces changements seront mis en œuvre progressivement au cours de plusieurs versions à venir. Pour plus de détails, voir le message de notre directeur technique Chris Wright.

FOURNIR UN RETOUR D'INFORMATION SUR LA DOCUMENTATION DE RED HAT

Nous apprécions vos commentaires sur notre documentation. Faites-nous savoir comment nous pouvons l'améliorer.

Soumettre des commentaires sur des passages spécifiques

1. Consultez la documentation au format **Multi-page HTML** et assurez-vous que le bouton **Feedback** apparaît dans le coin supérieur droit après le chargement complet de la page.
2. Utilisez votre curseur pour mettre en évidence la partie du texte que vous souhaitez commenter.
3. Cliquez sur le bouton **Add Feedback** qui apparaît près du texte en surbrillance.
4. Ajoutez vos commentaires et cliquez sur **Submit**.

Soumettre des commentaires via Bugzilla (compte requis)

1. Connectez-vous au site Web de Bugzilla.
2. Sélectionnez la version correcte dans le menu **Version**.
3. Saisissez un titre descriptif dans le champ **Summary**.
4. Saisissez votre suggestion d'amélioration dans le champ **Description**. Incluez des liens vers les parties pertinentes de la documentation.
5. Cliquez sur **Submit Bug**.

CHAPITRE 1. LE NOYAU LINUX

Découvrez le noyau Linux et le paquetage RPM du noyau Linux fourni et maintenu par Red Hat (Red Hat kernel). Maintenir le noyau Red Hat à jour, ce qui garantit que le système d'exploitation dispose de toutes les dernières corrections de bogues, améliorations de performances et correctifs, et qu'il est compatible avec le nouveau matériel.

1.1. CE QU'EST LE NOYAU

Le noyau est une partie centrale du système d'exploitation Linux qui gère les ressources du système et fournit une interface entre le matériel et les applications logicielles. Le noyau Red Hat est un noyau personnalisé basé sur le noyau Linux principal en amont, que les ingénieurs de Red Hat développent et renforcent en mettant l'accent sur la stabilité et la compatibilité avec les technologies et le matériel les plus récents.

Avant que Red Hat ne publie une nouvelle version du noyau, celui-ci doit passer un ensemble de tests d'assurance qualité rigoureux.

Les noyaux Red Hat sont empaquetés au format RPM afin d'être facilement mis à niveau et vérifiés par le **dnf** gestionnaire de paquets.

AVERTISSEMENT

Les noyaux qui n'ont pas été compilés par Red Hat sont **not** pris en charge par Red Hat.

1.2. PAQUETS RPM

Un paquetage RPM est un fichier contenant d'autres fichiers et leurs métadonnées (informations sur les fichiers nécessaires au système).

Plus précisément, un paquetage RPM se compose de l'archive **cpio**.

L'archive **cpio** contient :

- Dossiers
- En-tête RPM (métadonnées du paquet)
  Le gestionnaire de paquets **rpm** utilise ces métadonnées pour déterminer les dépendances, l'endroit où installer les fichiers et d'autres informations.

Types de paquets RPM
Il existe deux types de paquets RPM. Les deux types partagent le format de fichier et l'outillage, mais ont des contenus différents et servent des objectifs différents :

- Source RPM (SRPM)
  Un SRPM contient le code source et un fichier SPEC, qui décrit comment construire le code source en un RPM binaire. En option, les correctifs du code source sont également inclus.
- RPM binaire

Un RPM binaire contient les binaires construits à partir des sources et des correctifs.

1.3. APERÇU DES PAQUETS RPM DU NOYAU LINUX

Le RPM **kernel** est un méta-paquet qui ne contient aucun fichier, mais qui s'assure que les sous-paquets suivants sont correctement installés :

- **kernel-core** - contient l'image binaire du noyau Linux (**vmlinuz**).
- **kernel-modules-core** - contient les modules de base du noyau pour assurer la fonctionnalité de base. Il s'agit des modules essentiels au bon fonctionnement du matériel le plus couramment utilisé.
- **kernel-modules** - contient les modules restants du noyau qui ne sont pas présents dans **kernel-core**.

Les sous-paquets **kernel-core** et **kernel-modules-core** peuvent être utilisés dans des environnements virtualisés et en nuage pour fournir un noyau RHEL 9 avec un temps de démarrage rapide et un faible encombrement sur le disque. Le sous-paquet **kernel-modules** n'est généralement pas nécessaire pour de tels déploiements.

Les paquets optionnels du noyau sont par exemple

- **kernel-modules-extra** - contient des modules de noyau pour du matériel rare et des modules dont le chargement est désactivé par défaut.
- **kernel-debug** - contient un noyau avec de nombreuses options de débogage activées pour le diagnostic du noyau, au détriment des performances.
- **kernel-tools** - contient des outils pour manipuler le noyau Linux et de la documentation.
- **kernel-devel** - contient les en-têtes du noyau et les fichiers makefiles nécessaires à la construction de modules à partir du paquetage **kernel**.
- **kernel-abi-stablelists** - contient des informations relatives à l'ABI du noyau RHEL, notamment une liste des symboles du noyau qui sont nécessaires aux modules externes du noyau Linux et un plug-in **dnf** pour faciliter la mise en œuvre.
- **kernel-headers** - comprend les fichiers d'en-tête C qui spécifient l'interface entre le noyau Linux et les bibliothèques et programmes de l'espace utilisateur. Les fichiers d'en-tête définissent les structures et les constantes nécessaires à la construction de la plupart des programmes standard.
- **kernel-uki-virt** - contient l'image unifiée du noyau (UKI) du noyau RHEL. L'UKI combine le noyau Linux, **initramfs**, et la ligne de commande du noyau en un seul binaire signé qui peut être démarré directement à partir du firmware UEFI. **kernel-uki-virt** contient les modules du noyau nécessaires pour fonctionner dans des environnements virtualisés et en nuage et peut être utilisé à la place du sous-paquet **kernel-core**.

IMPORTANT

kernel-uki-virt est fourni en tant qu'aperçu technologique dans RHEL 9.2.

Ressources supplémentaires

- *What are the kernel-core, kernel-modules, and kernel-modules-extras packages?*

1.4. AFFICHAGE DU CONTENU DU PAQUETAGE DU NOYAU

Affichez le contenu du paquet du noyau et de ses sous-paquets sans les installer à l'aide de la commande **rpm**.

Conditions préalables

- Obtenu les paquets RPM **kernel**, **kernel-core**, **kernel-modules**, **kernel-modules-extra** pour l'architecture de votre processeur

Procédure

- Liste des modules pour **kernel**:

```
$ rpm -qlp <kernel_rpm>
(contains no files)
…
```

- Liste des modules pour **kernel-core**:

```
$ rpm -qlp <kernel-core_rpm>
…
/lib/modules/5.14.0-1.el9.x86_64/kernel/fs/udf/udf.ko.xz
/lib/modules/5.14.0-1.el9.x86_64/kernel/fs/xfs
/lib/modules/5.14.0-1.el9.x86_64/kernel/fs/xfs/xfs.ko.xz
/lib/modules/5.14.0-1.el9.x86_64/kernel/kernel
/lib/modules/5.14.0-1.el9.x86_64/kernel/kernel/trace
/lib/modules/5.14.0-1.el9.x86_64/kernel/kernel/trace/ring_buffer_benchmark.ko.xz
/lib/modules/5.14.0-1.el9.x86_64/kernel/lib
/lib/modules/5.14.0-1.el9.x86_64/kernel/lib/cordic.ko.xz
…
```

- Liste des modules pour **kernel-modules**:

```
$ rpm -qlp <kernel-modules_rpm>
…
/lib/modules/5.14.0-1.el9.x86_64/kernel/drivers/infiniband/hw/mlx4/mlx4_ib.ko.xz
/lib/modules/5.14.0-1.el9.x86_64/kernel/drivers/infiniband/hw/mlx5/mlx5_ib.ko.xz
/lib/modules/5.14.0-1.el9.x86_64/kernel/drivers/infiniband/hw/qedr/qedr.ko.xz
/lib/modules/5.14.0-1.el9.x86_64/kernel/drivers/infiniband/hw/usnic/usnic_verbs.ko.xz
/lib/modules/5.14.0-1.el9.x86_64/kernel/drivers/infiniband/hw/vmw_pvrdma/vmw_pvrdma.ko.xz
…
```

- Liste des modules pour **kernel-modules-extra**:

```
$ rpm -qlp <kernel-modules-extra_rpm>
…
/lib/modules/5.14.0-1.el9.x86_64/extra/net/sched/sch_cbq.ko.xz
/lib/modules/5.14.0-1.el9.x86_64/extra/net/sched/sch_choke.ko.xz
/lib/modules/5.14.0-1.el9.x86_64/extra/net/sched/sch_drr.ko.xz
/lib/modules/5.14.0-1.el9.x86_64/extra/net/sched/sch_dsmark.ko.xz
/lib/modules/5.14.0-1.el9.x86_64/extra/net/sched/sch_gred.ko.xz
…
```

Ressources supplémentaires

- La page du manuel **rpm(8)**
- Paquets RPM

1.5. MISE À JOUR DU NOYAU

Mettez à jour le noyau en utilisant le gestionnaire de paquets **dnf** le gestionnaire de paquets.

Procédure

1. Pour mettre à jour le noyau, entrez la commande suivante :

   ```
   # dnf update kernel
   ```

   Cette commande met à jour le noyau ainsi que toutes les dépendances vers la dernière version disponible.

2. Redémarrez votre système pour que les modifications soient prises en compte.

Ressources supplémentaires

- gestionnaire de paquets
- La page du manuel **dnf(8)**

1.6. INSTALLATION DE VERSIONS SPÉCIFIQUES DU NOYAU

Installez de nouveaux noyaux à l'aide du **dnf** gestionnaire de paquets.

Procédure

- Pour installer une version spécifique du noyau, entrez la commande suivante :

  ```
  # dnf install kernel-{version}
  ```

Ressources supplémentaires

- *Red Hat Code Browser*
- *Red Hat Enterprise Linux Release Dates*

CHAPITRE 2. GESTION DES MODULES DU NOYAU

Découvrez les modules du noyau, comment afficher leurs informations et comment effectuer des tâches administratives de base avec les modules du noyau.

2.1. INTRODUCTION AUX MODULES DU NOYAU

Le noyau de Red Hat Enterprise Linux peut être étendu avec des éléments optionnels et supplémentaires de fonctionnalité, appelés modules de noyau, sans avoir à redémarrer le système. Sur Red Hat Enterprise Linux 9, les modules du noyau sont des codes supplémentaires du noyau qui sont intégrés dans des fichiers objets compressés **<KERNEL_MODULE_NAME>.ko.xz**.

Les fonctionnalités les plus courantes permises par les modules du noyau sont les suivantes :

- Pilote de périphérique qui ajoute la prise en charge d'un nouveau matériel
- Prise en charge d'un système de fichiers tel que **GFS2** ou **NFS**
- Appels système

Sur les systèmes modernes, les modules du noyau sont automatiquement chargés en cas de besoin. Toutefois, dans certains cas, il est nécessaire de charger ou de décharger les modules manuellement.

Comme le noyau lui-même, les modules peuvent prendre des paramètres qui personnalisent leur comportement si nécessaire.

Des outils sont fournis pour vérifier quels modules sont en cours d'exécution, quels modules sont disponibles pour être chargés dans le noyau et quels sont les paramètres acceptés par un module. L'outil fournit également un mécanisme pour charger et décharger les modules du noyau dans le noyau en cours d'exécution.

2.2. DÉPENDANCES DU MODULE DU NOYAU

Certains modules du noyau dépendent parfois d'un ou de plusieurs autres modules du noyau. Le fichier **/lib/modules/<KERNEL_VERSION>/modules.dep** contient une liste complète des dépendances des modules du noyau pour la version respective du noyau.

depmod

Le fichier de dépendances est généré par le programme **depmod**, qui fait partie du paquetage **kmod**. De nombreux utilitaires fournis par **kmod** prennent en compte les dépendances des modules lors de l'exécution des opérations, de sorte que le suivi des dépendances de **manual** est rarement nécessaire.

AVERTISSEMENT

Le code des modules du noyau est exécuté dans l'espace du noyau en mode non restreint. Pour cette raison, vous devez faire attention aux modules que vous chargez.

weak-modules

En plus de **depmod**, Red Hat Enterprise Linux fournit le script **weak-modules** livré également avec le paquetage **kmod**. **weak-modules** détermine quels modules sont compatibles kABI avec les noyaux installés. Lors de la vérification de la compatibilité des modules avec le noyau, **weak-modules** traite les dépendances des symboles des modules de la version supérieure à la version inférieure du noyau pour lequel ils ont été construits. Cela signifie que **weak-modules** traite chaque module indépendamment de la version du noyau pour lequel il a été construit.

Ressources supplémentaires

- La page du manuel **modules.dep(5)**
- La page du manuel **depmod(8)**
- Quel est l'objectif du script weak-modules fourni avec Red Hat Enterprise Linux ?
- Qu'est-ce que l'interface binaire d'application du noyau (kABI) ?

2.3. LISTE DES MODULES DU NOYAU INSTALLÉS

La commande **grubby --info=ALL** affiche une liste indexée des noyaux installés sur les installations **!BLS** et **BLS**.

Procédure

- Lister les noyaux installés à l'aide de la commande suivante :

```
# grubby --info=ALL | grep title
```

  La liste de tous les noyaux installés s'affiche comme suit :

```
title="Red Hat Enterprise Linux (5.14.0-1.el9.x86_64) 9.0 (Plow)"
title="Red Hat Enterprise Linux (0-rescue-0d772916a9724907a5d1350bcd39ac92) 9.0
(Plow)"
```

L'exemple ci-dessus affiche la liste des noyaux installés de grubby-8.40-17, à partir du menu GRUB.

2.4. LISTE DES MODULES DU NOYAU ACTUELLEMENT CHARGÉS

Affiche les modules du noyau actuellement chargés.

Conditions préalables

- Le paquet **kmod** est installé.

Procédure

- Pour obtenir la liste de tous les modules du noyau actuellement chargés, entrez :

```
$ lsmod

Module                  Size  Used by
fuse                  126976  3
uinput                 20480  1
```

```
xt_CHECKSUM           16384  1
ipt_MASQUERADE        16384  1
xt_conntrack          16384  1
ipt_REJECT            16384  1
nft_counter           16384  16
nf_nat_tftp           16384  0
nf_conntrack_tftp     16384  1 nf_nat_tftp
tun                   49152  1
bridge                192512  0
stp                   16384  1 bridge
llc                   16384  2 bridge,stp
nf_tables_set         32768  5
nft_fib_inet          16384  1
…
```

Dans l'exemple ci-dessus :

- La première colonne fournit l'adresse **names** des modules actuellement chargés.
- La deuxième colonne indique la quantité de **memory** par module en kilo-octets.
- La dernière colonne indique le nombre et, éventuellement, les noms des modules qui sont **dependent** sur un module particulier.

Ressources supplémentaires

- Le fichier **/usr/share/doc/kmod/README**
- La page du manuel **lsmod(8)**

2.5. DÉFINITION D'UN NOYAU PAR DÉFAUT

Définir un noyau spécifique par défaut à l'aide de l'outil de ligne de commande **grubby** et de GRUB.

Procédure

- Définition du noyau par défaut, à l'aide de l'outil **grubby**
  - Entrez la commande suivante pour définir le noyau par défaut à l'aide de l'outil **grubby**:

    ```
    # grubby --set-default $kernel_path
    ```

    La commande utilise comme argument un identifiant de machine sans le suffixe **.conf**.

    NOTE

    L'identifiant de la machine se trouve dans le répertoire **/boot/loader/entries/**.

- Définir le noyau par défaut, en utilisant l'argument **id**
  - Lister les entrées de démarrage à l'aide de l'argument **id**, puis définir un noyau par défaut :

    ```
    # grubby --info ALL | grep id
    # grubby --set-default /boot/vmlinuz-<version>.<architecture>
    ```

NOTE

Pour dresser la liste des entrées de démarrage à l'aide de l'argument **title**, exécutez la commande **# grubby --info=ALL | grep title** commande.

- Définition du noyau par défaut uniquement pour le prochain démarrage
  - Exécutez la commande suivante pour définir le noyau par défaut uniquement pour le prochain redémarrage à l'aide de la commande **grub2-reboot**:

```
# grub2-reboot <index|title|id>
```

AVERTISSEMENT

Définissez le noyau par défaut uniquement pour le prochain démarrage avec précaution. L'installation de nouveaux RPM de noyaux, de noyaux auto-construits et l'ajout manuel d'entrées dans le répertoire **/boot/loader/entries/** peuvent modifier les valeurs de l'index.

2.6. AFFICHAGE D'INFORMATIONS SUR LES MODULES DU NOYAU

La commande **modinfo** permet d'afficher des informations détaillées sur le module du noyau spécifié.

Conditions préalables

- Le paquet **kmod** est installé.

Procédure

- Pour afficher des informations sur n'importe quel module du noyau, entrez :

```
$ modinfo <KERNEL_MODULE_NAME>
```

Par exemple :

```
$ modinfo virtio_net

filename:       /lib/modules/5.14.0-1.el9.x86_64/kernel/drivers/net/virtio_net.ko.xz
license:        GPL
description:    Virtio network driver
rhelversion:    9.0
srcversion:     8809CDDBE7202A1B00B9F1C
alias:          virtio:d00000001v*
depends:        net_failover
retpoline:      Y
intree:         Y
name:           virtio_net
vermagic:       5.14.0-1.el9.x86_64 SMP mod_unload modversions
…
```

```
parm:           napi_weight:int
parm:           csum:bool
parm:           gso:bool
parm:           napi_tx:bool
```

Vous pouvez demander des informations sur tous les modules disponibles, qu'ils soient chargés ou non. Les entrées du site **parm** indiquent les paramètres que l'utilisateur peut définir pour le module et le type de valeur qu'il attend.

NOTE

Lorsque vous saisissez le nom d'un module du noyau, n'ajoutez pas l'extension **.ko.xz** à la fin du nom. Les noms des modules du noyau n'ont pas d'extension, mais les fichiers correspondants en ont une.

Ressources supplémentaires

- La page du manuel **modinfo(8)**

2.7. CHARGEMENT DES MODULES DU NOYAU AU MOMENT DE L'EXÉCUTION DU SYSTÈME

La meilleure façon d'étendre les fonctionnalités du noyau Linux est de charger des modules de noyau. Utilisez la commande **modprobe** pour trouver et charger un module de noyau dans le noyau en cours d'exécution.

Conditions préalables

- Autorisations de la racine
- Le paquet **kmod** est installé.
- Le module du noyau concerné n'est pas chargé. Pour s'en assurer, il convient de dresser la liste des modules du noyau chargés .

Procédure

1. Sélectionnez le module du noyau que vous souhaitez charger.
   Les modules sont situés dans le répertoire **/lib/modules/$(uname -r)/kernel/<SUBSYSTEM>/**.
2. Charger le module du noyau concerné :

   ```
   # modprobe <MODULE_NAME>
   ```

   NOTE

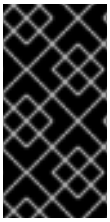
   Lorsque vous saisissez le nom d'un module du noyau, n'ajoutez pas l'extension **.ko.xz** à la fin du nom. Les noms des modules du noyau n'ont pas d'extension, mais les fichiers correspondants en ont une.

3. En option, vérifier que le module concerné a été chargé :

   ```
   $ lsmod | grep <MODULE_NAME>
   ```

Si le module a été correctement chargé, cette commande affiche le module du noyau correspondant. Par exemple :

```
$ lsmod | grep serio_raw
serio_raw              16384  0
```



IMPORTANT

Les modifications décrites dans cette procédure **will not persist** sont effectuées après le redémarrage du système. Pour plus d'informations sur le chargement des modules du noyau à l'adresse **persist** lors des redémarrages du système, voir Chargement automatique des modules du noyau au démarrage du système.

Ressources supplémentaires

- La page du manuel **modprobe(8)**

2.8. DÉCHARGEMENT DES MODULES DU NOYAU AU MOMENT DE L'EXÉCUTION DU SYSTÈME

Il arrive parfois que vous deviez décharger certains modules du noyau en cours d'exécution. Utilisez la commande **modprobe** pour trouver et décharger un module du noyau au moment de l'exécution du système à partir du noyau actuellement chargé.

Conditions préalables

- Autorisations de la racine
- Le paquet **kmod** est installé.

Procédure

1. Entrez la commande **lsmod** et sélectionnez un module du noyau que vous souhaitez décharger. Si un module du noyau a des dépendances, déchargez-les avant de décharger le module du noyau. Pour plus de détails sur l'identification des modules ayant des dépendances, voir Lister les modules du noyau actuellement chargés et Dépendances des modules du noyau .
2. Décharger le module du noyau concerné :

   ```
   # modprobe -r <MODULE_NAME>
   ```

   Lorsque vous saisissez le nom d'un module du noyau, n'ajoutez pas l'extension **.ko.xz** à la fin du nom. Les noms des modules du noyau n'ont pas d'extension, mais les fichiers correspondants en ont une.

AVERTISSEMENT

Ne pas décharger les modules du noyau lorsqu'ils sont utilisés par le système en cours d'exécution. Cela peut conduire à un système instable ou non opérationnel.

3. Optionnellement, vérifier que le module concerné a été déchargé :

```
$ lsmod | grep <MODULE_NAME>
```

Si le module a été déchargé avec succès, cette commande n'affiche aucun résultat.

IMPORTANT

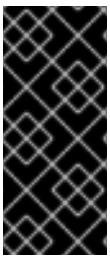
Une fois cette procédure terminée, les modules du noyau qui sont définis pour être automatiquement chargés au démarrage, **will not stay unloaded** après le redémarrage du système. Pour plus d'informations sur la manière d'éviter ce résultat, voir Empêcher le chargement automatique des modules du noyau au moment du démarrage du système.

Ressources supplémentaires

- **modprobe(8)** page du manuel

2.9. DÉCHARGEMENT DES MODULES DU NOYAU AUX PREMIERS STADES DU PROCESSUS DE DÉMARRAGE

Dans certaines situations, il est nécessaire de décharger un module du noyau très tôt dans le processus de démarrage. Par exemple, lorsque le module du noyau contient un code qui rend le système insensible et que l'utilisateur n'est pas en mesure d'atteindre le stade de désactivation permanente du module du noyau. Dans ce cas, il est possible de bloquer temporairement le chargement du module du noyau à l'aide d'un chargeur de démarrage.



IMPORTANT

Les modifications décrites dans cette procédure **will not persist** sont appliquées après le prochain redémarrage. Pour plus d'informations sur la manière d'ajouter un module de noyau à une liste de refus afin qu'il ne soit pas automatiquement chargé au cours du processus de démarrage, voir Empêcher le chargement automatique des modules de noyau au moment du démarrage du système.

Conditions préalables

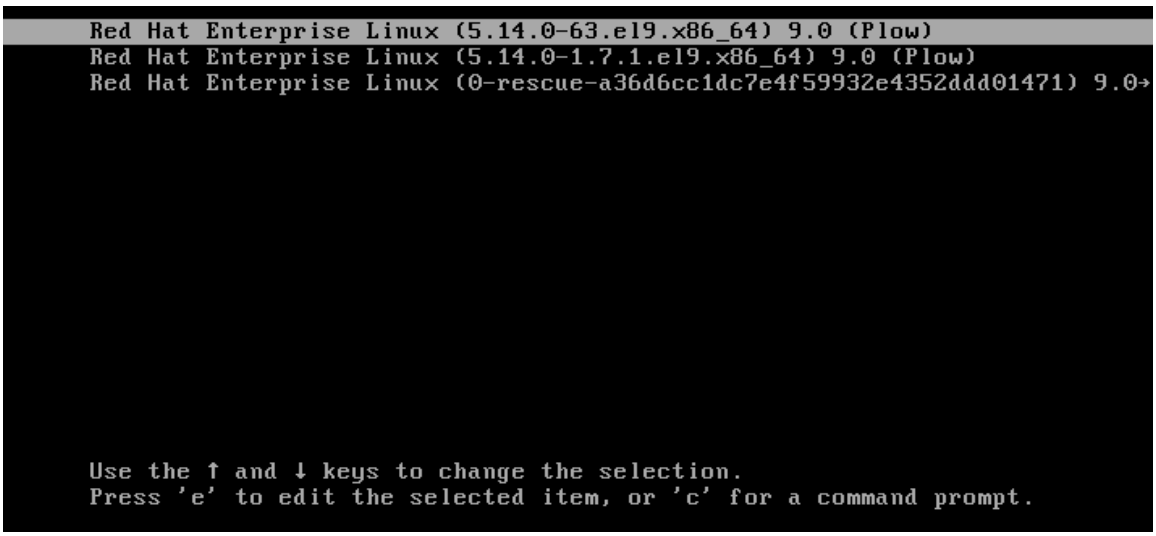
- Vous disposez d'un module de noyau chargeable, dont vous voulez empêcher le chargement pour une raison quelconque.

Procédure

- Modifiez l'entrée correspondante du chargeur de démarrage pour décharger le module de noyau souhaité avant que la séquence de démarrage ne se poursuive.

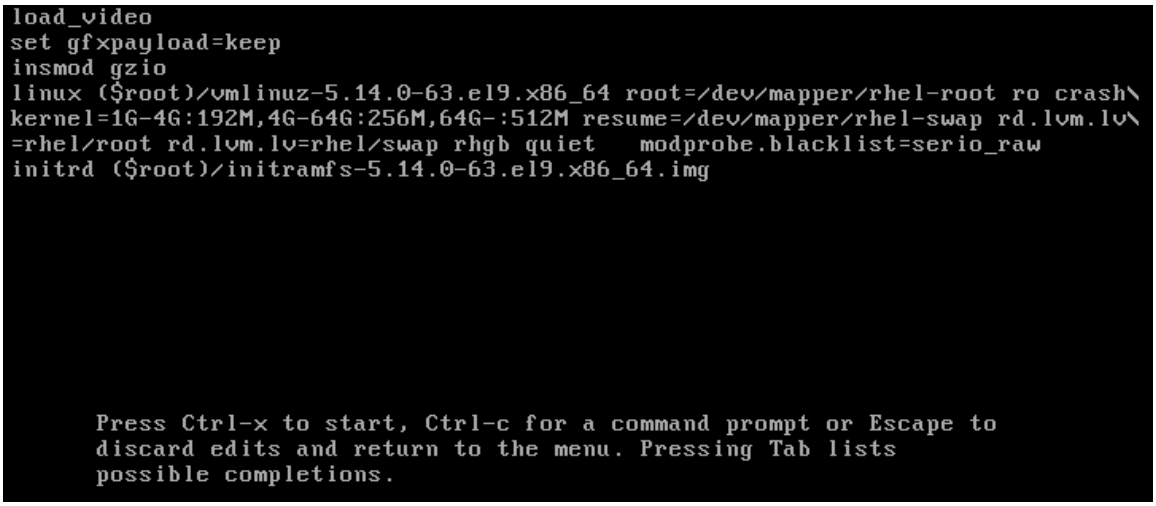
- Utilisez les touches du curseur pour mettre en évidence l'entrée du chargeur de démarrage concerné.
- Appuyez sur la touche **e** pour modifier l'entrée.

Figure 2.1. Menu de démarrage du noyau



- Utilisez les touches du curseur pour naviguer jusqu'à la ligne qui commence par **linux**.
- Ajouter **modprobe.blacklist=*module_name*** à la fin de la ligne.

Figure 2.2. Entrée de démarrage du noyau



Le module du noyau **serio_raw** illustre un module malveillant qui doit être déchargé au début du processus d'amorçage.

- Appuyez sur les touches **CTRL x** pour démarrer en utilisant la configuration modifiée.

Vérification

- Une fois que le système a complètement démarré, vérifiez que le module du noyau concerné n'est pas chargé.

```
# lsmod | grep serio_raw
```

Ressources supplémentaires

- Gestion des modules du noyau

2.10. CHARGEMENT AUTOMATIQUE DES MODULES DU NOYAU AU DÉMARRAGE DU SYSTÈME

Configurer un module du noyau pour qu'il soit chargé automatiquement pendant le processus de démarrage.

Conditions préalables

- Autorisations de la racine
- Le paquet **kmod** est installé.

Procédure

1. Sélectionnez un module du noyau que vous souhaitez charger pendant le processus de démarrage.
   Les modules sont situés dans le répertoire **/lib/modules/$(uname -r)/kernel/<SUBSYSTEM>/**.
2. Créer un fichier de configuration pour le module :

   ```
   # echo <MODULE_NAME> > /etc/modules-load.d/<MODULE_NAME>.conf
   ```

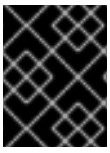
   NOTE

   Lorsque vous saisissez le nom d'un module du noyau, n'ajoutez pas l'extension **.ko.xz** à la fin du nom. Les noms des modules du noyau n'ont pas d'extension, mais les fichiers correspondants en ont une.

3. En option, après le redémarrage, vérifiez que le module concerné a été chargé :

   ```
   $ lsmod | grep <MODULE_NAME>
   ```

   L'exemple de commande ci-dessus devrait réussir et afficher le module du noyau concerné.



IMPORTANT

Les changements décrits dans cette procédure **will persist** après le redémarrage du système.

Ressources supplémentaires

- **modules-load.d(5)** page du manuel

2.11. EMPÊCHER LE CHARGEMENT AUTOMATIQUE DES MODULES DU NOYAU AU DÉMARRAGE DU SYSTÈME

Ajouter un module du noyau à une liste de refus afin qu'il ne soit pas automatiquement chargé pendant le processus de démarrage.

Conditions préalables

- Autorisations de la racine
- Le paquet **kmod** est installé.
- Assurez-vous qu'un module du noyau figurant dans une liste de refus n'est pas vital pour la configuration actuelle de votre système.

Procédure

1. Sélectionnez un module du noyau que vous souhaitez placer dans une liste de refus :

```
$ lsmod

Module                  Size  Used by
fuse                  126976  3
xt_CHECKSUM            16384  1
ipt_MASQUERADE         16384  1
uinput                 20480  1
xt_conntrack           16384  1
…
```

La commande **lsmod** affiche une liste des modules chargés dans le noyau en cours d'exécution.

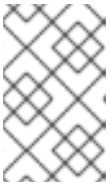
- Vous pouvez également identifier un module du noyau non chargé que vous souhaitez empêcher de se charger.
  Tous les modules du noyau sont situés dans le répertoire **/lib/modules/<KERNEL_VERSION>/kernel/<SUBSYSTEM>/**.

2. Créer un fichier de configuration pour une liste de deniers :

```
# vim /etc/modprobe.d/blacklist.conf

# Blacklists <KERNEL_MODULE_1>
blacklist <MODULE_NAME_1>
install <MODULE_NAME_1> /bin/false

# Blacklists <KERNEL_MODULE_2>
blacklist <MODULE_NAME_2>
install <MODULE_NAME_2> /bin/false

# Blacklists <KERNEL_MODULE_n>
blacklist <MODULE_NAME_n>
install <MODULE_NAME_n> /bin/false
…
```

L'exemple montre le contenu du fichier **blacklist.conf**, édité par l'éditeur **vim**. La ligne **blacklist** garantit que le module du noyau concerné ne sera pas automatiquement chargé pendant le processus de démarrage. La commande **blacklist**, cependant, n'empêche pas le module d'être chargé en tant que dépendance d'un autre module du noyau qui n'est pas dans une liste de dénylage. Par conséquent, la ligne **install** entraîne l'exécution de la commande **/bin/false** au lieu de l'installation d'un module.

Les lignes commençant par un signe dièse sont des commentaires destinés à rendre le fichier plus lisible.



NOTE

Lorsque vous saisissez le nom d'un module du noyau, n'ajoutez pas l'extension **.ko.xz** à la fin du nom. Les noms des modules du noyau n'ont pas d'extension, mais les fichiers correspondants en ont une.

3. Créez une copie de sauvegarde de l'image ramdisk initiale actuelle avant de la reconstruire :

```
# cp /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r).bak.$(date %m-\r%H%M%S).img
```

La commande ci-dessus crée une image de sauvegarde **initramfs** au cas où la nouvelle version aurait un problème inattendu.

- Vous pouvez également créer une copie de sauvegarde d'une autre image initiale de ramdisk qui correspond à la version du noyau pour laquelle vous souhaitez placer des modules de noyau dans une liste de dénombrement :

```
# cp /boot/initramfs-<SOME_VERSION>.img /boot/initramfs-<SOME_VERSION>.img.bak.$(date %m-\r%H%M%S)
```

4. Générer une nouvelle image ramdisk initiale pour refléter les changements :

```
# dracut -f -v
```

- Si vous construisez une image initiale de ramdisk pour une version de noyau différente de celle dans laquelle vous êtes actuellement démarré, indiquez à la fois la cible **initramfs** et la version du noyau :

```
# dracut -f -v /boot/initramfs-<TARGET_VERSION>.img <CORRESPONDING_TARGET_KERNEL_VERSION>
```

5. Redémarrer le système :

```
$ reboot
```

IMPORTANT

Les changements décrits dans cette procédure **will take effect and persist**après le redémarrage du système. Si vous placez incorrectement un module clé du noyau dans une liste de refus, vous risquez d'avoir un système instable ou non opérationnel.

Ressources supplémentaires

- Comment empêcher le chargement automatique d'un module du noyau ?
- **dracut(8)** page du manuel

2.12. COMPILATION DE MODULES DE NOYAU PERSONNALISÉS

Vous pouvez créer un module de noyau d'échantillonnage en fonction des différentes configurations matérielles et logicielles.

Conditions préalables

- Vous avez installé les paquets **kernel-devel**, **gcc** et **elfutils-libelf-devel**.

```
# dnf install kernel-devel-$(uname -r) gcc elfutils-libelf-devel
```

- Vous disposez des droits d'accès à la racine.
- Vous avez créé le répertoire **/root/testmodule/** dans lequel vous compilez le module de noyau personnalisé.

Procédure

1. Créez le fichier **/root/testmodule/test.c** avec le contenu suivant.

```
#include <linux/module.h>
#include <linux/kernel.h>

int init_module(void)
  { printk("Hello World\n This is a test\n"); return 0; }

void cleanup_module(void)
  { printk("Good Bye World"); }

MODULE_LICENSE("GPL");
```

Le fichier **test.c** est un fichier source qui fournit les principales fonctionnalités du module du noyau. Le fichier a été créé dans un répertoire **/root/testmodule/** dédié à des fins d'organisation. Après la compilation du module, le répertoire **/root/testmodule/** contiendra plusieurs fichiers.

Le fichier **test.c** inclut les bibliothèques du système :

- Le fichier d'en-tête **linux/kernel.h** est nécessaire pour la fonction **printk()** dans le code d'exemple.
- Le fichier **linux/module.h** contient des déclarations de fonctions et des définitions de macros à partager entre plusieurs fichiers sources écrits en langage de programmation C. Suivez ensuite les fonctions **init_module()** et **cleanup_module()** pour démarrer et terminer la fonction d'enregistrement du noyau **printk()**, qui imprime du texte.

2. Créez le fichier **/root/testmodule/Makefile** avec le contenu suivant.

```
obj-m := test.o
```

Le Makefile contient des instructions selon lesquelles le compilateur doit produire un fichier objet spécifiquement nommé **test.o**. La directive **obj-m** spécifie que le fichier **test.ko** résultant sera compilé en tant que module de noyau chargeable. Alternativement, la directive **obj-y** indiquerait de compiler **test.ko** en tant que module de noyau intégré.

3. Compiler le module du noyau.

```
# make -C /lib/modules/$(uname -r)/build M=/root/testmodule modules
make: Entering directory '/usr/src/kernels/5.14.0-70.17.1.el9_0.x86_64'
  CC [M]  /root/testmodule/test.o
  MODPOST /root/testmodule/Module.symvers
```

```
  CC [M]  /root/testmodule/test.mod.o
  LD [M]  /root/testmodule/test.ko
  BTF [M] /root/testmodule/test.ko
Skipping BTF generation for /root/testmodule/test.ko due to unavailability of vmlinux
make: Leaving directory '/usr/src/kernels/5.14.0-70.17.1.el9_0.x86_64'
```

Le compilateur crée un fichier objet (**test.o**) pour chaque fichier source (**test.c**) en tant qu'étape intermédiaire avant de les lier ensemble dans le module final du noyau (**test.ko**).

Après une compilation réussie, **/root/testmodule/** contient des fichiers supplémentaires relatifs au module de noyau personnalisé compilé. Le module compilé lui-même est représenté par le fichier **test.ko**.

Vérification

1. Facultatif : vérifiez le contenu du répertoire **/root/testmodule/**:

```
# ls -l /root/testmodule/
total 152
-rw-r—r--. 1 root root    16 Jul 26 08:19 Makefile
-rw-r—r--. 1 root root    25 Jul 26 08:20 modules.order
-rw-r—r--. 1 root root     0 Jul 26 08:20 Module.symvers
-rw-r—r--. 1 root root   224 Jul 26 08:18 test.c
-rw-r—r--. 1 root root 62176 Jul 26 08:20 test.ko
-rw-r—r--. 1 root root    25 Jul 26 08:20 test.mod
-rw-r—r--. 1 root root   849 Jul 26 08:20 test.mod.c
-rw-r—r--. 1 root root 50936 Jul 26 08:20 test.mod.o
-rw-r—r--. 1 root root 12912 Jul 26 08:20 test.o
```

2. Copiez le module du noyau dans le répertoire **/lib/modules/$(uname -r)/**:

```
# cp /root/testmodule/test.ko /lib/modules/$(uname -r)/
```

3. Mise à jour de la liste des dépendances modulaires :

```
# depmod -a
```

4. Charger le module du noyau :

```
# modprobe -v test
insmod /lib/modules/5.14.0-1.el9.x86_64/test.ko
```

5. Vérifiez que le module du noyau a été chargé avec succès :

```
# lsmod | grep test
test                   16384  0
```

6. Lire les derniers messages du tampon circulaire du noyau :

```
# dmesg
[74422.545004] Hello World
               This is a test
```

Ressources supplémentaires

- Gestion des modules du noyau

CHAPITRE 3. SIGNER UN NOYAU ET DES MODULES POUR LE DÉMARRAGE SÉCURISÉ

Vous pouvez renforcer la sécurité de votre système en utilisant un noyau et des modules de noyau signés. Sur les systèmes de construction basés sur l'UEFI où Secure Boot est activé, vous pouvez signer vous-même un noyau ou des modules de noyau construits de manière privée. En outre, vous pouvez importer votre clé publique dans un système cible où vous souhaitez déployer votre noyau ou vos modules de noyau.

Si l'option Secure Boot est activée, tous les composants suivants doivent être signés à l'aide d'une clé privée et authentifiés à l'aide de la clé publique correspondante :

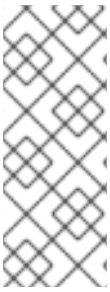
- Chargeur de démarrage du système d'exploitation UEFI
- Le noyau Red Hat Enterprise Linux
- Tous les modules du noyau

Si l'un de ces composants n'est pas signé et authentifié, le système ne peut pas terminer le processus de démarrage.

Red Hat Enterprise Linux 9 comprend :

- Chargeurs de démarrage signés
- Noyaux signés
- Modules signés du noyau

De plus, le chargeur de démarrage de première étape signé et le noyau signé incluent des clés publiques Red Hat intégrées. Ces binaires exécutables signés et ces clés intégrées permettent à Red Hat Enterprise Linux 9 de s'installer, de démarrer et de s'exécuter avec les clés de l'autorité de certification Microsoft UEFI Secure Boot qui sont fournies par le microprogramme UEFI sur les systèmes qui prennent en charge UEFI Secure Boot.



NOTE

- Tous les systèmes basés sur l'UEFI ne prennent pas en charge le démarrage sécurisé.
- Le système de construction, dans lequel vous construisez et signez votre module de noyau, n'a pas besoin d'activer le Secure Boot UEFI ni même d'être un système basé sur l'UEFI.

3.1. CONDITIONS PRÉALABLES

- Pour pouvoir signer les modules du noyau construits en externe, installez les utilitaires des paquets suivants :

```
# dnf install pesign openssl kernel-devel mokutil keyutils
```

Tableau 3.1. Services publics requis

Utilité	Fourni par paquet	Utilisé le	Objectif
efikeygen	**pesign**	Système de construction	Génère des paires de clés X.509 publiques et privées
openssl	**openssl**	Système de construction	Exporte la clé privée non chiffrée
sign-file	**kernel-devel**	Système de construction	Fichier exécutable utilisé pour signer un module du noyau avec la clé privée
mokutil	**mokutil**	Système cible	Utilitaire facultatif utilisé pour enregistrer manuellement la clé publique
keyctl	**keyutils**	Système cible	Utilitaire facultatif utilisé pour afficher les clés publiques dans le trousseau de clés du système

3.2. COMPRENDRE LE DÉMARRAGE SÉCURISÉ DE L'UEFI

La technologie Secure Boot de *Unified Extensible Firmware Interface* (UEFI) permet d'empêcher l'exécution du code de l'espace noyau qui n'a pas été signé par une clé de confiance. Le chargeur de démarrage du système est signé par une clé cryptographique. La base de données des clés publiques, qui est contenue dans le microprogramme, autorise la clé de signature. Vous pouvez ensuite vérifier une signature dans le chargeur de démarrage de l'étape suivante et dans le noyau.

L'UEFI Secure Boot établit une chaîne de confiance entre le microprogramme et les pilotes et modules du noyau signés, comme suit :

- Une clé privée UEFI signe et une clé publique authentifie le chargeur de démarrage de première étape **shim**. Un *certificate authority* (CA) signe à son tour la clé publique. L'autorité de certification est stockée dans la base de données du microprogramme.
- Le fichier **shim** contient la clé publique de Red Hat **Red Hat Secure Boot (CA key 1)**pour authentifier le chargeur de démarrage GRUB et le noyau.
- Le noyau contient quant à lui des clés publiques pour authentifier les pilotes et les modules.

Secure Boot est le composant de validation du chemin de démarrage de la spécification UEFI. La spécification définit :

- Interface de programmation pour les variables UEFI protégées cryptographiquement dans une mémoire non volatile.
- Stockage des certificats racine X.509 de confiance dans les variables UEFI.
- Validation des applications UEFI telles que les chargeurs de démarrage et les pilotes.
- Procédures de révocation des certificats et des hachages d'applications connus comme mauvais.

L'UEFI Secure Boot permet de détecter les modifications non autorisées, mais ne permet pas d'accéder au site **not**:

- Empêcher l'installation ou la suppression des chargeurs de démarrage de deuxième niveau.
- Exiger une confirmation explicite de ces modifications par l'utilisateur.
- Arrêter les manipulations du chemin d'amorçage. Les signatures sont vérifiées lors du démarrage, et non lors de l'installation ou de la mise à jour du chargeur de démarrage.

Si le chargeur de démarrage ou le noyau ne sont pas signés par une clé de confiance du système, Secure Boot les empêche de démarrer.

3.3. PRISE EN CHARGE DE L'UEFI SECURE BOOT

Vous pouvez installer et exécuter Red Hat Enterprise Linux 9 sur des systèmes avec UEFI Secure Boot activé si le noyau et tous les pilotes chargés sont signés avec une clé de confiance. Red Hat fournit des noyaux et des pilotes qui sont signés et authentifiés par les clés Red Hat appropriées.

Si vous souhaitez charger des noyaux ou des pilotes construits en externe, vous devez également les signer.

Restrictions imposées par UEFI Secure Boot

- Le système n'exécute le code en mode noyau que lorsque sa signature a été correctement authentifiée.
- Le chargement du module GRUB est désactivé parce qu'il n'existe pas d'infrastructure pour la signature et la vérification des modules GRUB. Leur chargement constitue l'exécution d'un code non fiable à l'intérieur du périmètre de sécurité défini par Secure Boot.
- Red Hat fournit un binaire GRUB signé qui contient tous les modules pris en charge sur Red Hat Enterprise Linux 9.

Ressources supplémentaires

- Restrictions imposées par UEFI Secure Boot

3.4. EXIGENCES POUR L'AUTHENTIFICATION DES MODULES DU NOYAU AVEC DES CLÉS X.509

Dans Red Hat Enterprise Linux 9, lorsqu'un module du noyau est chargé, le noyau vérifie la signature du module par rapport aux clés publiques X.509 du trousseau de clés du système du noyau (**.builtin_trusted_keys**) et du trousseau de clés de la plate-forme du noyau (**.platform**). Le trousseau **.platform** contient des clés provenant de fournisseurs de plateformes tiers et des clés publiques personnalisées. Les clés du trousseau du système du noyau **.blacklist** sont exclues de la vérification.

Certaines conditions doivent être remplies pour charger les modules du noyau sur les systèmes dont la fonctionnalité UEFI Secure Boot est activée :

- Si l'UEFI Secure Boot est activé ou si le paramètre **module.sig_enforce** kernel a été spécifié :
  - Vous ne pouvez charger que les modules signés du noyau dont les signatures ont été authentifiées à l'aide des clés du trousseau de clés du système (**.builtin_trusted_keys**) et du trousseau de clés de la plate-forme (**.platform**).

    - La clé publique ne doit pas figurer dans le trousseau des clés révoquées du système (**.blacklist**).

- Si UEFI Secure Boot est désactivé et que le paramètre **module.sig_enforce** kernel n'a pas été spécifié :
    - Vous pouvez charger des modules de noyau non signés et des modules de noyau signés sans clé publique.
- Si le système n'est pas basé sur l'UEFI ou si le Secure Boot de l'UEFI est désactivé :
    - Seules les clés intégrées dans le noyau sont chargées sur **.builtin_trusted_keys** et **.platform**.
    - Vous n'avez pas la possibilité d'augmenter ce jeu de clés sans reconstruire le noyau.

Tableau 3.2. Exigences d'authentification du module du noyau pour le chargement

Module signé	Clé publique trouvée et signature valide	État de l'UEFI Secure Boot	sig_enforce	Chargement du module	Noyau altéré
Non signé	-	Non activé	Non activé	Réussite	Oui
		Non activé	Activé	Échecs	-
		Activé	-	Échecs	-
Signé	Non	Non activé	Non activé	Réussite	Oui
		Non activé	Activé	Échecs	-
		Activé	-	Échecs	-
Signé	Oui	Non activé	Non activé	Réussite	Non
		Non activé	Activé	Réussite	Non
		Activé	-	Réussite	Non

3.5. SOURCES DES CLÉS PUBLIQUES

Lors du démarrage, le noyau charge les clés X.509 à partir d'un ensemble de réserves de clés persistantes dans les trousseaux de clés suivants :

- Le porte-clés du système (**.builtin_trusted_keys**)
- Le porte-clés **.platform**
- Le système **.blacklist** porte-clés

Tableau 3.3. Sources pour les porte-clés système

Source des clés X.509	L'utilisateur peut ajouter des clés	État de l'UEFI Secure Boot	Clés chargées au démarrage
Intégrée dans le noyau	Non	-	**.builtin_trusted_keys**
UEFI **db**	Limitée	Non activé	Non
		Activé	**.platform**
Intégrée dans le chargeur de démarrage **shim**	Non	Non activé	Non
		Activé	**.platform**
Liste des clés du propriétaire de la machine (MOK)	Oui	Non activé	Non
		Activé	**.platform**

.builtin_trusted_keys

- Un porte-clés construit sur mesure
- Contient des clés publiques de confiance
- **root** des privilèges sont nécessaires pour visualiser les clés

.platform

- Un porte-clés construit sur mesure
- Contient des clés provenant de fournisseurs de plateformes tiers et des clés publiques personnalisées
- **root** des privilèges sont nécessaires pour visualiser les clés

.blacklist

- Un trousseau de clés X.509 révoquées
- Un module signé par une clé provenant de **.blacklist** échouera à l'authentification même si votre clé publique se trouve dans **.builtin_trusted_keys**

UEFI Secure Boot**db**

- Une base de données de signatures
- Stocke les clés (hachages) des applications UEFI, des pilotes UEFI et des chargeurs de démarrage
- Les clés peuvent être chargées sur la machine

UEFI Secure Boot**dbx**

- Une base de données de signatures révoquées
- Empêche le chargement des clés
- Les clés révoquées de cette base de données sont ajoutées au trousseau de clés **.blacklist**

3.6. GÉNÉRER UNE PAIRE DE CLÉS PUBLIQUE ET PRIVÉE

Pour utiliser un noyau personnalisé ou des modules de noyau personnalisés sur un système compatible avec Secure Boot, vous devez générer une paire de clés X.509 publique et privée. Vous pouvez utiliser la clé privée générée pour signer le noyau ou les modules du noyau. Vous pouvez également valider le noyau ou les modules signés en ajoutant la clé publique correspondante à la clé du propriétaire de la machine (MOK) pour Secure Boot.

AVERTISSEMENT

Appliquez des mesures de sécurité et des politiques d'accès strictes pour protéger le contenu de votre clé privée. Entre de mauvaises mains, la clé pourrait être utilisée pour compromettre tout système authentifié par la clé publique correspondante.

Procédure

- Créez une paire de clés publiques et privées X.509 :
  - Si vous souhaitez uniquement signer un noyau personnalisé *modules*:

```
# efikeygen --dbdir /etc/pki/pesign \
        --self-sign \
        --module \
        --common-name 'CN=Organization signing key' \
        --nickname 'Custom Secure Boot key'
```

  - Si vous souhaitez signer un document personnalisé *kernel*:

```
# efikeygen --dbdir /etc/pki/pesign \
        --self-sign \
        --kernel \
        --common-name 'CN=Organization signing key' \
        --nickname 'Custom Secure Boot key'
```

  - Lorsque le système RHEL fonctionne en mode FIPS :

```
# efikeygen --dbdir /etc/pki/pesign \
        --self-sign \
        --kernel \
        --common-name 'CN=Organization signing key' \
        --nickname 'Custom Secure Boot key'
        --token 'pkcs11:token=NSS%20FIPS%20140-2%20Certificate%20DB'
```

Par défaut, lorsque l'option **--token** n'est pas spécifiée, **pkcs11:token=NSS Certificate DB** est utilisé.

Les clés publiques et privées sont maintenant stockées dans le répertoire **/etc/pki/pesign/**.

IMPORTANT

C'est une bonne pratique de sécurité de signer le noyau et les modules du noyau pendant la période de validité de sa clé de signature. Cependant, l'utilitaire **sign-file** ne vous avertit pas et la clé sera utilisable dans Red Hat Enterprise Linux 9 indépendamment des dates de validité.

Ressources supplémentaires

- **openssl(1)** page du manuel
- *RHEL Security Guide*
- Enregistrement de la clé publique sur le système cible en ajoutant la clé publique à la liste MOK

3.7. EXEMPLE DE SORTIE DES TROUSSEAUX DE CLÉS DU SYSTÈME

Vous pouvez afficher des informations sur les clés des trousseaux de clés du système à l'aide de l'utilitaire **keyctl** du paquetage **keyutils**.

Conditions préalables

- Vous disposez des droits d'accès à la racine.
- Vous avez installé l'utilitaire **keyctl** à partir du paquetage **keyutils**.

Exemple 3.1. Sortie de porte-clés

Voici un exemple abrégé de la sortie des trousseaux de clés **.builtin_trusted_keys**, **.platform**, et **.blacklist** d'un système Red Hat Enterprise Linux 9 où l'UEFI Secure Boot est activé.

```
# keyctl list %:.builtin_trusted_keys
6 keys in keyring:
...asymmetric: Red Hat Enterprise Linux Driver Update Program (key 3): bf57f3e87...
...asymmetric: Red Hat Secure Boot (CA key 1): 4016841644ce3a810408050766e8f8a29...
...asymmetric: Microsoft Corporation UEFI CA 2011: 13adbf4309bd82709c8cd54f316ed...
...asymmetric: Microsoft Windows Production PCA 2011: a92902398e16c49778cd90f99e...
...asymmetric: Red Hat Enterprise Linux kernel signing key: 4249689eefc77e95880b...
...asymmetric: Red Hat Enterprise Linux kpatch signing key: 4d38fd864ebe18c5f0b7...

# keyctl list %:.platform
4 keys in keyring:
...asymmetric: VMware, Inc.: 4ad8da0472073...
...asymmetric: Red Hat Secure Boot CA 5: cc6fafe72...
...asymmetric: Microsoft Windows Production PCA 2011: a929f298e1...
...asymmetric: Microsoft Corporation UEFI CA 2011: 13adbf4e0bd82...

# keyctl list %:.blacklist
4 keys in keyring:
...blacklist: bin:f5ff83a...
```

```
...blacklist: bin:0dfdbec...
...blacklist: bin:38f1d22...
...blacklist: bin:51f831f...
```

Le trousseau **.builtin_trusted_keys** de l'exemple montre l'ajout de deux clés provenant des clés UEFI Secure Boot **db** ainsi que de la clé **Red Hat Secure Boot (CA key 1)**, qui est intégrée dans le chargeur de démarrage **shim**.

Exemple 3.2. Sortie de la console du noyau

L'exemple suivant montre la sortie de la console du noyau. Les messages identifient les clés avec une source liée à l'UEFI Secure Boot. Il s'agit de UEFI Secure Boot **db**, embedded **shim**, et MOK list.

```
# dmesg | egrep 'integrity.*cert'
[1.512966] integrity: Loading X.509 certificate: UEFI:db
[1.513027] integrity: Loaded X.509 cert 'Microsoft Windows Production PCA 2011: a929023...
[1.513028] integrity: Loading X.509 certificate: UEFI:db
[1.513057] integrity: Loaded X.509 cert 'Microsoft Corporation UEFI CA 2011: 13adbf4309...
[1.513298] integrity: Loading X.509 certificate: UEFI:MokListRT (MOKvar table)
[1.513549] integrity: Loaded X.509 cert 'Red Hat Secure Boot CA 5: cc6fa5e72868ba494e93...
```

Ressources supplémentaires

- **keyctl(1)**, **dmesg(1)** pages de manuel

3.8. ENREGISTREMENT DE LA CLÉ PUBLIQUE SUR LE SYSTÈME CIBLE EN AJOUTANT LA CLÉ PUBLIQUE À LA LISTE MOK

Vous devez enregistrer votre clé publique sur tous les systèmes où vous souhaitez authentifier et charger votre noyau ou vos modules de noyau. Vous pouvez importer la clé publique sur un système cible de différentes manières afin que le trousseau de clés de la plate-forme (**.platform**) puisse utiliser la clé publique pour authentifier le noyau ou les modules du noyau.

Lorsque RHEL 9 démarre sur un système basé sur l'UEFI avec Secure Boot activé, le noyau charge sur le trousseau de la plate-forme (**.platform**) toutes les clés publiques qui se trouvent dans la base de données de clés Secure Boot **db**. En même temps, le noyau exclut les clés de la base de données **dbx** des clés révoquées.

Vous pouvez utiliser la fonction de facilité de la clé du propriétaire de la machine (MOK) pour étendre la base de données de clés de l'UEFI Secure Boot. Lorsque RHEL 9 démarre sur un système UEFI avec Secure Boot activé, les clés de la liste MOK sont également ajoutées au trousseau de clés de la plate-forme (**.platform**) en plus des clés de la base de données de clés. Les clés de la liste MOK sont également stockées de manière persistante et sécurisée de la même manière que les clés de la base de données Secure Boot, mais il s'agit de deux installations distinctes. La fonction MOK est prise en charge par **shim**, **MokManager**, **GRUB** et l'utilitaire **mokutil**.

NOTE

Pour faciliter l'authentification de votre module de noyau sur vos systèmes, envisagez de demander à votre fournisseur de système d'incorporer votre clé publique dans la base de données de clés UEFI Secure Boot dans leur image de micrologiciel d'usine.

Conditions préalables

- Vous avez généré une paire de clés publique et privée et vous connaissez les dates de validité de vos clés publiques. Pour plus d'informations, voir Générer une paire de clés publique et privée.

Procédure

1. Exportez votre clé publique vers le fichier **sb_cert.cer**:

```
# certutil -d /etc/pki/pesign \
        -n 'Custom Secure Boot key' \
        -Lr \
        > sb_cert.cer
```

2. Importez votre clé publique dans la liste MOK :

```
# mokutil --import sb_cert.cer
```

3. Saisissez un nouveau mot de passe pour cette demande d'inscription au MOK.

4. Redémarrer la machine.
   Le chargeur de démarrage **shim** remarque la demande d'enrôlement de la clé MOK en attente et lance **MokManager.efi** pour vous permettre de terminer l'enrôlement à partir de la console UEFI.

5. Choisissez **Enroll MOK**, saisissez le mot de passe que vous avez précédemment associé à cette demande lorsque vous y êtes invité et confirmez l'inscription.
   Votre clé publique est ajoutée à la liste MOK, qui est persistante.

   Une fois qu'une clé figure sur la liste MOK, elle est automatiquement propagée dans le trousseau **.platform** lors de ce démarrage et des suivants lorsque l'UEFI Secure Boot est activé.

3.9. SIGNER UN NOYAU AVEC LA CLÉ PRIVÉE

Vous pouvez améliorer la sécurité de votre système en chargeant un noyau signé si le mécanisme de démarrage sécurisé de l'UEFI est activé.

Conditions préalables

- Vous avez généré une paire de clés publique et privée et vous connaissez les dates de validité de vos clés publiques. Pour plus d'informations, voir Générer une paire de clés publique et privée.
- Vous avez enregistré votre clé publique sur le système cible. Pour plus de détails, voir Enrôler une clé publique sur le système cible en ajoutant la clé publique à la liste MOK.
- Vous disposez d'une image de noyau au format ELF disponible pour la signature.

Procédure

- Sur l'architecture x64 :

  a. Créer une image signée :

```
# pesign --certificate 'Custom Secure Boot key' \
        --in vmlinuz-version \
        --sign \
        --out vmlinuz-version.signed
```

Remplacez ***version*** par le suffixe de version de votre fichier **vmlinuz**, et ***Custom Secure Boot key*** par le nom que vous avez choisi précédemment.

b. Facultatif : Vérifier les signatures :

```
# pesign --show-signature \
        --in vmlinuz-version.signed
```

c. Remplacer l'image non signée par l'image signée :

```
# mv vmlinuz-version.signed vmlinuz-version
```

- Sur l'architecture ARM 64 bits :

a. Décompressez le fichier **vmlinuz**:

```
# zcat vmlinuz-version > vmlinux-version
```

b. Créer une image signée :

```
# pesign --certificate 'Custom Secure Boot key' \
        --in vmlinux-version \
        --sign \
        --out vmlinux-version.signed
```

c. Facultatif : Vérifier les signatures :

```
# pesign --show-signature \
        --in vmlinux-version.signed
```

d. Compresser le fichier **vmlinux**:

```
# gzip --to-stdout vmlinux-version.signed > vmlinuz-version
```

e. Supprimer le fichier non compressé **vmlinux**:

```
# rm vmlinux-version*
```

3.10. SIGNER UN BUILD GRUB AVEC LA CLÉ PRIVÉE

Sur un système où le mécanisme de démarrage sécurisé UEFI est activé, vous pouvez signer une version GRUB avec une clé privée personnalisée existante. Vous devez procéder ainsi si vous utilisez une version personnalisée de GRUB ou si vous avez supprimé l'ancre de confiance Microsoft de votre système.

Conditions préalables

- Vous avez généré une paire de clés publique et privée et vous connaissez les dates de validité de vos clés publiques. Pour plus d'informations, voir Générer une paire de clés publique et privée.
- Vous avez enregistré votre clé publique sur le système cible. Pour plus de détails, voir Enrôler une clé publique sur le système cible en ajoutant la clé publique à la liste MOK.
- Vous disposez d'un binaire GRUB EFI à signer.

Procédure

- Sur l'architecture x64 :
  a. Créer un binaire EFI GRUB signé :

```
# pesign --in /boot/efi/EFI/redhat/grubx64.efi \
        --out /boot/efi/EFI/redhat/grubx64.efi.signed \
        --certificate 'Custom Secure Boot key' \
        --sign
```

  Remplacez ***Custom Secure Boot key*** par le nom que vous avez choisi précédemment.

  b. Facultatif : Vérifier les signatures :

```
# pesign --in /boot/efi/EFI/redhat/grubx64.efi.signed \
        --show-signature
```

  c. Remplacer le binaire non signé par le binaire signé :

```
# mv /boot/efi/EFI/redhat/grubx64.efi.signed \
    /boot/efi/EFI/redhat/grubx64.efi
```

- Sur l'architecture ARM 64 bits :
  a. Créer un binaire EFI GRUB signé :

```
# pesign --in /boot/efi/EFI/redhat/grubaa64.efi \
        --out /boot/efi/EFI/redhat/grubaa64.efi.signed \
        --certificate 'Custom Secure Boot key' \
        --sign
```

  Remplacez ***Custom Secure Boot key*** par le nom que vous avez choisi précédemment.

  b. Facultatif : Vérifier les signatures :

```
# pesign --in /boot/efi/EFI/redhat/grubaa64.efi.signed \
        --show-signature
```

  c. Remplacer le binaire non signé par le binaire signé :

```
# mv /boot/efi/EFI/redhat/grubaa64.efi.signed \
    /boot/efi/EFI/redhat/grubaa64.efi
```

3.11. SIGNER LES MODULES DU NOYAU AVEC LA CLÉ PRIVÉE

Vous pouvez renforcer la sécurité de votre système en chargeant des modules de noyau signés si le mécanisme de démarrage sécurisé de l'UEFI est activé.

Votre module noyau signé est également chargeable sur les systèmes où le Secure Boot UEFI est désactivé ou sur un système non UEFI. Par conséquent, vous n'avez pas besoin de fournir une version signée et une version non signée de votre module noyau.

Conditions préalables

- Vous avez généré une paire de clés publique et privée et vous connaissez les dates de validité de vos clés publiques. Pour plus d'informations, voir Générer une paire de clés publique et privée.
- Vous avez enregistré votre clé publique sur le système cible. Pour plus de détails, voir Enrôler une clé publique sur le système cible en ajoutant la clé publique à la liste MOK.
- Vous disposez d'un module de noyau au format d'image ELF disponible pour la signature.

Procédure

1. Exportez votre clé publique vers le fichier **sb_cert.cer**:

```
# certutil -d /etc/pki/pesign \
        -n 'Custom Secure Boot key' \
        -Lr \
        > sb_cert.cer
```

2. Extraire la clé de la base de données du SSN sous la forme d'un fichier PKCS #12 :

```
# pk12util -o sb_cert.p12 \
        -n 'Custom Secure Boot key' \
        -d /etc/pki/pesign
```

3. Lorsque la commande précédente vous y invite, entrez un nouveau mot de passe qui crypte la clé privée.

4. Exporter la clé privée non chiffrée :

```
# openssl pkcs12 \
      -in sb_cert.p12 \
      -out sb_cert.priv \
      -nocerts \
      -noenc
```

IMPORTANT

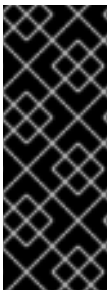
Manipulez la clé privée non chiffrée avec précaution.

5. Signez votre module de noyau. La commande suivante ajoute la signature directement à l'image ELF dans le fichier du module du noyau :

```
# /usr/src/kernels/$(uname -r)/scripts/sign-file \
        sha256 \
        sb_cert.priv \
```

```
        sb_cert.cer \
        my_module.ko
```

Votre module noyau est maintenant prêt à être chargé.



IMPORTANT

Dans Red Hat Enterprise Linux 9, les dates de validité de la paire de clés sont importantes. La clé n'expire pas, mais le module du noyau doit être signé pendant la période de validité de sa clé de signature. L'utilitaire **sign-file** ne vous en avertira pas. Par exemple, une clé qui n'est valide qu'en 2021 peut être utilisée pour authentifier un module du noyau signé en 2021 avec cette clé. Cependant, les utilisateurs ne peuvent pas utiliser cette clé pour signer un module de noyau en 2022.

Vérification

1. Affiche des informations sur la signature du module du noyau :

```
# modinfo my_module.ko | grep signer
signer:         Your Name Key
```

   Vérifiez que la signature mentionne votre nom tel qu'il a été saisi lors de la génération.

NOTE

La signature annexée n'est pas contenue dans une section de l'image ELF et ne constitue pas une partie formelle de l'image ELF. Par conséquent, des utilitaires tels que **readelf** ne peuvent pas afficher la signature sur votre module de noyau.

2. Charger le module :

```
# insmod my_module.ko
```

3. Retirer (décharger) le module :

```
# rmmod my_module.ko
```

Ressources supplémentaires

- Affichage d'informations sur les modules du noyau

3.12. CHARGEMENT DES MODULES SIGNÉS DU NOYAU

Une fois que votre clé publique est enregistrée dans le trousseau de clés du système (**.builtin_trusted_keys**) et dans la liste MOK, et après avoir signé le module du noyau concerné avec votre clé privée, vous pouvez charger le module du noyau signé à l'aide de la commande **modprobe**.

Conditions préalables

- Vous avez généré la paire de clés publique et privée. Pour plus d'informations, voir Générer une paire de clés publique et privée.

- Vous avez enregistré la clé publique dans le trousseau de clés du système. Pour plus de détails, voir Enrôler une clé publique sur le système cible en ajoutant la clé publique à la liste MOK .
- Vous avez signé un module du noyau avec la clé privée. Pour plus de détails, voir Signer des modules du noyau avec la clé privée.
- Installez le paquet **kernel-modules-extra**, qui crée le répertoire **/lib/modules/$(uname -r)/extra/**:

```
# dnf -y install kernel-modules-extra
```

Procédure

1. Vérifiez que vos clés publiques se trouvent sur le trousseau de clés du système :

```
# keyctl list %:.platform
```

2. Copiez le module du noyau dans le répertoire **extra/** du noyau que vous voulez :

```
# cp my_module.ko /lib/modules/$(uname -r)/extra/
```

3. Mise à jour de la liste des dépendances modulaires :

```
# depmod -a
```

4. Charger le module du noyau :

```
# modprobe -v my_module
```

5. En option, pour charger le module au démarrage, ajoutez-le au fichier **/etc/modules-loaded.d/*my_module*.conf** pour charger le module au démarrage :

```
# echo "my_module" > /etc/modules-load.d/my_module.conf
```

Vérification

- Vérifiez que le module a été chargé avec succès :

```
# lsmod | grep my_module
```

Ressources supplémentaires

- Gestion des modules du noyau

CHAPITRE 4. MISE À JOUR DE LA LISTE DE RÉVOCATION DE L'AMORÇAGE SÉCURISÉ

Vous pouvez mettre à jour la liste de révocation UEFI Secure Boot de votre système afin que Secure Boot identifie les logiciels présentant des problèmes de sécurité connus et les empêche de compromettre votre processus de démarrage.

4.1. CONDITIONS PRÉALABLES

- Secure Boot est activé sur votre système.

4.2. COMPRENDRE LE DÉMARRAGE SÉCURISÉ DE L'UEFI

La technologie Secure Boot de *Unified Extensible Firmware Interface* (UEFI) permet d'empêcher l'exécution du code de l'espace noyau qui n'a pas été signé par une clé de confiance. Le chargeur de démarrage du système est signé par une clé cryptographique. La base de données des clés publiques, qui est contenue dans le microprogramme, autorise la clé de signature. Vous pouvez ensuite vérifier une signature dans le chargeur de démarrage de l'étape suivante et dans le noyau.

L'UEFI Secure Boot établit une chaîne de confiance entre le microprogramme et les pilotes et modules du noyau signés, comme suit :

- Une clé privée UEFI signe et une clé publique authentifie le chargeur de démarrage de première étape **shim**. Un *certificate authority* (CA) signe à son tour la clé publique. L'autorité de certification est stockée dans la base de données du microprogramme.
- Le fichier **shim** contient la clé publique de Red Hat **Red Hat Secure Boot (CA key 1)** pour authentifier le chargeur de démarrage GRUB et le noyau.
- Le noyau contient quant à lui des clés publiques pour authentifier les pilotes et les modules.

Secure Boot est le composant de validation du chemin de démarrage de la spécification UEFI. La spécification définit :

- Interface de programmation pour les variables UEFI protégées cryptographiquement dans une mémoire non volatile.
- Stockage des certificats racine X.509 de confiance dans les variables UEFI.
- Validation des applications UEFI telles que les chargeurs de démarrage et les pilotes.
- Procédures de révocation des certificats et des hachages d'applications connus comme mauvais.

L'UEFI Secure Boot permet de détecter les modifications non autorisées, mais ne permet pas d'accéder au site **not**:

- Empêcher l'installation ou la suppression des chargeurs de démarrage de deuxième niveau.
- Exiger une confirmation explicite de ces modifications par l'utilisateur.
- Arrêter les manipulations du chemin d'amorçage. Les signatures sont vérifiées lors du démarrage, et non lors de l'installation ou de la mise à jour du chargeur de démarrage.

Si le chargeur de démarrage ou le noyau ne sont pas signés par une clé de confiance du système, Secure Boot les empêche de démarrer.

4.3. LA LISTE DE RÉVOCATION DE SECURE BOOT

La Secure Boot Revocation List de l'UEFI, ou Secure Boot Forbidden Signature Database (**dbx**), est une liste qui identifie les logiciels dont l'exécution n'est plus autorisée par Secure Boot.

Lorsqu'un problème de sécurité ou de stabilité est détecté dans un logiciel interfacé avec Secure Boot, tel que le chargeur de démarrage GRUB, la liste de révocation enregistre sa signature de hachage. Les logiciels dont la signature est reconnue ne peuvent pas être exécutés pendant le démarrage, et le démarrage du système échoue afin d'éviter de compromettre le système.

Par exemple, une certaine version de GRUB peut contenir un problème de sécurité qui permet à un attaquant de contourner le mécanisme de démarrage sécurisé. Lorsque le problème est détecté, la liste de révocation ajoute les signatures de hachage de toutes les versions de GRUB qui contiennent le problème. Par conséquent, seules les versions sécurisées de GRUB peuvent démarrer sur le système.

La liste des révocations nécessite des mises à jour régulières afin de prendre en compte les problèmes nouvellement découverts. Lors de la mise à jour de la liste de révocation, veillez à utiliser une méthode de mise à jour sûre qui n'entraîne pas l'arrêt du démarrage de votre système actuellement installé.

4.4. APPLICATION D'UNE MISE À JOUR EN LIGNE DE LA LISTE DE RÉVOCATION

Vous pouvez mettre à jour la liste de révocation de l'amorçage sécurisé de votre système afin que l'amorçage sécurisé empêche les problèmes de sécurité connus. Cette procédure est sûre et garantit que la mise à jour n'empêche pas le démarrage de votre système.

Conditions préalables

- Votre système peut accéder à l'internet pour les mises à jour.

Procédure

1. Déterminer la version actuelle de la liste de révocation :

```
# fwupdmgr get-devices
```

   Voir le champ **Current version** sous **UEFI dbx**.

2. Activer le référentiel de listes de révocation LVFS :

```
# fwupdmgr enable-remote lvfs
```

3. Actualiser les métadonnées du référentiel :

```
# fwupdmgr refresh
```

4. Appliquer la mise à jour de la liste de révocation :

   - Sur la ligne de commande :

```
# fwupdmgr update
```

- Dans l'interface graphique :
  i. Ouvrir l'application **Software**
  ii. Naviguez jusqu'à l'onglet **Updates**.
  iii. Trouvez l'entrée **Secure Boot dbx Configuration Update**
  iv. Cliquez sur **Mettre à jour**.

5. À la fin de la mise à jour, **fwupdmgr** ou **Software** vous demande de redémarrer le système. Confirmez le redémarrage.

Vérification

- Après le redémarrage, vérifiez à nouveau la version actuelle de la liste de révocation :

```
# fwupdmgr get-devices
```

4.5. APPLICATION D'UNE MISE À JOUR DE LA LISTE DE RÉVOCATION HORS LIGNE

Sur un système sans connexion internet, vous pouvez mettre à jour la Secure Boot Revocation List à partir de RHEL afin que Secure Boot prévienne les problèmes de sécurité connus. Cette procédure est sûre et garantit que la mise à jour n'empêche pas votre système de démarrer.

Procédure

1. Déterminer la version actuelle de la liste de révocation :

```
# fwupdmgr get-devices
```

   Voir le champ **Current version** sous **UEFI dbx**.

2. Liste des mises à jour disponibles auprès de RHEL :

```
# ls /usr/share/dbxtool/
```

3. Sélectionnez le fichier de mise à jour le plus récent pour votre architecture. Les noms de fichiers utilisent le format suivant :

```
DBXUpdate-date-architecture.cab
```

4. Installer le fichier de mise à jour sélectionné :

```
# fwupdmgr install /usr/share/dbxtool/DBXUpdate-date-architecture.cab
```

5. À la fin de la mise à jour, **fwupdmgr** vous demande de redémarrer le système. Confirmez le redémarrage.

Vérification

- Après le redémarrage, vérifiez à nouveau la version actuelle de la liste de révocation :

```
# fwupdmgr get-devices
```

CHAPITRE 5. CONFIGURATION DES PARAMÈTRES DE LA LIGNE DE COMMANDE DU NOYAU

Les paramètres de la ligne de commande du noyau vous permettent de modifier le comportement de certains aspects du noyau de Red Hat Enterprise Linux au moment du démarrage. En tant qu'administrateur système, vous avez un contrôle total sur les options qui sont définies au démarrage. Certains comportements du noyau ne peuvent être définis qu'au moment du démarrage, c'est pourquoi la compréhension de la manière d'effectuer ces changements est une compétence clé en matière d'administration.

IMPORTANT

Changer le comportement du système en modifiant les paramètres de la ligne de commande du noyau peut avoir des effets négatifs sur votre système. Testez toujours les modifications avant de les déployer dans la production. Pour plus d'informations, contactez l'assistance Red Hat.

5.1. COMPRENDRE LES PARAMÈTRES DE LA LIGNE DE COMMANDE DU NOYAU

Les paramètres de la ligne de commande du noyau permettent d'écraser les valeurs par défaut et de définir des paramètres matériels spécifiques. Au moment du démarrage, vous pouvez configurer les fonctions suivantes :

- Le noyau Red Hat Enterprise Linux
- Le disque RAM initial
- Les caractéristiques de l'espace utilisateur

Par défaut, les paramètres de ligne de commande du noyau pour les systèmes utilisant le chargeur d'amorçage GRUB sont définis dans le fichier de configuration de l'entrée d'amorçage pour chaque entrée d'amorçage du noyau.

Vous pouvez manipuler les fichiers de configuration du chargeur de démarrage à l'aide de l'utilitaire **grubby**. L'utilitaire **grubby** vous permet d'effectuer les opérations suivantes :

- Modifier l'entrée de démarrage par défaut.
- Ajouter ou supprimer des arguments à une entrée du menu GRUB.

Ressources supplémentaires

- **kernel-command-line(7)**, **bootparam(7)** et **dracut.cmdline(7)** pages de manuel
- Comment installer et démarrer des noyaux personnalisés dans Red Hat Enterprise Linux 8
- La page du manuel **grubby(8)**

5.2. COMPRENDRE LES ENTRÉES DE DÉMARRAGE

Une entrée de démarrage est un ensemble d'options stockées dans un fichier de configuration et liées à une version particulière du noyau. En pratique, vous avez au moins autant d'entrées de démarrage que de noyaux installés sur votre système. Le fichier de configuration de l'entrée de démarrage est situé

dans le répertoire **/boot/loader/entries/** et peut ressembler à ceci :

```
d8712ab6d4f14683c5625e87b52b6b6e-5.14.0-1.el9.x86_64.conf
```

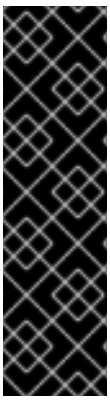
Le nom du fichier ci-dessus se compose d'un identifiant de machine stocké dans le fichier **/etc/machine-id** et d'une version du noyau.

Le fichier de configuration de l'entrée de démarrage contient des informations sur la version du noyau, l'image initiale du disque dur et les paramètres de la ligne de commande du noyau. L'exemple du contenu d'un fichier de configuration d'entrée de démarrage est présenté ci-dessous :

```
title Red Hat Enterprise Linux (5.14.0-1.el9.x86_64) 9.0 (Plow)
version 5.14.0-1.el9.x86_64
linux /vmlinuz-5.14.0-1.el9.x86_64
initrd /initramfs-5.14.0-1.el9.x86_64.img
options root=/dev/mapper/rhel_kvm--02--guest08-root ro crashkernel=1G-4G:192M,4G-64G:256M,64G-:512M resume=/dev/mapper/rhel_kvm--02--guest08-swap rd.lvm.lv=rhel_kvm-02-guest08/root rd.lvm.lv=rhel_kvm-02-guest08/swap console=ttyS0,115200
grub_users $grub_users
grub_arg --unrestricted
grub_class kernel
```

5.3. MODIFICATION DES PARAMÈTRES DE LA LIGNE DE COMMANDE DU NOYAU POUR TOUTES LES ENTRÉES DE DÉMARRAGE

Modifier les paramètres de la ligne de commande du noyau pour toutes les entrées de démarrage de votre système.



IMPORTANT

Lors de l'installation d'une nouvelle version du noyau dans les systèmes RHEL 9, l'outil **grubby** transmet les arguments de la ligne de commande du noyau de la version précédente.

Toutefois, cela ne s'applique pas à la version 9.0 de RHEL, dans laquelle les noyaux nouvellement installés perdent les options de ligne de commande précédentes. Vous devez exécuter la commande **grub2-mkconfig** sur le noyau nouvellement installé pour transmettre les paramètres à votre nouveau noyau. Pour plus d'informations sur ce problème connu, voir Chargeur de démarrage .

Conditions préalables

- Vérifiez que l'utilitaire **grubby** est installé sur votre système.
- Vérifiez que l'utilitaire **zipl** est installé sur votre système IBM Z.

Procédure

- Pour ajouter un paramètre :

```
# grubby --update-kernel=ALL --args="<NEW_PARAMETER>"
```

Pour les systèmes qui utilisent le chargeur d'amorçage GRUB et, sur IBM Z, qui utilisent le chargeur d'amorçage zIPL, la commande ajoute un nouveau paramètre de noyau à chaque fichier **/boot/loader/entries/*<ENTRY>*.conf** fichier.

  - Sur IBM Z, exécutez la commande **zipl** sans options pour mettre à jour le menu de démarrage.

- Pour supprimer un paramètre :

```
# grubby --update-kernel=ALL --remove-args="<PARAMETER_TO_REMOVE>"
```

  - Sur IBM Z, exécutez la commande **zipl** sans options pour mettre à jour le menu de démarrage.

- Après chaque mise à jour de votre paquetage de noyau, propagez les options de noyau configurées vers les nouveaux noyaux :

```
# grub2-mkconfig -o /etc/grub2.cfg
```

Ressources supplémentaires

- Comprendre les paramètres de la ligne de commande du noyau
- **grubby(8)** et **zipl(8)** pages de manuel
- *grubby tool*

5.4. MODIFICATION DES PARAMÈTRES DE LA LIGNE DE COMMANDE DU NOYAU POUR UNE SEULE ENTRÉE DE DÉMARRAGE

Modifier les paramètres de la ligne de commande du noyau pour une seule entrée de démarrage sur votre système.

Conditions préalables

- Vérifiez que les utilitaires **grubby** et **zipl** sont installés sur votre système.

Procédure

- Pour ajouter un paramètre :

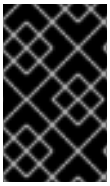
```
# grubby --update-kernel=/boot/vmlinuz-$(uname -r) --args="<NEW_PARAMETER>"
```

  - Sur IBM Z, exécutez la commande **zipl** sans options pour mettre à jour le menu de démarrage.

- Pour supprimer un paramètre, procédez comme suit :

```
# grubby --update-kernel=/boot/vmlinuz-$(uname -r) --remove-args="
<PARAMETER_TO_REMOVE>"
```

  - Sur IBM Z, exécutez la commande **zipl** sans options pour mettre à jour le menu de démarrage.



IMPORTANT

- **grubby** modifie et stocke les paramètres de la ligne de commande du noyau d'une entrée de démarrage individuelle du noyau dans le fichier **/boot/loader/entries/*<ENTRY>*.conf** dans le fichier

Ressources supplémentaires

- Comprendre les paramètres de la ligne de commande du noyau
- **grubby(8)** et **zipl(8)** pages de manuel
- *grubby tool*

5.5. MODIFICATION TEMPORAIRE DES PARAMÈTRES DE LA LIGNE DE COMMANDE DU NOYAU AU MOMENT DU DÉMARRAGE

Apporter des modifications temporaires à une entrée de menu du noyau en modifiant les paramètres du noyau uniquement au cours d'un processus de démarrage unique.

Procédure

1. Sélectionnez le noyau que vous souhaitez lancer lorsque le menu de démarrage de GRUB 2 s'affiche et appuyez sur la touche **e** pour modifier les paramètres du noyau.
2. Trouvez la ligne de commande du noyau en déplaçant le curseur vers le bas. La ligne de commande du noyau commence par **linux** sur les systèmes IBM Power Series 64 bits et x86-64 basés sur le BIOS, ou par **linuxefi** sur les systèmes UEFI.
3. Déplacer le curseur à la fin de la ligne.

NOTE

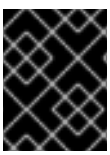
Appuyer sur **Ctrl**+**a** pour sauter au début de la ligne et **Ctrl**+**e** pour passer à la fin de la ligne. Sur certains systèmes, les touches **Home** et **End** peuvent également fonctionner.

4. Modifiez les paramètres du noyau en fonction des besoins. Par exemple, pour faire fonctionner le système en mode d'urgence, ajoutez le paramètre *emergency* à la fin de la ligne **linux**:

```
linux ($root)/vmlinuz-5.14.0-63.el9.x86_64 root=/dev/mapper/rhel-root ro crashkernel=1G-4G:192M,4G-64G:256M,64G-:512M resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb quiet pass:quotes[_emergency_]
```

Pour activer les messages du système, supprimez les paramètres **rhgb** et **quiet**.

1. Appuyer sur **Ctrl**+**x** pour démarrer avec le noyau sélectionné et les paramètres de ligne de commande modifiés.



IMPORTANT

Appuyez sur la touche **Esc** pour quitter l'édition de la ligne de commande et toutes les modifications effectuées par l'utilisateur seront supprimées.

NOTE

Cette procédure ne s'applique qu'à un seul démarrage et n'apporte pas de modifications permanentes.

5.6. CONFIGURATION DES PARAMÈTRES GRUB POUR PERMETTRE LA CONNEXION À UNE CONSOLE SÉRIE

La console série est utile lorsque vous devez vous connecter à un serveur sans tête ou à un système embarqué et que le réseau est en panne. Ou lorsque vous devez contourner les règles de sécurité et obtenir un accès de connexion sur un autre système.

Vous devez configurer certains paramètres par défaut de GRUB pour utiliser la connexion de console série.

Conditions préalables

- Vous disposez des droits d'accès à la racine.

Procédure

1. Ajoutez les deux lignes suivantes au fichier **/etc/default/grub**:

```
GRUB_TERMINAL="serial"
GRUB_SERIAL_COMMAND="serial --speed=9600 --unit=0 --word=8 --parity=no --stop=1"
```

   La première ligne désactive le terminal graphique. La clé **GRUB_TERMINAL** remplace les valeurs des clés **GRUB_TERMINAL_INPUT** et **GRUB_TERMINAL_OUTPUT**.

   La deuxième ligne ajuste le débit en bauds (**--speed**), la parité et d'autres valeurs en fonction de votre environnement et de votre matériel. Notez qu'une vitesse de transmission beaucoup plus élevée, par exemple 115200, est préférable pour des tâches telles que le suivi des fichiers journaux.

2. Mettre à jour le fichier de configuration GRUB.

   - Sur les machines basées sur le BIOS :

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

   - Sur les machines basées sur l'UEFI :

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

3. Redémarrez le système pour que les modifications soient prises en compte.

CHAPITRE 6. CONFIGURATION DES PARAMÈTRES DU NOYAU AU MOMENT DE L'EXÉCUTION

En tant qu'administrateur système, vous pouvez modifier de nombreuses facettes du comportement du noyau de Red Hat Enterprise Linux lors de l'exécution. Configurez les paramètres du noyau au moment de l'exécution en utilisant la commande **sysctl** et en modifiant les fichiers de configuration dans les répertoires **/etc/sysctl.d/** et **/proc/sys/**.

6.1. QU'EST-CE QUE LES PARAMÈTRES DU NOYAU ?

Les paramètres du noyau sont des valeurs réglables que vous pouvez ajuster pendant que le système fonctionne. Il n'est pas nécessaire de redémarrer ou de recompiler le noyau pour que les modifications prennent effet.

Il est possible d'adresser les paramètres du noyau par l'intermédiaire de :

- La commande **sysctl**
- Le système de fichiers virtuel monté dans le répertoire **/proc/sys/**
- Les fichiers de configuration dans le répertoire **/etc/sysctl.d/**

Les paramètres sont divisés en classes par le sous-système du noyau. Red Hat Enterprise Linux possède les classes de paramètres suivantes :

Tableau 6.1. Tableau des classes sysctl

Classe accordable	Sous-système
abi	Domaines d'exécution et personnalités
crypto	Interfaces cryptographiques
débogage	Interfaces de débogage du noyau
dev	Informations spécifiques à l'appareil
fs	Paramètres globaux et spécifiques du système de fichiers
noyau	Paramètres globaux du noyau
net	Paramètres de réseau
sunrpc	Appel de procédure à distance Sun (NFS)
utilisateur	Limites de l'espace de nommage de l'utilisateur
vm	Optimisation et gestion de la mémoire, des tampons et du cache

IMPORTANT

La configuration des paramètres du noyau sur un système de production nécessite une planification minutieuse. Des modifications non planifiées peuvent rendre le noyau instable et nécessiter un redémarrage du système. Vérifiez que vous utilisez des options valides avant de modifier les valeurs du noyau.

Ressources supplémentaires

- **sysctl(8)**et **sysctl.d(5)** pages de manuel

6.2. CONFIGURER TEMPORAIREMENT LES PARAMÈTRES DU NOYAU AVEC SYSCTL

La commande **sysctl** permet de définir temporairement les paramètres du noyau au moment de l'exécution. Cette commande est également utile pour lister et filtrer les paramètres.

Conditions préalables

- Autorisations de la racine

Procédure

1. Liste de tous les paramètres et de leurs valeurs.

```
# sysctl -a
```

NOTE

La commande **# sysctl -a** affiche les paramètres du noyau, qui peuvent être ajustés en cours d'exécution et au démarrage.

2. Pour configurer temporairement un paramètre, entrez :

```
# sysctl <TUNABLE_CLASS>.<PARAMETER>=<TARGET_VALUE>
```

L'exemple de commande ci-dessus modifie la valeur du paramètre alors que le système est en cours d'exécution. Les modifications prennent effet immédiatement, sans qu'il soit nécessaire de redémarrer le système.

NOTE

Les modifications reviennent aux valeurs par défaut après le redémarrage du système.

Ressources supplémentaires

- La page du manuel **sysctl(8)**
- Configuration permanente des paramètres du noyau avec sysctl
- Utilisation des fichiers de configuration dans /etc/sysctl.d/ pour ajuster les paramètres du noyau

6.3. CONFIGURATION PERMANENTE DES PARAMÈTRES DU NOYAU AVEC SYSCTL

Utilisez la commande **sysctl** pour définir de manière permanente les paramètres du noyau.

Conditions préalables

- Autorisations de la racine

Procédure

1. Liste de tous les paramètres.

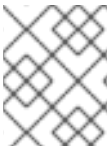
```
# sysctl -a
```

La commande affiche tous les paramètres du noyau qui peuvent être configurés au moment de l'exécution.

2. Configurer un paramètre de manière permanente :

```
# sysctl -w <TUNABLE_CLASS>.<PARAMETER>=<TARGET_VALUE> >> /etc/sysctl.conf
```

L'exemple de commande modifie la valeur ajustable et l'écrit dans le fichier **/etc/sysctl.conf**, ce qui remplace les valeurs par défaut des paramètres du noyau. Les modifications prennent effet immédiatement et de manière persistante, sans qu'il soit nécessaire de redémarrer le système.



NOTE

Pour modifier de manière permanente les paramètres du noyau, vous pouvez également modifier manuellement les fichiers de configuration dans le répertoire **/etc/sysctl.d/**.

Ressources supplémentaires

- Les pages du manuel **sysctl(8)** et **sysctl.conf(5)**
- Utilisation des fichiers de configuration dans /etc/sysctl.d/ pour ajuster les paramètres du noyau

6.4. UTILISATION DES FICHIERS DE CONFIGURATION DANS /ETC/SYSCTL.D/ POUR AJUSTER LES PARAMÈTRES DU NOYAU

Modifiez manuellement les fichiers de configuration dans le répertoire **/etc/sysctl.d/** pour définir de manière permanente les paramètres du noyau.

Conditions préalables

- Autorisations de la racine

Procédure

1. Créez un nouveau fichier de configuration à l'adresse **/etc/sysctl.d/**.

```
# vim /etc/sysctl.d/<some_file.conf>
```

2. Inclure les paramètres du noyau, un par ligne.

```
<TUNABLE_CLASS>.<PARAMETER>=<TARGET_VALUE>
<TUNABLE_CLASS>.<PARAMETER>=<TARGET_VALUE>
```

3. Sauvegarder le fichier de configuration.

4. Redémarrez la machine pour que les modifications soient prises en compte.

   - Sinon, pour appliquer les changements sans redémarrer, entrez :

     ```
     # sysctl -p /etc/sysctl.d/<some_file.conf>
     ```

     Cette commande vous permet de lire les valeurs du fichier de configuration que vous avez créé précédemment.

Ressources supplémentaires

- **sysctl(8)**, **sysctl.d(5)** pages de manuel

6.5. CONFIGURATION TEMPORAIRE DES PARAMÈTRES DU NOYAU VIA /PROC/SYS/

Définir temporairement les paramètres du noyau par le biais des fichiers du répertoire du système de fichiers virtuels /**proc**/**sys**/.

Conditions préalables

- Autorisations de la racine

Procédure

1. Identifiez un paramètre du noyau que vous souhaitez configurer.

   ```
   # ls -l /proc/sys/<TUNABLE_CLASS>/
   ```

   Les fichiers accessibles en écriture renvoyés par la commande peuvent être utilisés pour configurer le noyau. Les fichiers en lecture seule fournissent des informations sur les paramètres actuels.

2. Attribuer une valeur cible au paramètre du noyau.

   ```
   # echo <TARGET_VALUE> > /proc/sys/<TUNABLE_CLASS>/<PARAMETER>
   ```

   La commande apporte des modifications à la configuration qui disparaîtront après le redémarrage du système.

3. En option, vérifiez la valeur du nouveau paramètre du noyau.

   ```
   # cat /proc/sys/<TUNABLE_CLASS>/<PARAMETER>
   ```

Ressources supplémentaires

- Configuration permanente des paramètres du noyau avec sysctl
- Utilisation des fichiers de configuration dans /etc/sysctl.d/ pour ajuster les paramètres du noyau

CHAPITRE 7. MAINTENIR LES PARAMÈTRES DE PANIQUE DU NOYAU DÉSACTIVÉS DANS LES ENVIRONNEMENTS VIRTUALISÉS

Lors de la configuration d'un environnement virtualisé dans RHEL 9, vous ne devez pas activer les paramètres du noyau **softlockup_panic** et **nmi_watchdog**, car l'environnement virtualisé peut déclencher un verrouillage logiciel intempestif qui ne devrait pas nécessiter une panique du système.

Les sections suivantes expliquent les raisons de ce conseil.

7.1. QU'EST-CE QU'UN VERROUILLAGE SOUPLE ?

Un soft lockup est une situation généralement causée par un bogue, lorsqu'une tâche s'exécute dans l'espace du noyau sur une unité centrale sans être reprogrammée. La tâche ne permet pas non plus à une autre tâche de s'exécuter sur cette unité centrale particulière. En conséquence, un avertissement est affiché à l'utilisateur par le biais de la console système. Ce problème est également connu sous le nom de "soft lockup firing".

Ressources supplémentaires

- *What is a CPU soft lockup?*

7.2. PARAMÈTRES CONTRÔLANT LA PANIQUE DU NOYAU

Les paramètres suivants du noyau peuvent être définis pour contrôler le comportement d'un système lorsqu'un verrouillage progressif est détecté.

softlockup_panic

Contrôle si le noyau panique ou non lorsqu'un blocage progressif est détecté.

Type	Valeur	Effect
Integer	0	le noyau ne panique pas lors d'un verrouillage progressif
Integer	1	le noyau panique lors d'un verrouillage progressif

Par défaut, sur RHEL8, cette valeur est 0.

Pour pouvoir paniquer, le système doit d'abord détecter un blocage dur. La détection est contrôlée par le paramètre **nmi_watchdog**.

nmi_watchdog

Contrôle si les mécanismes de détection de verrouillage (**watchdogs**) sont actifs ou non. Ce paramètre est de type entier.

Valeur	Effect
0	désactive le détecteur de verrouillage
1	permet d'activer le détecteur de verrouillage

Le détecteur de blocage dur surveille la capacité de chaque unité centrale à répondre aux interruptions.

watchdog_thresh

Contrôle la fréquence du chien de garde **hrtimer**, des événements NMI et des seuils de verrouillage doux/dur.

Seuil par défaut	Seuil de verrouillage souple
10 secondes	2 * **watchdog_thresh**

La mise à zéro de ce paramètre désactive complètement la détection de verrouillage.

Ressources supplémentaires

- *Softlockup detector and hardlockup detector*
- *Kernel sysctl*

7.3. BLOCAGES INTEMPESTIFS DANS LES ENVIRONNEMENTS VIRTUALISÉS

Le verrouillage progressif qui se déclenche sur les hôtes physiques, comme décrit dans Qu'est-ce qu'un verrouillage progressif, représente généralement un bogue du noyau ou du matériel. Le même phénomène se produisant sur des systèmes d'exploitation invités dans des environnements virtualisés peut représenter un faux avertissement.

Une charge de travail importante sur un hôte ou une forte contention sur une ressource spécifique telle que la mémoire, provoque généralement un déclenchement intempestif du verrouillage progressif. En effet, l'hôte peut planifier l'arrêt de l'unité centrale invitée pendant une période supérieure à 20 secondes. Ensuite, lorsque l'unité centrale invitée est à nouveau programmée pour fonctionner sur l'hôte, elle subit un *time jump* qui déclenche les temporisateurs nécessaires. Les temporisateurs comprennent également le chien de garde **hrtimer**, qui peut par conséquent signaler un verrouillage progressif de l'unité centrale invitée.

Étant donné qu'un verrouillage progressif dans un environnement virtualisé peut être erroné, vous ne devez pas activer les paramètres du noyau qui provoqueraient une panique du système lorsqu'un verrouillage progressif est signalé sur un processeur invité.

IMPORTANT

Pour comprendre les blocages logiciels dans les invités, il est essentiel de savoir que l'hôte planifie l'invité en tant que tâche, et que l'invité planifie ensuite ses propres tâches.

Ressources supplémentaires

- Qu'est-ce qu'un verrouillage souple ?
- *Virtual machine components and their interaction*
- *Virtual machine reports a "BUG: soft lockup"*

CHAPITRE 8. AJUSTEMENT DES PARAMÈTRES DU NOYAU POUR LES SERVEURS DE BASE DE DONNÉES

Il existe différents ensembles de paramètres du noyau qui peuvent affecter les performances d'applications de bases de données spécifiques. Pour garantir un fonctionnement efficace des serveurs de bases de données et des bases de données, il convient de configurer les paramètres du noyau en conséquence.

8.1. INTRODUCTION TO DATABASE SERVERS

A database server is a service that provides features of a database management system (DBMS). DBMS provides utilities for database administration and interacts with end users, applications, and databases.

Red Hat Enterprise Linux 9 provides the following database management systems:

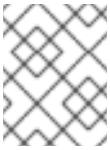
- MariaDB 10.5
- MySQL 8.0
- PostgreSQL 13
- Redis 6

8.2. PARAMÈTRES AFFECTANT LA PERFORMANCE DES APPLICATIONS DE BASE DE DONNÉES

Les paramètres suivants du noyau affectent les performances des applications de base de données.

fs.aio-max-nr

Définit le nombre maximum d'opérations d'E/S asynchrones que le système peut gérer sur le serveur.



NOTE

L'augmentation du paramètre **fs.aio-max-nr** n'entraîne aucune autre modification que l'augmentation de la limite aio.

fs.file-max

Définit le nombre maximum de file handles (noms de fichiers temporaires ou ID attribués aux fichiers ouverts) que le système peut prendre en charge à tout moment.
Le noyau alloue dynamiquement des gestionnaires de fichiers chaque fois qu'un gestionnaire de fichier est demandé par une application. Cependant, le noyau ne libère pas ces poignées de fichiers lorsqu'elles sont libérées par l'application. Il les recycle à la place. Cela signifie qu'au fil du temps, le nombre total de gestionnaires de fichiers alloués augmentera même si le nombre de gestionnaires de fichiers actuellement utilisés est faible.

kernel.shmall

Définit le nombre total de pages de mémoire partagée pouvant être utilisées à l'échelle du système. Pour utiliser la totalité de la mémoire principale, la valeur du paramètre **kernel.shmall** doit être ≤ taille totale de la mémoire principale.

kernel.shmmax

Définit la taille maximale en octets d'un seul segment de mémoire partagée qu'un processus Linux peut allouer dans son espace d'adressage virtuel.

kernel.shmmni

Définit le nombre maximum de segments de mémoire partagée que le serveur de base de données peut gérer.

net.ipv4.ip_local_port_range

Définit la plage de ports que le système peut utiliser pour les programmes qui souhaitent se connecter à un serveur de base de données sans numéro de port spécifique.

net.core.rmem_default

Définit la mémoire de la socket de réception par défaut via le protocole de contrôle de transmission (TCP).

net.core.rmem_max

Définit la mémoire maximale de la socket de réception via le protocole de contrôle de transmission (TCP).

net.core.wmem_default

Définit la mémoire de la socket d'envoi par défaut via le protocole de contrôle de transmission (TCP).

net.core.wmem_max

Définit la mémoire maximale de la socket d'envoi via le protocole de contrôle de transmission (TCP).

vm.dirty_bytes / vm.dirty_ratio

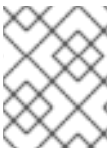
Définit un seuil en octets / en pourcentage de mémoire salissable à partir duquel un processus générant des données salies est lancé dans la fonction **write()**.

NOTE

Either **vm.dirty_bytes** or **vm.dirty_ratio** peuvent être spécifiés à la fois.

vm.dirty_background_bytes / vm.dirty_background_ratio

Définit un seuil en octets / en pourcentage de la mémoire salissable à partir duquel le noyau tente d'écrire activement les données salies sur le disque dur.



NOTE

Either **vm.dirty_background_bytes** or **vm.dirty_background_ratio** peuvent être spécifiés à la fois.

vm.dirty_writeback_centisecs

Définit un intervalle de temps entre les réveils périodiques des threads du noyau responsables de l'écriture des données sales sur le disque dur.

Les paramètres de ce noyau sont mesurés en centièmes de seconde.

vm.dirty_expire_centisecs

Définit le délai après lequel les données sales sont suffisamment anciennes pour être écrites sur le disque dur.

Les paramètres de ce noyau sont mesurés en centièmes de seconde.

Ressources supplémentaires

- *Dirty pagecache writeback and vm.dirty parameters*

CHAPITRE 9. DÉMARRER AVEC LA JOURNALISATION DU NOYAU

Les fichiers journaux sont des fichiers qui contiennent des messages sur le système, y compris le noyau, les services et les applications qui s'y exécutent. Le système de journalisation de Red Hat Enterprise Linux est basé sur le protocole intégré **syslog**. Divers utilitaires utilisent ce système pour enregistrer des événements et les organiser dans des fichiers journaux. Ces fichiers sont utiles lors de l'audit du système d'exploitation ou de la résolution de problèmes.

9.1. QU'EST-CE QUE L'ANNEAU TAMPON DU NOYAU ?

Pendant le processus de démarrage, la console fournit de nombreuses informations importantes sur la phase initiale du démarrage du système. Pour éviter la perte des premiers messages, le noyau utilise ce que l'on appelle un tampon circulaire. Ce tampon stocke tous les messages, y compris les messages de démarrage, générés par la fonction **printk()** dans le code du noyau. Les messages du tampon circulaire du noyau sont ensuite lus et stockés dans des fichiers journaux sur un support permanent, par exemple par le service **syslog**.

Le tampon mentionné ci-dessus est une structure de données cyclique qui a une taille fixe et qui est codée en dur dans le noyau. Les utilisateurs peuvent afficher les données stockées dans le tampon circulaire du noyau par le biais de la commande **dmesg** ou du fichier **/var/log/boot.log**. Lorsque le tampon circulaire est plein, les nouvelles données écrasent les anciennes.

Ressources supplémentaires

- **syslog(2)** et **dmesg(1)** page du manuel

9.2. RÔLE DE PRINTK SUR LES NIVEAUX DE JOURNALISATION ET LA JOURNALISATION DU NOYAU

Chaque message rapporté par le noyau est associé à un niveau de journalisation qui définit l'importance du message. Le tampon circulaire du noyau, tel que décrit dans Qu'est-ce que le tampon circulaire du noyau, recueille les messages du noyau de tous les niveaux de journalisation. C'est le paramètre **kernel.printk** qui définit quels messages du tampon sont imprimés sur la console.

Les valeurs du niveau d'enregistrement se décomposent dans l'ordre suivant :

- 0 - Urgence du noyau. Le système est inutilisable.
- 1 - Alerte du noyau. Des mesures doivent être prises immédiatement.
- 2 - L'état du noyau est considéré comme critique.
- 3 - Condition d'erreur générale du noyau.
- 4 - Condition d'avertissement général du noyau.
- 5 - Notification par le noyau d'une condition normale mais significative.
- 6 - Message d'information du noyau.
- 7 - Messages de débogage du noyau.

Par défaut, **kernel.printk** dans RHEL 9 contient les quatre valeurs suivantes :

```
# sysctl kernel.printk
kernel.printk = 7 4 1 7
```

Les quatre valeurs définissent ce qui suit :

1. valeur. Console log-level, définit la priorité la plus basse des messages imprimés sur la console.
2. valeur. Niveau de journalisation par défaut pour les messages sans niveau de journalisation explicite.
3. valeur. Définit la configuration du niveau de journalisation le plus bas possible pour le niveau de journalisation de la console.
4. valeur. Définit la valeur par défaut du niveau de journalisation de la console au démarrage. Chacune de ces valeurs définit une règle différente pour le traitement des messages d'erreur.

IMPORTANT

La valeur par défaut **7 4 1 7** **printk** permet un meilleur débogage de l'activité du noyau. Toutefois, lorsqu'il est associé à une console série, ce paramètre **printk** peut provoquer des rafales d'E/S intenses susceptibles d'empêcher temporairement un système RHEL de répondre. Pour éviter ce genre de situation, la valeur **printk** de **4 4 1 7** fonctionne généralement, mais au prix de la perte des informations de débogage supplémentaires.

Notez également que certains paramètres de la ligne de commande du noyau, tels que **quiet** ou **debug**, modifient les valeurs par défaut de **kernel.printk**.

Ressources supplémentaires

- **syslog(2)** page du manuel

CHAPITRE 10. LE NOYAU DE 64K PAGES

kernel-64k est un paquet de noyau supplémentaire et optionnel pour l'architecture ARM 64 bits qui prend en charge 64k pages. Ce noyau supplémentaire existe parallèlement au noyau RHEL 9 for ARM qui prend en charge 4k pages.

Les performances optimales du système sont directement liées aux différentes exigences en matière de configuration de la mémoire. Ces exigences sont prises en compte par les deux variantes du noyau, chacune convenant à des charges de travail différentes. RHEL 9 sur matériel ARM 64 bits propose ainsi deux tailles de page MMU :

- noyau de 4k pages pour une utilisation efficace de la mémoire dans les petits environnements,
- **kernel-64k** pour les charges de travail avec de grands ensembles de travail à mémoire contiguë.

Le noyau 4k pages et **kernel-64k** ne diffèrent pas au niveau de l'expérience utilisateur puisque l'espace utilisateur est le même. Vous pouvez choisir la variante qui répond le mieux à votre situation.

noyau de 4k pages

Utilisez des pages 4k pour une utilisation plus efficace de la mémoire dans des environnements plus petits, tels que ceux des instances Edge et des petites instances cloud à faible coût. Dans ces environnements, il n'est pas possible d'augmenter la quantité de mémoire physique du système en raison des contraintes d'espace, d'énergie et de coût. Par ailleurs, tous les processeurs ARM 64 bits ne prennent pas en charge une taille de page de 64 000.
Le noyau de 4 pages prend en charge l'installation graphique à l'aide d'Anaconda, les installations basées sur des images de système ou de nuage, ainsi que les installations avancées à l'aide de Kickstart.

kernel-64k

Le noyau de 64k pages est une option utile pour les grands ensembles de données sur les plates-formes ARM. **kernel-64k** convient aux charges de travail à forte intensité de mémoire car il permet d'améliorer considérablement les performances globales du système, notamment en ce qui concerne les grandes bases de données, le calcul intensif et les performances des réseaux à haut débit.
Vous devez choisir la taille de la page sur les systèmes à architecture ARM 64 bits au moment de l'installation. Vous pouvez installer **kernel-64k** uniquement par Kickstart en ajoutant le paquet **kernel-64k** à la liste des paquets dans le fichier **Kickstart**.

Ressources supplémentaires

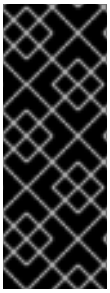
- Effectuer une installation avancée de RHEL 9

CHAPITRE 11. INSTALLATION DE KDUMP

Le service **kdump** est installé et activé par défaut sur les nouvelles installations de Red Hat Enterprise Linux. Découvrez **kdump** et comment installer **kdump** lorsqu'il n'est pas activé par défaut.

11.1. QU'EST-CE QUE KDUMP ?

kdump est un service qui fournit un mécanisme de vidage en cas de panne. Ce service vous permet de sauvegarder le contenu de la mémoire du système à des fins d'analyse. **kdump** utilise l'appel système **kexec** pour démarrer dans le second noyau (a *capture kernel*) sans redémarrer ; il capture ensuite le contenu de la mémoire du noyau accidenté (a *crash dump* ou a *vmcore*) et l'enregistre dans un fichier. Le second noyau réside dans une partie réservée de la mémoire du système.



IMPORTANT

Un crash dump du noyau peut être la seule information disponible en cas de défaillance du système (bogue critique). Par conséquent, le site opérationnel **kdump** est important dans les environnements critiques. Red Hat conseille aux administrateurs système de mettre à jour et de tester régulièrement **kexec-tools** dans le cadre du cycle normal de mise à jour du noyau. Ceci est particulièrement important lorsque de nouvelles fonctionnalités du noyau sont mises en œuvre.

Vous pouvez activer **kdump** pour tous les noyaux installés sur une machine ou seulement pour des noyaux spécifiques. Cette option est utile lorsque plusieurs noyaux sont utilisés sur une machine, dont certains sont suffisamment stables pour ne pas risquer de tomber en panne.

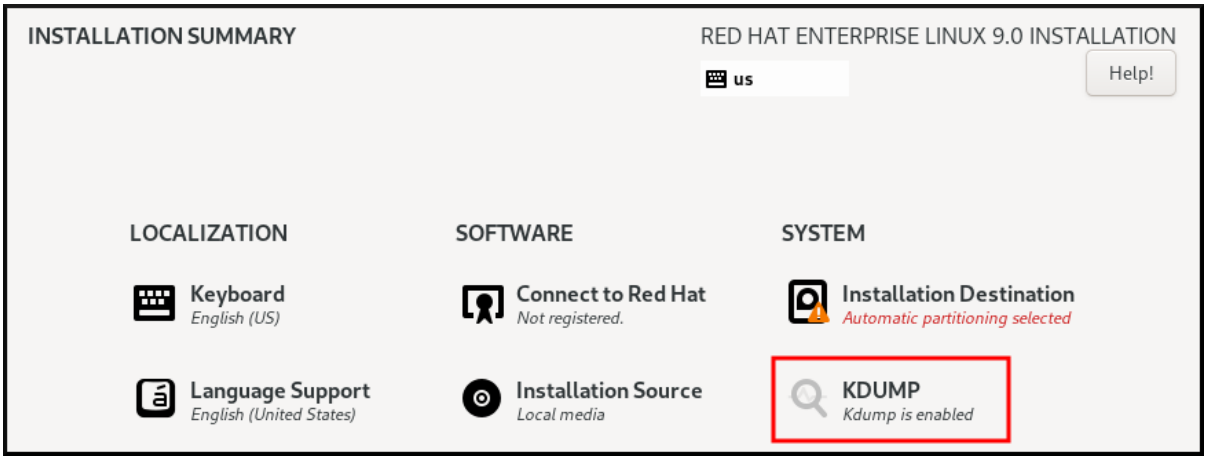
Lors de l'installation de **kdump**, un fichier par défaut **/etc/kdump.conf** est créé. Ce fichier contient la configuration minimale par défaut de **kdump**. Vous pouvez modifier ce fichier pour personnaliser la configuration de **kdump**, mais ce n'est pas obligatoire.

11.2. INSTALLATION DE KDUMP AVEC ANACONDA

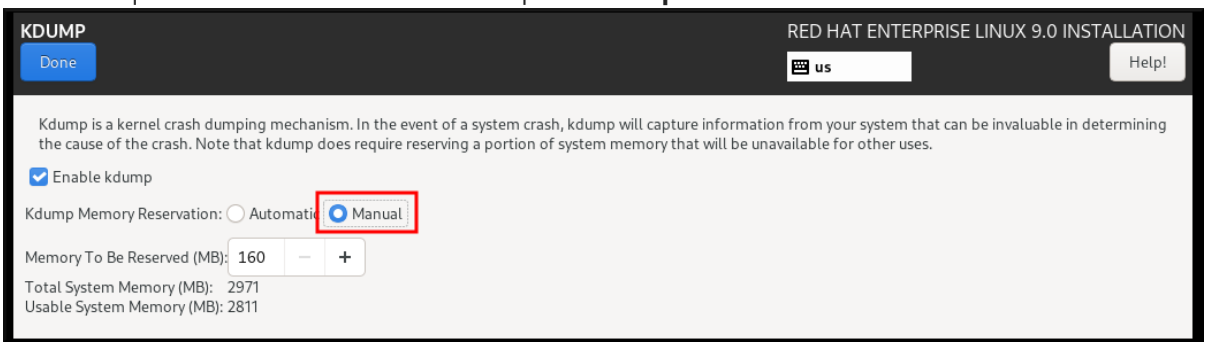
Le programme d'installation **Anaconda** fournit un écran d'interface graphique pour la configuration de **kdump** au cours d'une installation interactive. L'écran de l'installateur est intitulé **KDUMP** et est disponible à partir de l'écran principal **Installation Summary**. Vous pouvez activer **kdump** et réserver la quantité de mémoire nécessaire.

Procédure

1. Allez dans le champ **Kdump**.
2. Activer **kdump** si ce n'est pas déjà fait.



3. Définir la quantité de mémoire à réserver pour **kdump**.



11.3. INSTALLATION DE KDUMP EN LIGNE DE COMMANDE

Certaines options d'installation, telles que les installations personnalisées **Kickstart**, n'installent pas **not** ou n'activent pas **kdump** par défaut. Si c'est le cas, suivez la procédure ci-dessous.

Conditions préalables

- Un abonnement RHEL actif
- Le paquet **kexec-tools**
- Satisfait aux exigences des configurations et des cibles **kdump**. Pour plus de détails, voir Configurations et cibles kdump prises en charge .

Procédure

1. Vérifiez si **kdump** est installé sur votre système :

```
# rpm -q kexec-tools
```

Indique si le paquet est installé :

```
# kexec-tools-2.0.22-13.el9.x86_64
```

Sortie si le paquet n'est pas installé :

```
le paquet kexec-tools n'est pas installé
```

2. Installez **kdump** et les autres paquets nécessaires en :

```
# dnf install kexec-tools
```

CHAPITRE 12. CONFIGURATION DE KDUMP SUR LA LIGNE DE COMMANDE

La mémoire pour **kdump** est réservée lors du démarrage du système. La taille de la mémoire est configurée dans le fichier de configuration du Grand Unified Bootloader (GRUB) du système. La taille de la mémoire dépend de la valeur **crashkernel=** spécifiée dans le fichier de configuration et de la taille de la mémoire physique du système.

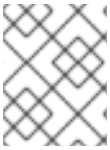
12.1. CONFIGURATION DE L'UTILISATION DE LA MÉMOIRE DE KDUMP

Le paquetage **kexec-tools** conserve les valeurs par défaut de réservation de la mémoire **crashkernel**. Le service **kdump** utilise la valeur par défaut pour réserver la mémoire **crashkernel** pour chaque noyau.

L'allocation automatique de mémoire pour **kdump** varie en fonction de l'architecture matérielle du système et de la taille de la mémoire disponible. Par exemple, sur l'architecture ARM 64 bits, les paramètres par défaut de **crashkernel** ne fonctionnent que lorsque la mémoire disponible est supérieure à 1 Go. **kexec-tools** par défaut, configure la réserve de mémoire suivante sur l'architecture ARM 64 bits :

crashkernel=1G-4G:256M,4G-64G:320M,64G:576M

Les besoins en mémoire du noyau de crash peuvent varier en fonction du matériel et des spécifications de la machine. Si la valeur par défaut de **crashkernel** ne fonctionne pas sur votre système, vous pouvez exécuter la commande **kdumpctl estimate** et demander une estimation approximative sans déclencher de crash. La valeur estimée de **crashkernel** peut ne pas être exacte et peut servir de référence pour définir une valeur appropriée de **crashkernel**.



NOTE

L'option **crashkernel=auto** de la ligne de commande de démarrage n'est plus prise en charge sur RHEL 9 et les versions ultérieures.

Conditions préalables

- Vous disposez des privilèges de root sur le système.
- Satisfait aux exigences de **kdump** en ce qui concerne les configurations et les cibles. Pour plus de détails, voir Configurations et cibles kdump prises en charge
- Sur les systèmes IBM Z, assurez-vous que l'utilitaire **zipl** est installé.

Procédure

1. Configurer la valeur par défaut de **crashkernel**:

```
# kdumpctl reset-crashkernel --kernel=ALL
```

2. (Facultatif) Pour utiliser une valeur personnalisée **crashkernel**:
   a. Configurer la réserve de mémoire nécessaire :

```
crashkernel=1G-4G:192M,4G-:256M
```

L'exemple réserve 192 Mo de mémoire si la quantité totale de mémoire du système est égale ou supérieure à 1 Go et inférieure à 4 Go. Si la quantité totale de mémoire est supérieure à 4 Go, 256 Mo sont réservés pour **kdump**.

- (Facultatif) Décaler la mémoire réservée :
  Certains systèmes exigent de réserver de la mémoire avec un certain décalage fixe, car la réservation du noyau en cas de crash est très précoce, et il veut réserver une zone pour un usage spécial. Si le décalage est défini, la mémoire réservée commence à cet endroit. Pour décaler la mémoire réservée, utilisez la syntaxe suivante :

  ```
  crashkernel=192M@16M
  ```

  L'exemple ci-dessus réserve 192 Mo de mémoire en commençant par 16 Mo (adresse physique 0x01000000). Si le paramètre offset est fixé à 0 ou omis complètement, **kdump** décale automatiquement la mémoire réservée. Vous pouvez également décaler la mémoire lors de la définition d'une réservation de mémoire variable en spécifiant le décalage comme dernière valeur. Par exemple, **crashkernel=1G-4G:192M,2G-64G:256M@16M**.

b. Mettre à jour la configuration du chargeur de démarrage :

```
# grubby --update-kernel ALL --args "crashkernel=<CUSTOM-VALUE>"
```

Sur les systèmes IBM Z qui utilisent le chargeur de démarrage zIPL, la commande ajoute un nouveau paramètre de noyau à chaque fichier **/boot/loader/entries/<ENTRY>.conf**.

- Sur les systèmes IBM Z, pour mettre à jour le menu de démarrage, exécutez la commande **zipl** sans spécifier d'options :

```
# zipl
```

3. Redémarrez pour que les modifications soient prises en compte :

```
# reboot
```

Vérification

1. Activez la touche **sysrq** pour démarrer avec le noyau **kdump**:

```
# echo 1 > /proc/sys/kernel/sysrq
# echo c > /proc/sysrq-trigger
```

Cela oblige le noyau Linux à se planter et à copier le fichier **address-YYYY-MM-DD-HH:MM:SS/vmcore** à l'emplacement cible spécifié dans le fichier de configuration.

2. Vérifiez que le fichier **vmcore** est déversé dans la cible comme spécifié dans le fichier **/etc/kdump.conf**:

```
$ ls /var/crash/127.0.0.1-2022-01-18-0

/var/crash/127.0.0.1-2022-01-18-05:23:10':
kexec-dmesg.log  vmcore  vmcore-dmesg.txt
```

Dans cet exemple, le noyau enregistre le fichier **vmcore** dans le répertoire cible par défaut, **/var/crash/**.

12.2. CONFIGURATION DE LA CIBLE KDUMP

Le crash dump est généralement stocké sous la forme d'un fichier dans un système de fichiers local, écrit directement sur un périphérique. Il est également possible de configurer l'envoi du crash dump sur un réseau à l'aide des protocoles **NFS** ou **SSH**. Une seule de ces options de conservation d'un fichier de crash dump peut être définie à la fois. Par défaut, le fichier est stocké dans le répertoire **/var/crash/** du système de fichiers local.

Conditions préalables

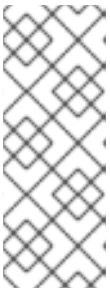
- **Root** autorisations.
- Satisfait aux exigences des configurations et des cibles **kdump**. Pour plus de détails, voir Configurations et cibles kdump prises en charge .

Procédure

- Pour stocker le fichier crash dump dans le répertoire **/var/crash/** du système de fichiers local, modifiez le fichier **/etc/kdump.conf** et indiquez le chemin d'accès :

```
chemin /var/crash
```

L'option **path /var/crash** représente le chemin d'accès au système de fichiers dans lequel **kdump** enregistre le fichier crash dump.



NOTE

- Lorsque vous spécifiez une cible de vidage dans le fichier **/etc/kdump.conf**, le chemin d'accès est **relative** vers la cible de vidage spécifiée.
- Si vous ne spécifiez pas de cible de vidage dans le fichier **/etc/kdump.conf**, le chemin d'accès représente le chemin d'accès de **absolute** à partir du répertoire racine.

En fonction de ce qui est monté dans le système actuel, la cible de vidage et le chemin de vidage ajusté sont pris automatiquement.

Exemple 12.1. La configuration de la cible kdump

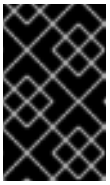
```
# grep -v ^# /etc/kdump.conf | grep -v ^$
ext4 /dev/mapper/vg00-varcrashvol
path /var/crash
core_collector makedumpfile -c --message-level 1 -d 31
```

Ici, la cible de vidage est spécifiée (**ext4 /dev/mapper/vg00-varcrashvol**), et donc montée à **/var/crash**. L'option **path** est également définie sur **/var/crash**, de sorte que **kdump** enregistre le fichier **vmcore** dans le répertoire **/var/crash/var/crash**.

- Pour modifier le répertoire local dans lequel le crash dump doit être sauvegardé, en tant que **root**, éditez le fichier de configuration **/etc/kdump.conf**:

1. Supprimer le signe dièse ("#") au début de la ligne **#path /var/crash**.
2. Remplacez la valeur par le chemin d'accès au répertoire souhaité. Par exemple :

```
chemin /usr/local/cores
```



IMPORTANT

Dans RHEL 9, le répertoire défini comme cible du kdump à l'aide de la directive **path** doit exister lorsque le service **kdump systemd** est démarré - sinon le service échoue.

- Pour écrire le fichier sur une autre partition, modifiez le fichier de configuration **/etc/kdump.conf**:
  1. Supprimer le signe dièse ("#") au début de la ligne **#ext4**, en fonction de votre choix.
     - le nom de l'appareil (ligne **#ext4 /dev/vg/lv_kdump**)
     - étiquette du système de fichiers (ligne **#ext4 LABEL=/boot**)
     - UUID (ligne **#ext4 UUID=03138356-5e61-4ab3-b58e-27507ac41937**)
  2. Modifiez le type de système de fichiers ainsi que le nom, l'étiquette ou l'UUID du périphérique en fonction des valeurs souhaitées. Par exemple :

```
ext4 UUID=03138356-5e61-4ab3-b58e-27507ac41937
```

NOTE

La syntaxe correcte pour spécifier les valeurs UUID est **UUID="correct-uuid"** et **UUID=correct-uuid**.

IMPORTANT

Il est recommandé de spécifier les périphériques de stockage à l'aide d'un **LABEL=** ou d'un **UUID=**. La cohérence des noms de périphériques de disque tels que **/dev/sda3** n'est pas garantie au redémarrage.

IMPORTANT

Lors de la vidange vers un dispositif de stockage à accès direct (DASD) sur du matériel IBM Z, il est essentiel que les dispositifs de vidange soient correctement spécifiés dans **/etc/dasd.conf** avant de procéder.

- Pour écrire le crash dump directement sur un périphérique, modifiez le fichier de configuration **/etc/kdump.conf**:
  1. Supprimer le signe dièse ("#") au début de la ligne **#raw /dev/vg/lv_kdump**.
  2. Remplacez la valeur par le nom de l'appareil prévu. Par exemple :

```
brut /dev/sdb1
```

- Pour enregistrer le crash dump sur une machine distante à l'aide du protocole **NFS**:

  1. Supprimer le signe dièse ("#") au début de la ligne **#nfs my.server.com:/export/tmp**.
  2. Remplacez la valeur par un nom d'hôte et un chemin d'accès au répertoire valides. Par exemple :

     ```
     nfs penguin.example.com:/export/cores
     ```

- Pour enregistrer le crash dump sur une machine distante à l'aide du protocole **SSH**:

  1. Supprimer le signe dièse ("#") au début de la ligne **#ssh user@my.server.com** ligne.
  2. Remplacez la valeur par un nom d'utilisateur et un nom d'hôte valides.
  3. Incluez votre clé **SSH** dans la configuration.
     - Supprimer le signe dièse au début de la ligne **#sshkey /root/.ssh/kdump_id_rsa**.
     - Remplacez la valeur par l'emplacement d'une clé valide sur le serveur vers lequel vous essayez de faire un dumping. Par exemple :

       ```
       ssh john@penguin.example.com
       sshkey /root/.ssh/mykey
       ```

12.3. CONFIGURATION DU COLLECTEUR PRINCIPAL

Le service **kdump** utilise le programme **core_collector** pour capturer l'image **vmcore**. Dans RHEL, l'utilitaire **makedumpfile** est le collecteur de noyau par défaut.

makedumpfile est un programme de vidage qui permet de copier uniquement les pages nécessaires en utilisant différents niveaux de vidage et de compresser la taille d'un fichier de vidage.

makedumpfile est un programme de vidage qui permet de ne copier que les pages nécessaires en utilisant différents niveaux de vidage et de compresser la taille d'un fichier de vidage.

En utilisant **makedumpfile**, vous pouvez créer un fichier de vidage de petite taille soit en compressant les données de vidage, soit en excluant des pages, soit les deux. Il a besoin des informations de débogage du premier noyau pour distinguer les pages inutiles en analysant la façon dont le premier noyau utilise la mémoire.

Syntaxe

```
core_collector makedumpfile -z -d 31 --message-level 1
```

Options

- **-c**, **-l**, **-z**, ou **-p**: spécifie le format du fichier compressé pour chaque page lorsque vous utilisez l'une de ces options : **-c** pour **zlib**, **-l** pour **lzo**, **-z** pour **zstd**, ou **-p** pour **snappy**.
- **-d (dump_level)** le paramètre : exclut des pages afin qu'elles ne soient pas copiées dans le fichier de vidage.
- **--message-level**: spécifie les types de messages.

En utilisant **--message-level**, vous pouvez limiter les sorties à imprimer. Par exemple, en spécifiant 7 comme niveau de message, les messages courants et les messages d'erreur sont imprimés. La valeur maximale pour **--message_level** est 31.

Conditions préalables

- Respecter les exigences de **kdump** pour les configurations et les objectifs.

Procédure

1. En tant qu'utilisateur de **root**, modifiez le fichier de configuration de **/etc/kdump.conf** pour supprimer le signe de hachage ("#") au début de la commande suivante :

```
core_collector makedumpfile -z -d 31 --message-level 1
```

2. Pour activer la compression du fichier dump, spécifiez l'une des options **makedumpfile**:

```
core_collector makedumpfile -z -d 31 --message-level 1
```

où,

- **-z** spécifie le format de fichier compressé **dump**.
- **-d** spécifie que le niveau de vidage est 31.
- **--message-level** spécifie que le niveau du message est 1.

Prenons également l'exemple suivant, qui utilise **-l**:

- Pour compresser un fichier dump à l'aide de **-l**:

```
core_collector makedumpfile -l -d 31 --message-level 1
```

Ressources supplémentaires

- **makedumpfile(8)** page du manuel

12.4. CONFIGURATION DES RÉPONSES D'ÉCHEC PAR DÉFAUT DE KDUMP

Par défaut, lorsque **kdump** ne parvient pas à créer un fichier de vidage de crash à l'emplacement cible configuré, le système redémarre et le vidage est perdu dans le processus. Pour modifier ce comportement, suivez la procédure ci-dessous.

Conditions préalables

- Vous disposez des droits d'accès au système.
- Satisfait aux exigences des configurations et des objectifs de **kdump**.

Procédure

1. Comme pour **root**, supprimez le signe de hachage ("#") au début de la ligne **#failure_action** dans le fichier de configuration **/etc/kdump.conf**.

2. Remplacer la valeur par une action souhaitée.

```
failure_action poweroff
```

12.5. FICHIER DE CONFIGURATION POUR KDUMP

Le fichier de configuration du noyau **kdump** est **/etc/sysconfig/kdump**. Ce fichier contrôle les paramètres de la ligne de commande du noyau **kdump**.

Pour la plupart des configurations, utilisez les options par défaut. Toutefois, dans certains cas, vous devrez peut-être modifier certains paramètres pour contrôler le comportement du noyau **kdump**. Par exemple, modifier l'option **KDUMP_COMMANDLINE_APPEND** pour ajouter la ligne de commande du noyau **kdump** afin d'obtenir une sortie de débogage détaillée ou l'option **KDUMP_COMMANDLINE_REMOVE** pour supprimer les arguments de la ligne de commande **kdump**.

Pour plus d'informations sur les options de configuration supplémentaires, voir **Documentation/admin-guide/kernel-parameters.txt** ou le fichier **/etc/sysconfig/kdump**.

- **KDUMP_COMMANDLINE_REMOVE**
  Cette option supprime les arguments de la ligne de commande **kdump** actuelle. Elle supprime les paramètres susceptibles de provoquer des erreurs sur **kdump** ou des échecs de démarrage du noyau sur **kdump**. Ces paramètres peuvent avoir été analysés par le processus **KDUMP_COMMANDLINE** précédent ou hérités du fichier **/proc/cmdline**. Lorsque cette variable n'est pas configurée, elle hérite de toutes les valeurs du fichier **/proc/cmdline**. La configuration de cette option fournit également des informations utiles pour le débogage d'un problème.

  Example

  Pour supprimer certains arguments, ajoutez-les à **KDUMP_COMMANDLINE_REMOVE** comme suit :

  ```
  # KDUMP_COMMANDLINE_REMOVE="hugepages hugepagesz slub_debug quiet log_buf_len swiotlb"
  ```

- **KDUMP_COMMANDLINE_APPEND**
  Cette option ajoute des arguments à la ligne de commande actuelle. Ces arguments peuvent avoir été analysés par la variable **KDUMP_COMMANDLINE_REMOVE** précédente.

  Pour le noyau **kdump**, la désactivation de certains modules tels que **mce**, **cgroup**, **numa**, **hest_disable** peut aider à prévenir les erreurs du noyau. Ces modules peuvent consommer une part importante de la mémoire du noyau réservée à **kdump** ou provoquer des échecs de démarrage du noyau **kdump**.

  Example

  Pour désactiver la mémoire **cgroups** sur la ligne de commande du noyau **kdump**, exécutez la commande suivante :

  ```
  # KDUMP_COMMANDLINE_APPEND="cgroup_disable=memory"
  ```

Ressources supplémentaires

- **Documentation/admin-guide/kernel-parameters.txt** fichier

- **/etc/sysconfig/kdump** fichier

12.6. ACTIVATION ET DÉSACTIVATION DU SERVICE KDUMP

Pour lancer le service **kdump** au moment du démarrage, suivez la procédure ci-dessous.

Conditions préalables

- Respecter les exigences de **kdump** pour les configurations et les objectifs.
- Toutes les configurations pour l'installation de **kdump** sont établies en fonction de vos besoins.

Procédure

1. Pour activer le service **kdump**, utilisez la commande suivante :

   ```
   # systemctl enable kdump.service
   ```

   Cela permet au service de se rendre sur le site **multi-user.target**.

2. Pour démarrer le service dans la session en cours, utilisez la commande suivante :

   ```
   # systemctl start kdump.service
   ```

3. Pour arrêter le service **kdump**, tapez la commande suivante :

   ```
   # systemctl stop kdump.service
   ```

4. Pour désactiver le service **kdump**, exécutez la commande suivante :

   ```
   # systemctl disable kdump.service
   ```

AVERTISSEMENT

Il est recommandé de définir **kptr_restrict=1** comme valeur par défaut. Lorsque **kptr_restrict** est défini sur (1) par défaut, le service **kdumpctl** charge le noyau de secours même si KASLR (Kernel Address Space Layout) est activé ou non.

Étape de dépannage

Lorsque **kptr_restrict** n'a pas la valeur (1) et que KASLR est activé, le contenu du fichier **/proc/kore** est généré sous la forme de zéros. Par conséquent, le service **kdumpctl** ne parvient pas à accéder à **/proc/kcore** et à charger le noyau de secours.

Pour contourner ce problème, le fichier **kexec-kdump-howto.txt** affiche un message d'avertissement, qui précise de conserver le paramètre recommandé, à savoir **kptr_restrict=1**.

Pour s'assurer que le service **kdumpctl** charge le noyau de crash, vérifiez que :

- Kernel **kptr_restrict=1** dans le fichier **sysctl.conf**.

12.7. TEST DE LA CONFIGURATION DE KDUMP

Vous pouvez tester le fonctionnement et la validité du processus de crash dump avant que la machine n'entre en production.

AVERTISSEMENT

Les commandes ci-dessous provoquent un crash du noyau. Soyez prudent lorsque vous suivez ces étapes, et ne les utilisez jamais sans précaution sur un système de production actif.

Procédure

1. Redémarrez le système avec **kdump** activé.
2. Assurez-vous que **kdump** est en cours d'exécution :

```
# systemctl is-active kdump
active
```

3. Forcer le noyau Linux à se bloquer :

```
echo 1 > /proc/sys/kernel/sysrq
echo c > /proc/sysrq-trigger
```

AVERTISSEMENT

La commande ci-dessus plante le noyau, et un redémarrage est nécessaire.

Une fois redémarré, le fichier ***address-YYYY-MM-DD-HH:MM:SS*/vmcore** est créé à l'emplacement que vous avez spécifié dans le fichier **/etc/kdump.conf** (par défaut **/var/crash/**).

NOTE

Cette action confirme la validité de la configuration. Il est également possible d'utiliser cette action pour enregistrer le temps nécessaire à l'exécution d'un crash dump avec une charge de travail représentative.

Ressources supplémentaires

- Configuration de la cible kdump

12.8. EMPÊCHER LE CHARGEMENT DES PILOTES DU NOYAU POUR KDUMP

Vous pouvez empêcher le noyau de capture de charger certains pilotes du noyau en ajoutant la variable **KDUMP_COMMANDLINE_APPEND=** dans le fichier de configuration **/etc/sysconfig/kdump**. En utilisant cette méthode, vous pouvez empêcher l'image de disque RAM initiale **kdump initramfs** de charger le module de noyau spécifié. Cela permet d'éviter les erreurs de killer out-of-memory (oom) ou d'autres défaillances du noyau de capture.

Vous pouvez ajouter la variable **KDUMP_COMMANDLINE_APPEND=** en utilisant l'une des options de configuration suivantes :

- **rd.driver.blacklist=<modules>**
- **modprobe.blacklist=<modules>**

Procédure

1. Sélectionnez un module du noyau dont vous souhaitez bloquer le chargement.

```
$ lsmod

Module                  Size  Used by
fuse                  126976  3
xt_CHECKSUM            16384  1
ipt_MASQUERADE         16384  1
uinput                 20480  1
xt_conntrack           16384  1
```

   La commande **lsmod** affiche une liste des modules chargés dans le noyau en cours d'exécution.

2. Mettre à jour la variable **KDUMP_COMMANDLINE_APPEND=** dans le fichier **/etc/sysconfig/kdump**.

```
# KDUMP_COMMANDLINE_APPEND="rd.driver.blacklist=hv_vmbus,hv_storvsc,hv_utils,
hv_netvsc,hid-hyperv"
```

   Prenons également l'exemple suivant, qui utilise l'option de configuration **modprobe.blacklist=<modules>**.

```
# KDUMP_COMMANDLINE_APPEND="modprobe.blacklist=emcp
modprobe.blacklist=bnx2fc modprobe.blacklist=libfcoe modprobe.blacklist=fcoe"
```

3. Redémarrez le service **kdump**.

```
# systemctl restart kdump
```

Ressources supplémentaires

- **dracut.cmdline** page de manuel

12.9. EXÉCUTION DE KDUMP SUR DES SYSTÈMES DONT LE DISQUE EST CRYPTÉ

Lorsque vous exécutez une partition chiffrée Linux Unified Key Setup (LUKS), le système a besoin d'une certaine quantité de mémoire disponible. Si le système dispose d'une quantité de mémoire disponible inférieure à celle requise, le service **systemd-cryptsetup** ne parvient pas à monter la partition. Par conséquent, la capture du fichier **vmcore** vers un emplacement cible crypté échoue dans le deuxième noyau (noyau de capture).

La commande **kdumpctl estimate** vous aide à estimer la quantité de mémoire dont vous avez besoin pour **kdump**. Elle imprime la valeur recommandée pour **crashkernel**, qui est la taille de mémoire la plus appropriée pour **kdump**.

La valeur recommandée de **crashkernel** est calculée en fonction de la taille actuelle du noyau, des modules du noyau, de **initramfs** et de la mémoire cible cryptée LUKS requise.

Si vous utilisez l'option personnalisée **crashkernel**, **kdumpctl estimate** imprime la valeur **LUKS required size**. Cette valeur correspond à la taille de la mémoire requise pour la cible cryptée LUKS.

Procédure

1. Imprimer la valeur estimée de **crashkernel**:

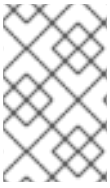
```
# kdumpctl estimate

Encrypted kdump target requires extra memory, assuming using the keyslot with minimum memory requirement
  Reserved crashkernel:    256M
  Recommended crashkernel: 652M
  Kernel image size:   47M
  Kernel modules size: 8M
  Initramfs size:      20M
  Runtime reservation: 64M
  LUKS required size:  512M
  Large modules: none
  WARNING: Current crashkernel size is lower than recommended size 652M.
```

2. Configurez la quantité de mémoire requise en augmentant **crashkernel** jusqu'à la valeur souhaitée.

```
# grubby –args=”crashkernel=652M” --update-kernel=ALL
```

3. Redémarrez pour que les modifications soient prises en compte.

```
# reboot
```



NOTE

Si le service **kdump** ne parvient toujours pas à enregistrer le fichier dump sur la cible cryptée, augmentez progressivement la valeur de **crashkernel** pour configurer une quantité de mémoire appropriée.

CHAPITRE 13. CONFIGURATIONS ET CIBLES KDUMP PRISES EN CHARGE

13.1. MÉMOIRE REQUISE POUR KDUMP

Pour que **kdump** puisse capturer un crash dump du noyau et le sauvegarder en vue d'une analyse ultérieure, une partie de la mémoire du système doit être réservée en permanence au noyau de capture. Lorsqu'elle est réservée, cette partie de la mémoire système n'est pas disponible pour le noyau principal.

Les besoins en mémoire varient en fonction de certains paramètres du système. L'un des principaux facteurs est l'architecture matérielle du système. Pour connaître l'architecture exacte de la machine (telle que Intel 64 et AMD64, également connue sous le nom de x86_64) et l'imprimer sur la sortie standard, utilisez la commande suivante :

```
$ uname -m
```

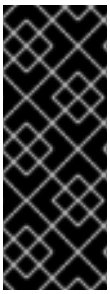
Le tableau suivant répertorie les exigences minimales en matière de mémoire afin de réserver automatiquement une taille de mémoire pour **kdump** sur les dernières versions disponibles. La taille varie en fonction de l'architecture du système et de la mémoire physique totale disponible.

Tableau 13.1. Quantité minimale de mémoire réservée requise pour kdump

Architecture	Mémoire disponible	Mémoire minimale réservée
AMD64 et Intel 64 (**x86_64**)	1 GB à 4 GB	160 Mo de RAM.
	4 Go à 64 Go	192 Mo de RAM.
	64 Go à 1 TB	256 Mo de RAM.
	1 TB et plus	512 Mo de RAM.
architecture ARM 64 bits (**arm64**)	2 Go et plus	448 Mo de RAM.
IBM Power Systems (**ppc64le**)	2 GB à 4 GB	384 Mo de RAM.
	4 Go à 16 Go	512 Mo de RAM.
	16 Go à 64 Go	1 Go de RAM.
	64 Go à 128 Go	2 Go de RAM.
	128 Go et plus	4 Go de RAM.
IBM Z (**s390x**)	1 GB à 4 GB	160 Mo de RAM.
	4 Go à 64 Go	192 Mo de RAM.

Architecture	Mémoire disponible	Mémoire minimale réservée
	64 Go à 1 TB	256 Mo de RAM.
	1 TB et plus	512 Mo de RAM.

Sur de nombreux systèmes, **kdump** est en mesure d'estimer la quantité de mémoire nécessaire et de la réserver automatiquement. Ce comportement est activé par défaut, mais ne fonctionne que sur les systèmes qui disposent de plus d'une certaine quantité de mémoire totale disponible, qui varie en fonction de l'architecture du système.



IMPORTANT

La configuration de la mémoire réservée basée sur la quantité totale de mémoire dans le système est une estimation au mieux. La mémoire réellement nécessaire peut varier en fonction d'autres facteurs tels que les périphériques d'E/S. L'utilisation d'une mémoire insuffisante peut empêcher un noyau de débogage de démarrer en tant que noyau de capture en cas de panique du noyau. Pour éviter ce problème, augmentez suffisamment la mémoire du noyau de débogage.

Ressources supplémentaires

- Comment le paramètre crashkernel a-t-il changé entre les versions mineures de RHEL8 ?
- Tableaux des capacités et des limites technologiques

13.2. SEUIL MINIMAL DE RÉSERVATION DE LA MÉMOIRE

L'utilitaire **kexec-tools** configure par défaut le paramètre de ligne de commande **crashkernel** et réserve une certaine quantité de mémoire pour **kdump**. Pour que la réservation de mémoire par défaut fonctionne, le système doit disposer d'une certaine quantité de mémoire totale. La quantité de mémoire requise varie en fonction de l'architecture du système.

Le tableau suivant répertorie les valeurs seuils minimales pour l'allocation de la mémoire. Si le système dispose d'une mémoire inférieure à la valeur seuil spécifiée, vous devez configurer la mémoire manuellement.

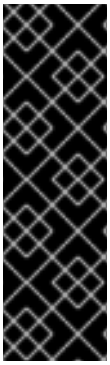
Tableau 13.2. Quantité minimale de mémoire requise pour la réservation de mémoire

Architecture	Mémoire requise
AMD64 et Intel 64 (**x86_64**)	1 GB
IBM Power Systems (**ppc64le**)	2 GB
IBM Z (**s390x**)	1 GB
ARM (aarch64)	2 GB

13.3. CIBLES KDUMP PRISES EN CHARGE

Lorsqu'un crash du noyau est capturé, le fichier dump vmcore peut être écrit directement sur un périphérique, stocké en tant que fichier sur un système de fichiers local ou envoyé sur un réseau. Le tableau ci-dessous contient une liste complète des cibles de vidage qui sont actuellement prises en charge ou explicitement non prises en charge par **kdump**.

Type	Objectifs soutenus	Cibles non soutenues
Dispositif brut	Tous les disques et partitions bruts attachés localement.	
Système de fichiers local	**ext2**, **ext3**, **ext4**, et **xfs** sur des lecteurs de disques directement connectés, des lecteurs logiques RAID matériels, des périphériques LVM, et des matrices **mdraid**.	Tout système de fichiers local non explicitement répertorié comme pris en charge dans ce tableau, y compris le type **auto** (détection automatique du système de fichiers).
Répertoire distant	Les répertoires distants auxquels on accède en utilisant le protocole **NFS** ou **SSH** sur **IPv4**.	Répertoires distants du système de fichiers **rootfs** auxquels on accède en utilisant le protocole **NFS**.
Accès à des répertoires distants à l'aide du protocole **iSCSI** via des initiateurs matériels et logiciels.	Répertoires distants accessibles à l'aide du protocole **iSCSI** sur le matériel **be2iscsi**.	Stockages basés sur des chemins multiples.
		Accès aux répertoires distants via **IPv6**.
		Répertoires distants auxquels on accède en utilisant le protocole **SMB** ou **CIFS**.
		Répertoires distants auxquels on accède en utilisant le protocole **FCoE** (*Fibre Channel over Ethernet*).
		Répertoires distants accessibles à l'aide d'interfaces de réseau sans fil.



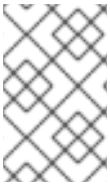
IMPORTANT

L'utilisation du dump assisté par le micrologiciel (**fadump**) pour capturer un vmcore et le stocker sur une machine distante à l'aide du protocole SSH ou NFS entraîne le renommage de l'interface réseau en **kdump-<interface-name>**. Le renommage se produit si le **<interface-name>** est générique, par exemple ***eth#**, **net#**, et ainsi de suite. Ce problème survient parce que les scripts de capture vmcore dans le disque RAM initial (**initrd**) ajoutent le préfixe *kdump-* au nom de l'interface réseau pour garantir un nommage persistant. Comme le même **initrd** est également utilisé pour un démarrage normal, le nom de l'interface est également modifié pour le noyau de production.

13.4. NIVEAUX DE FILTRAGE KDUMP PRIS EN CHARGE

Pour réduire la taille du fichier dump, **kdump** utilise le collecteur de base **makedumpfile** pour compresser les données et, éventuellement, pour omettre les informations indésirables. Le tableau ci-dessous contient une liste complète des niveaux de filtrage actuellement pris en charge par l'utilitaire **makedumpfile**.

Option	Description
1	Zéro page
2	Pages de cache
4	Cache privé
8	Pages d'utilisateurs
16	Pages gratuites



NOTE

La commande **makedumpfile** permet de supprimer les pages énormes transparentes et les pages hugetlbfs. Considérez ces deux types de pages hugepages comme des pages utilisateur et supprimez-les en utilisant le niveau **-8**.

13.5. RÉPONSES D'ÉCHEC PAR DÉFAUT PRISES EN CHARGE

Par défaut, lorsque **kdump** ne parvient pas à créer un core dump, le système d'exploitation redémarre. Vous pouvez cependant configurer **kdump** pour qu'il effectue une opération différente s'il ne parvient pas à enregistrer le core dump sur la cible principale. Le tableau ci-dessous répertorie toutes les actions par défaut actuellement prises en charge.

Option	Description
dump_to_rootfs	Tenter de sauvegarder le core dump dans le système de fichiers racine. Cette option est particulièrement utile en combinaison avec une cible réseau : si la cible réseau est inaccessible, cette option configure kdump pour qu'il enregistre le core dump localement. Le système est ensuite redémarré.
reboot	Redémarrer le système, en perdant le core dump dans le processus.
halt	Arrêter le système, en perdant le core dump dans le processus.
poweroff	Eteindre le système, en perdant le core dump dans le processus.

Option	Description
shell	Exécute une session shell à partir des initramfs, permettant à l'utilisateur d'enregistrer manuellement le core dump.
final_action	Active des opérations supplémentaires telles que les actions **reboot**, **halt**, et **poweroff** après une action réussie **kdump** ou lorsque l'action d'échec **shell** ou **dump_to_rootfs** se termine. L'option **final_action** par défaut est **reboot**.
failure_action	Spécifie l'action à effectuer lorsqu'un dump risque d'échouer en cas de crash du noyau. L'option par défaut **failure_action** est **reboot**.

13.6. UTILISATION DU PARAMÈTRE FINAL_ACTION

Le paramètre **final_action** vous permet d'utiliser certaines opérations supplémentaires telles que les actions **reboot**, **halt** et **poweroff** après la réussite de **kdump** ou à la fin d'un mécanisme **failure_action** invoqué à l'aide de **shell** ou **dump_to_rootfs**. Si l'option **final_action** n'est pas spécifiée, la valeur par défaut est **reboot**.

Procédure

1. Pour configurer **final_action**, modifiez le fichier **/etc/kdump.conf** et ajoutez l'une des options suivantes :

```
# final_action <reboot | halt | poweroff>
```

2. Redémarrez le service **kdump** pour que les modifications soient prises en compte :

```
# kdumpctl restart
```

13.7. UTILISATION DU PARAMÈTRE FAILURE_ACTION

Le paramètre **failure_action** spécifie l'action à effectuer lorsqu'un dump échoue dans le cas d'un crash du noyau. L'action par défaut pour **failure_action** est **reboot**, qui redémarre le système.

failure_action spécifie l'une des actions suivantes à entreprendre :

- **reboot**le système est redémarré après un échec de la vidange.
- **dump_to_rootfs**le fichier dump est enregistré sur le système de fichiers racine lorsqu'une cible de dump non racine est configurée.
- **halt**: arrête le système.
- **poweroff**arrête les opérations en cours sur le système.
- **shell**: démarre une session shell à l'intérieur de **initramfs**, à partir de laquelle vous pouvez effectuer manuellement des actions de récupération supplémentaires.

Procédure :

1. Pour configurer une action à entreprendre en cas d'échec du vidage, modifiez le fichier **/etc/kdump.conf** et spécifiez l'une des options **failure_action**:

```
# failure_action <reboot | halt | poweroff | shell | dump_to_rootfs>
```

2. Redémarrez le service **kdump** pour que les modifications soient prises en compte :

```
# kdumpctl restart
```

CHAPITRE 14. MÉCANISMES DE VIDAGE ASSISTÉS PAR MICROPROGRAMME

Firmware assisted dump (fadump) est un mécanisme de capture de dump, fourni comme alternative au mécanisme **kdump** sur les systèmes IBM POWER. Les mécanismes **kexec** et **kdump** sont utiles pour capturer les vidages du noyau sur les systèmes AMD64 et Intel 64. Cependant, certains matériels tels que les mini-systèmes et les ordinateurs centraux exploitent le micrologiciel embarqué pour isoler des régions de la mémoire et empêcher tout écrasement accidentel de données importantes pour l'analyse du crash. L'utilitaire **fadump** est optimisé pour les mécanismes **fadump** et leur intégration avec RHEL sur les systèmes IBM POWER.

14.1. VIDAGE ASSISTÉ DU MICROLOGICIEL SUR LE MATÉRIEL POWERPC D'IBM

L'utilitaire **fadump** capture le fichier **vmcore** à partir d'un système entièrement réinitialisé avec des périphériques PCI et E/S. Ce mécanisme utilise le microprogramme pour préserver les régions de mémoire lors d'un crash. Ce mécanisme utilise le micrologiciel pour préserver les régions de mémoire lors d'une panne, puis réutilise les scripts de l'espace utilisateur **kdump** pour sauvegarder le fichier **vmcore**. Les régions de mémoire comprennent tout le contenu de la mémoire du système, à l'exception de la mémoire d'amorçage, des registres du système et des entrées de la table des pages matérielles (PTE).

Le mécanisme **fadump** offre une fiabilité accrue par rapport au type de vidage traditionnel, en redémarrant la partition et en utilisant un nouveau noyau pour vidanger les données du crash du noyau précédent. Le site **fadump** nécessite une plate-forme matérielle basée sur le processeur IBM POWER6 ou une version ultérieure.

Pour plus de détails sur le mécanisme **fadump**, y compris les méthodes de réinitialisation du matériel propres à PowerPC, voir le fichier **/usr/share/doc/kexec-tools/fadump-howto.txt**.

NOTE

La zone de mémoire qui n'est pas préservée, connue sous le nom de mémoire d'amorçage, est la quantité de RAM nécessaire pour amorcer le noyau avec succès après une panne. Par défaut, la taille de la mémoire d'amorçage est de 256 Mo ou de 5 % de la mémoire vive totale du système, la valeur la plus élevée étant retenue.

Contrairement à l'événement **kexec-initiated**, le mécanisme **fadump** utilise le noyau de production pour récupérer un crash dump. Lors du redémarrage après une panne, le matériel PowerPC met le nœud de périphérique **/proc/device-tree/rtas/ibm.kernel-dump** à la disposition du système de fichiers **proc** (**procfs**). Les scripts **fadump-aware kdump** vérifient la présence du fichier **vmcore**, puis terminent le redémarrage du système proprement.

14.2. ACTIVATION DU MÉCANISME DE VIDAGE ASSISTÉ PAR LE MICROLOGICIEL

Vous pouvez améliorer les capacités de crash dumping des systèmes IBM POWER en activant le mécanisme de firmware assisted dump (**fadump**).

Dans l'environnement Secure Boot, le chargeur de démarrage **GRUB2** alloue une région de mémoire de démarrage, appelée Real Mode Area (RMA). La RMA a une taille de 512 Mo, qui est divisée entre les composants de démarrage et, si un composant dépasse la taille qui lui a été allouée, **GRUB2** échoue avec une erreur de mémoire insuffisante (**OOM**).

AVERTISSEMENT

N'activez pas le mécanisme de vidage assisté du micrologiciel (**fadump**) dans l'environnement Secure Boot sur RHEL 9.1 et les versions antérieures. Le chargeur de démarrage **GRUB2** échoue avec l'erreur suivante :

```
error: ../../grub-core/kern/mm.c:376:out of memory.
Press any key to continue…
```

Le système n'est récupérable que si vous augmentez la taille par défaut de **initramfs** en raison de la configuration de **fadump**.

Pour plus d'informations sur les méthodes de contournement permettant de restaurer le système, voir l'article Le démarrage du système s'arrête dans GRUB Out of Memory (OOM).

Conditions préalables

- Vous avez des privilèges d'administrateur (root privileges).

Procédure

1. Installez le paquetage **kexec-tools**.
2. Configurer la valeur par défaut de **crashkernel**.

```
# kdumpctl reset-crashkernel –fadump=on --kernel=ALL
```

3. (Facultatif) Réserver la mémoire de démarrage au lieu de la valeur par défaut.

```
# grubby --update-kernel ALL --args=”fadump=on crashkernel=xxM"
```

où **xx** est la taille de la mémoire requise en mégaoctets.

NOTE

Lorsque vous spécifiez des options de configuration de démarrage, testez les configurations en redémarrant le noyau avec **kdump** activé. Si le noyau **kdump** ne démarre pas, augmentez progressivement la valeur de **crashkernel** pour définir une valeur appropriée.

4. Redémarrez pour que les modifications soient prises en compte.

```
# reboot
```

14.3. MÉCANISMES DE VIDAGE ASSISTÉS PAR MICROPROGRAMME SUR LE MATÉRIEL IBM Z

Les systèmes IBM Z prennent en charge deux mécanismes de vidage assistés par microprogramme : Stand-alone dump (**sadump**) et **VMDUMP** dump file.

L'infrastructure **kdump** est prise en charge et utilisée sur les systèmes IBM Z. Toutefois, l'utilisation de l'une des méthodes de vidage assisté du micrologiciel (**fadump**) pour IBM Z peut offrir divers avantages :

- Le mécanisme **sadump** est lancé et contrôlé à partir de la console du système et est stocké sur un périphérique amorçable **IPL**.
- Le mécanisme **VMDUMP** est similaire à celui de **sadump**. Cet outil est lancé à partir de la console du système, mais il récupère le dump résultant du matériel et le copie sur le système pour l'analyser.
- Ces méthodes, similaires à d'autres mécanismes de vidage basés sur le matériel, permettent de capturer l'état d'une machine au début de la phase de démarrage, avant que le service **kdump** ne démarre.
- Bien que **VMDUMP** contienne un mécanisme permettant de recevoir le fichier dump dans un système Red Hat Enterprise Linux, la configuration et le contrôle de **VMDUMP** sont gérés à partir de la console IBM Z Hardware.

Ressources supplémentaires

- Utilisation des outils de vidage sur Red Hat Enterprise Linux 8.5
- Décharge autonome
- Création de dumps sur z/VM avec VMDUMP

14.4. UTILISATION DE SADUMP SUR LES SYSTÈMES FUJITSU PRIMEQUEST

Le mécanisme Fujitsu **sadump** est conçu pour fournir une capture de vidage **fallback** dans le cas où **kdump** n'est pas en mesure de s'achever avec succès. Le mécanisme **sadump** est invoqué manuellement à partir de l'interface de la carte de gestion du système (MMB). En utilisant le MMB, configurez **kdump** comme pour un serveur Intel 64 ou AMD 64, puis effectuez les étapes supplémentaires suivantes pour activer **sadump**.

Procédure

1. Ajoutez ou modifiez les lignes suivantes dans le fichier **/etc/sysctl.conf** pour vous assurer que **kdump** démarre comme prévu pour **sadump**:

```
kernel.panic=0
kernel.unknown_nmi_panic=1
```

AVERTISSEMENT

En particulier, assurez-vous qu'après **kdump**, le système ne redémarre pas. Si le système redémarre après que **kdump** n'a pas sauvegardé le fichier **vmcore**, il n'est pas possible d'invoquer le programme **sadump**.

2. Définissez le paramètre **failure_action** dans **/etc/kdump.conf** de manière appropriée comme **halt** ou **shell**.

```
failure_action shell
```

Ressources supplémentaires

- Manuel d'installation du serveur FUJITSU PRIMEQUEST 2000 Series

CHAPITRE 15. ANALYSE D'UN CORE DUMP

Pour déterminer la cause de la panne du système, vous pouvez utiliser l'utilitaire **crash** qui fournit une invite interactive très similaire au débogueur GNU (GDB). Cet utilitaire vous permet d'analyser de manière interactive un core dump créé par **kdump**, **netdump**, **diskdump** ou **xendump** ainsi qu'un système Linux en cours d'exécution. Vous pouvez également utiliser Kernel Oops Analyzer ou l'outil Kdump Helper.

15.1. INSTALLATION DE L'UTILITAIRE DE CRASH

Installer l'outil **crash** pour obtenir la suite d'analyse de base.

Procédure

1. Activer les référentiels concernés :

```
# subscription-manager repos --enable baseos repository
```

```
# subscription-manager repos --enable appstream repository
```

```
# subscription-manager repos --enable rhel-8-for-x86_64-baseos-debug-rpms
```

2. Installez le paquetage **crash**:

```
# dnf install crash
```

3. Installez le paquetage **kernel-debuginfo**:

```
# dnf install kernel-debuginfo
```

Le paquet correspond à votre noyau en cours d'exécution et fournit les données nécessaires à l'analyse du dump.

Ressources supplémentaires

- Configuration des paramètres de base du système

15.2. EXÉCUTION ET SORTIE DE L'UTILITAIRE DE CRASH

Lancez l'utilitaire **crash** pour analyser la cause de la panne du système.

Conditions préalables

- Identifier le noyau en cours d'exécution (par exemple **5.14.0-1.el9.x86_64**).

Procédure

1. Pour lancer l'utilitaire **crash**, deux paramètres nécessaires doivent être transmis à la commande :

- Les informations de débogage (une image vmlinuz décompressée), par exemple **/usr/lib/debug/lib/modules/5.14.0-1.el9.x86_64/vmlinux**, sont fournies par le biais d'un paquetage **kernel-debuginfo** spécifique.

- Le fichier vmcore actuel, par exemple **/var/crash/127.0.0.1-2021-09-13-14:05:33/vmcore**
  La commande **crash** qui en résulte se présente alors comme suit :

```
# crash /usr/lib/debug/lib/modules/5.14.0-1.el9.x86_64/vmlinux /var/crash/127.0.0.1-2021-09-13-14:05:33/vmcore
```

  Utilisez la même version de *<kernel>* qui a été capturée par **kdump**.

Exemple 15.1. Exécution de l'utilitaire de crash

L'exemple suivant montre l'analyse d'un core dump créé le 13 septembre 2021 à 14:05 PM, utilisant le noyau 5.14.0-1.el9.x86_64.

```
...
WARNING: kernel relocated [202MB]: patching 90160 gdb minimal_symbol values

      KERNEL: /usr/lib/debug/lib/modules/5.14.0-1.el9.x86_64/vmlinux
    DUMPFILE: /var/crash/127.0.0.1-2021-09-13-14:05:33/vmcore  [PARTIAL DUMP]
        CPUS: 2
        DATE: Mon Sep 13 14:05:16 2021
      UPTIME: 01:03:57
LOAD AVERAGE: 0.00, 0.00, 0.00
       TASKS: 586
    NODENAME: localhost.localdomain
     RELEASE: 5.14.0-1.el9.x86_64
     VERSION: #1 SMP Wed Aug 29 11:51:55 UTC 2018
     MACHINE: x86_64  (2904 Mhz)
      MEMORY: 2.9 GB
       PANIC: "sysrq: SysRq : Trigger a crash"
         PID: 10635
     COMMAND: "bash"
        TASK: ffff8d6c84271800  [THREAD_INFO: ffff8d6c84271800]
         CPU: 1
       STATE: TASK_RUNNING (SYSRQ)

crash>
```

2. Pour quitter l'invite interactive et mettre fin à **crash**, tapez **exit** ou **q**.

Exemple 15.2. Quitter l'utilitaire de crash

```
crash> exit
~]#
```

NOTE

La commande **crash** peut également être utilisée comme un outil puissant pour le débogage d'un système réel. Toutefois, il convient de l'utiliser avec précaution afin de ne pas casser votre système.

Ressources supplémentaires

- Guide des redémarrages inattendus du système

15.3. AFFICHAGE DE DIVERS INDICATEURS DANS L'UTILITAIRE DE CRASH

L'utilitaire **crash** permet d'afficher divers indicateurs, tels qu'un tampon de messages du noyau, une trace rétrospective, l'état d'un processus, des informations sur la mémoire virtuelle et les fichiers ouverts.

Affichage de la mémoire tampon des messages

- Pour afficher le tampon de messages du noyau, tapez la commande **log** à l'invite interactive, comme indiqué dans l'exemple ci-dessous :

```
crash> log
... several lines omitted ...
EIP: 0060:[<c068124f>] EFLAGS: 00010096 CPU: 2
EIP is at sysrq_handle_crash+0xf/0x20
EAX: 00000063 EBX: 00000063 ECX: c09e1c8c EDX: 00000000
ESI: c0a09ca0 EDI: 00000286 EBP: 00000000 ESP: ef4dbf24
 DS: 007b ES: 007b FS: 00d8 GS: 00e0 SS: 0068
Process bash (pid: 5591, ti=ef4da000 task=f196d560 task.ti=ef4da000)
Stack:
 c068146b c0960891 c0968653 00000003 00000000 00000002 efade5c0 c06814d0
<0> fffffffb c068150f b7776000 f2600c40 c0569ec4 ef4dbf9c 00000002 b7776000
<0> efade5c0 00000002 b7776000 c0569e60 c051de50 ef4dbf9c f196d560 ef4dbfb4
Call Trace:
 [<c068146b>] ? __handle_sysrq+0xfb/0x160
 [<c06814d0>] ? write_sysrq_trigger+0x0/0x50
 [<c068150f>] ? write_sysrq_trigger+0x3f/0x50
 [<c0569ec4>] ? proc_reg_write+0x64/0xa0
 [<c0569e60>] ? proc_reg_write+0x0/0xa0
 [<c051de50>] ? vfs_write+0xa0/0x190
 [<c051e8d1>] ? sys_write+0x41/0x70
 [<c0409adc>] ? syscall_call+0x7/0xb
Code: a0 c0 01 0f b6 41 03 19 d2 f7 d2 83 e2 03 83 e0 cf c1 e2 04 09 d0 88 41 03 f3 c3 90 c7 05
c8 1b 9e c0 01 00 00 00 0f ae f8 89 f6 <c6> 05 00 00 00 00 01 c3 89 f6 8d bc 27 00 00 00 00 8d 50
d0 83
EIP: [<c068124f>] sysrq_handle_crash+0xf/0x20 SS:ESP 0068:ef4dbf24
CR2: 0000000000000000
```

Tapez **help log** pour plus d'informations sur l'utilisation de la commande.

NOTE

Le tampon de messages du noyau contient les informations les plus essentielles sur la panne du système et, en tant que tel, il est toujours déversé en premier dans le fichier **vmcore-dmesg.txt**. Ceci est utile lorsqu'une tentative d'obtenir le fichier **vmcore** complet a échoué, par exemple en raison d'un manque d'espace sur l'emplacement cible. Par défaut, **vmcore-dmesg.txt** est situé dans le répertoire **/var/crash/**.

Affichage d'une trace rétrospective

- Pour afficher la trace de la pile du noyau, utilisez la commande **bt**.

```
crash> bt
PID: 5591   TASK: f196d560  CPU: 2   COMMAND: "bash"
 #0 [ef4dbdcc] crash_kexec at c0494922
 #1 [ef4dbe20] oops_end at c080e402
 #2 [ef4dbe34] no_context at c043089d
 #3 [ef4dbe58] bad_area at c0430b26
 #4 [ef4dbe6c] do_page_fault at c080fb9b
 #5 [ef4dbee4] error_code (via page_fault) at c080d809
    EAX: 00000063  EBX: 00000063  ECX: c09e1c8c  EDX: 00000000  EBP: 00000000
    DS:  007b      ESI: c0a09ca0  ES:  007b      EDI: 00000286  GS:  00e0
    CS:  0060      EIP: c068124f  ERR: ffffffff  EFLAGS: 00010096
 #6 [ef4dbf18] sysrq_handle_crash at c068124f
 #7 [ef4dbf24] __handle_sysrq at c0681469
 #8 [ef4dbf48] write_sysrq_trigger at c068150a
 #9 [ef4dbf54] proc_reg_write at c0569ec2
#10 [ef4dbf74] vfs_write at c051de4e
#11 [ef4dbf94] sys_write at c051e8cc
#12 [ef4dbfb0] system_call at c0409ad5
    EAX: ffffffda  EBX: 00000001  ECX: b7776000  EDX: 00000002
    DS:  007b      ESI: 00000002  ES:  007b      EDI: b7776000
    SS:  007b      ESP: bfcb2088  EBP: bfcb20b4  GS:  0033
    CS:  0073      EIP: 00edc416  ERR: 00000004  EFLAGS: 00000246
```

Type **bt *<pid>*** pour afficher la trace d'un processus spécifique ou tapez **help bt** pour plus d'informations sur l'utilisation de **bt**.

Affichage de l'état d'un processus

- Pour afficher l'état des processus dans le système, utilisez la commande **ps**.

```
crash> ps
   PID    PPID  CPU   TASK    ST  %MEM     VSZ    RSS  COMM
>     0      0   0  c09dc560  RU   0.0       0      0  [swapper]
>     0      0   1  f7072030  RU   0.0       0      0  [swapper]
      0      0   2  f70a3a90  RU   0.0       0      0  [swapper]
>     0      0   3  f70ac560  RU   0.0       0      0  [swapper]
      1      0   1  f705ba90  IN   0.0    2828   1424  init
... several lines omitted ...
   5566      1   1  f2592560  IN   0.0   12876    784  auditd
   5567      1   2  ef427560  IN   0.0   12876    784  auditd
   5587   5132   0  f196d030  IN   0.0   11064   3184  sshd
>  5591   5587   2  f196d560  RU   0.0    5084   1648  bash
```

Utiliser **ps *<pid>*** pour afficher l'état d'un seul processus spécifique. Utilisez *help ps* pour plus d'informations sur l'utilisation de **ps**.

Affichage des informations sur la mémoire virtuelle

- Pour afficher des informations de base sur la mémoire virtuelle, tapez la commande **vm** à l'invite interactive.

```
crash> vm
PID: 5591   TASK: f196d560  CPU: 2   COMMAND: "bash"
  MM       PGD      RSS    TOTAL_VM
f19b5900  ef9c6000  1648k    5084k
  VMA       START      END    FLAGS  FILE
f1bb0310    242000    260000 8000875  /lib/ld-2.12.so
f26af0b8    260000    261000 8100871  /lib/ld-2.12.so
efbc275c    261000    262000 8100873  /lib/ld-2.12.so
efbc2a18    268000    3ed000 8000075  /lib/libc-2.12.so
efbc23d8    3ed000    3ee000 8000070  /lib/libc-2.12.so
efbc2888    3ee000    3f0000 8100071  /lib/libc-2.12.so
efbc2cd4    3f0000    3f1000 8100073  /lib/libc-2.12.so
efbc243c    3f1000    3f4000 100073
efbc28ec    3f6000    3f9000 8000075  /lib/libdl-2.12.so
efbc2568    3f9000    3fa000 8100071  /lib/libdl-2.12.so
efbc2f2c    3fa000    3fb000 8100073  /lib/libdl-2.12.so
f26af888    7e6000    7fc000 8000075  /lib/libtinfo.so.5.7
f26aff2c    7fc000    7ff000 8100073  /lib/libtinfo.so.5.7
efbc211c    d83000    d8f000 8000075  /lib/libnss_files-2.12.so
efbc2504    d8f000    d90000 8100071  /lib/libnss_files-2.12.so
efbc2950    d90000    d91000 8100073  /lib/libnss_files-2.12.so
f26afe00    edc000    edd000 4040075
f1bb0a18   8047000   8118000 8001875  /bin/bash
f1bb01e4   8118000   811d000 8101873  /bin/bash
f1bb0c70   811d000   8122000 100073
f26afae0   9fd9000   9ffa000 100073
... several lines omitted ...
```

Utilisez **vm *<pid>*** pour afficher des informations sur un seul processus spécifique, ou utilisez **help vm** pour plus d'informations sur l'utilisation de **vm**.

Affichage des fichiers ouverts

- Pour afficher des informations sur les fichiers ouverts, utilisez la commande **files**.

```
crash> files
PID: 5591   TASK: f196d560  CPU: 2   COMMAND: "bash"
ROOT: /    CWD: /root
 FD    FILE     DENTRY    INODE    TYPE  PATH
  0  f734f640  eedc2c6c  eecd6048  CHR   /pts/0
  1  efade5c0  eee14090  f00431d4  REG   /proc/sysrq-trigger
  2  f734f640  eedc2c6c  eecd6048  CHR   /pts/0
 10  f734f640  eedc2c6c  eecd6048  CHR   /pts/0
255  f734f640  eedc2c6c  eecd6048  CHR   /pts/0
```

Utiliser **files *<pid>*** pour afficher les fichiers ouverts par un seul processus sélectionné, ou utilisez **help files** pour plus d'informations sur l'utilisation de **files**.

15.4. UTILISATION DE L'ANALYSEUR D'ERREURS DU NOYAU

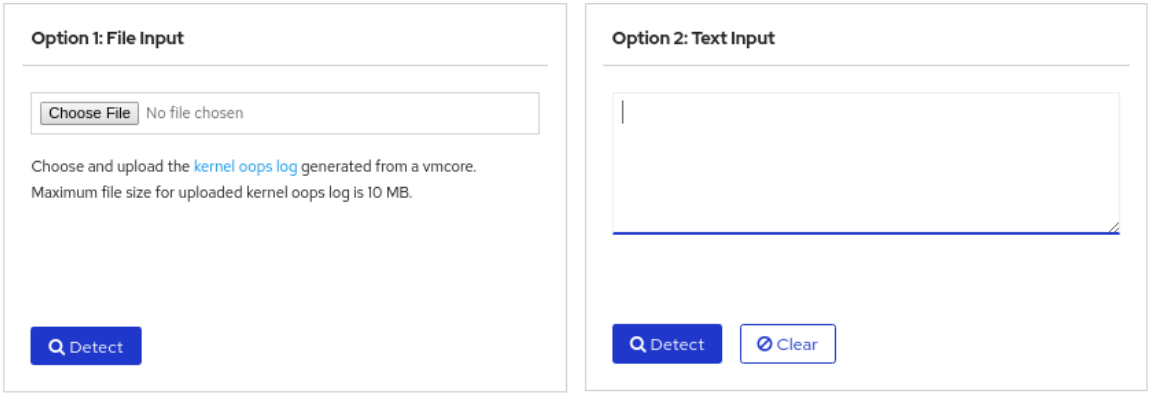
L'outil Kernel Oops Analyzer analyse le crash dump en comparant les messages oops avec les problèmes connus dans la base de connaissances.

Conditions préalables

- Sécuriser un message oops pour alimenter l'analyseur d'oops du noyau.

Procédure

1. Accédez à l'outil Kernel Oops Analyzer.
2. Pour diagnostiquer un problème de crash du noyau, téléchargez un journal des opérations du noyau généré à l'adresse **vmcore**.
   - Vous pouvez également diagnostiquer un problème de crash du noyau en fournissant un message texte ou une adresse **vmcore-dmesg.txt** comme entrée.



3. Cliquez sur **DETECT** pour comparer le message oops basé sur les informations de **makedumpfile** avec les solutions connues.

Ressources supplémentaires

- L' article Kernel Oops Analyzer
- Guide des redémarrages inattendus du système

15.5. L'OUTIL KDUMP HELPER

L'outil Kdump Helper permet de configurer le site **kdump** à l'aide des informations fournies. Kdump Helper génère un script de configuration basé sur vos préférences. L'exécution du script sur votre serveur permet de configurer le service **kdump**.

Ressources supplémentaires

- Aide Kdump

CHAPITRE 16. APPLICATION DE CORRECTIFS AVEC LE LIVE PATCHING DU NOYAU

Vous pouvez utiliser la solution de correction en direct du noyau de Red Hat Enterprise Linux pour corriger un noyau en cours d'exécution sans redémarrer ou relancer de processus.

Avec cette solution, les administrateurs de systèmes :

- Peut appliquer immédiatement des correctifs de sécurité critiques au noyau.
- Il n'est pas nécessaire d'attendre l'achèvement de tâches de longue haleine, la déconnexion des utilisateurs ou des temps d'arrêt programmés.
- Contrôlez davantage le temps de fonctionnement du système sans sacrifier la sécurité ou la stabilité.

Notez que tous les CVE critiques ou importants ne seront pas résolus à l'aide de la solution de correctifs en direct du noyau. Notre objectif est de réduire les redémarrages nécessaires pour les correctifs liés à la sécurité, et non de les éliminer complètement. Pour plus de détails sur le champ d'application des correctifs en direct, voir le fichier *Customer Portal Solutions article* .

AVERTISSEMENT

Certaines incompatibilités existent entre le live patching du noyau et d'autres sous-composants du noyau. Lire le

Limites de kpatch avant d'utiliser le live patching du noyau.

16.1. LIMITES DE KPATCH

- La fonction **kpatch** n'est pas un mécanisme général de mise à jour du noyau. Elle est utilisée pour appliquer des mises à jour simples de sécurité et de correction de bogues lorsque le redémarrage du système n'est pas possible dans l'immédiat.
- N'utilisez pas les outils **SystemTap** ou **kprobe** pendant ou après le chargement d'un correctif. Le correctif pourrait ne pas prendre effet tant que ces sondes n'ont pas été supprimées.

16.2. PRISE EN CHARGE DES CORRECTIFS EN DIRECT DE TIERS

L'utilitaire **kpatch** est le seul utilitaire de correction en direct du noyau pris en charge par Red Hat avec les modules RPM fournis par les dépôts Red Hat. Red Hat ne supportera aucun correctif en direct qui n'a pas été fourni par Red Hat lui-même.

Si vous avez besoin d'assistance pour un problème qui survient avec un correctif tiers, Red Hat vous recommande d'ouvrir un dossier avec le fournisseur du correctif au début de toute investigation dans laquelle une détermination de la cause première est nécessaire. Cela permet de fournir le code source si le vendeur le permet, et à son organisation de support de fournir une assistance dans la détermination de la cause première avant d'escalader l'enquête vers le support de Red Hat.

Pour tout système fonctionnant avec des correctifs tiers, Red Hat se réserve le droit de demander une

reproduction avec les logiciels fournis et pris en charge par Red Hat. Dans le cas où cela n'est pas possible, nous demandons qu'un système et une charge de travail similaires soient déployés sur votre environnement de test sans application de correctifs réels, afin de confirmer si le même comportement est observé.

Pour plus d'informations sur les politiques de prise en charge des logiciels tiers, voir Comment Red Hat Global Support Services gère-t-il les logiciels tiers, les pilotes et/ou le matériel/hyperviseurs non certifiés ou les systèmes d'exploitation invités ?

16.3. ACCÈS AUX CORRECTIFS DU NOYAU

La capacité de correction en direct du noyau est mise en œuvre sous la forme d'un module du noyau (**kmod**) livré sous la forme d'un paquetage RPM.

Tous les clients ont accès aux correctifs du noyau, qui sont livrés par les canaux habituels. Toutefois, les clients qui ne souscrivent pas à une offre de support étendu perdront l'accès aux nouveaux correctifs pour la version mineure actuelle dès que la version mineure suivante sera disponible. Par exemple, les clients ayant souscrit un abonnement standard ne pourront bénéficier de correctifs en direct pour le noyau RHEL 9.1 que jusqu'à la sortie du noyau RHEL 9.2.

16.4. COMPOSANTS DU CORRECTIF EN DIRECT DU NOYAU

Les composants du live patching du noyau sont les suivants :

Module de correction du noyau

- Mécanisme de distribution des correctifs du noyau.
- Un module de noyau qui est construit spécifiquement pour le noyau à corriger.
- Le module patch contient le code des correctifs souhaités pour le noyau.
- Les modules de correction s'enregistrent auprès du sous-système du noyau **livepatch** et fournissent des informations sur les fonctions originales à remplacer, ainsi que des pointeurs correspondants vers les fonctions de remplacement. Les modules de correction du noyau sont livrés sous forme de RPM.
- La convention d'appellation est la suivante : **kpatch_<kernel version>_<kpatch version>_<kpatch release>**. La partie "version du noyau" du nom a été remplacée par *underscores* à la place de *dots*.

L'utilitaire kpatch

Un utilitaire de ligne de commande pour la gestion des modules de correctifs.

Le service kpatch

Un service **systemd** requis par **multiuser.target**. Cette cible charge le module de correction du noyau au démarrage.

Le paquet kpatch-dnf

Un plugin DNF livré sous la forme d'un paquetage RPM. Ce plugin gère l'abonnement automatique aux correctifs du noyau.

16.5. COMMENT FONCTIONNE LE LIVE PATCHING DU NOYAU

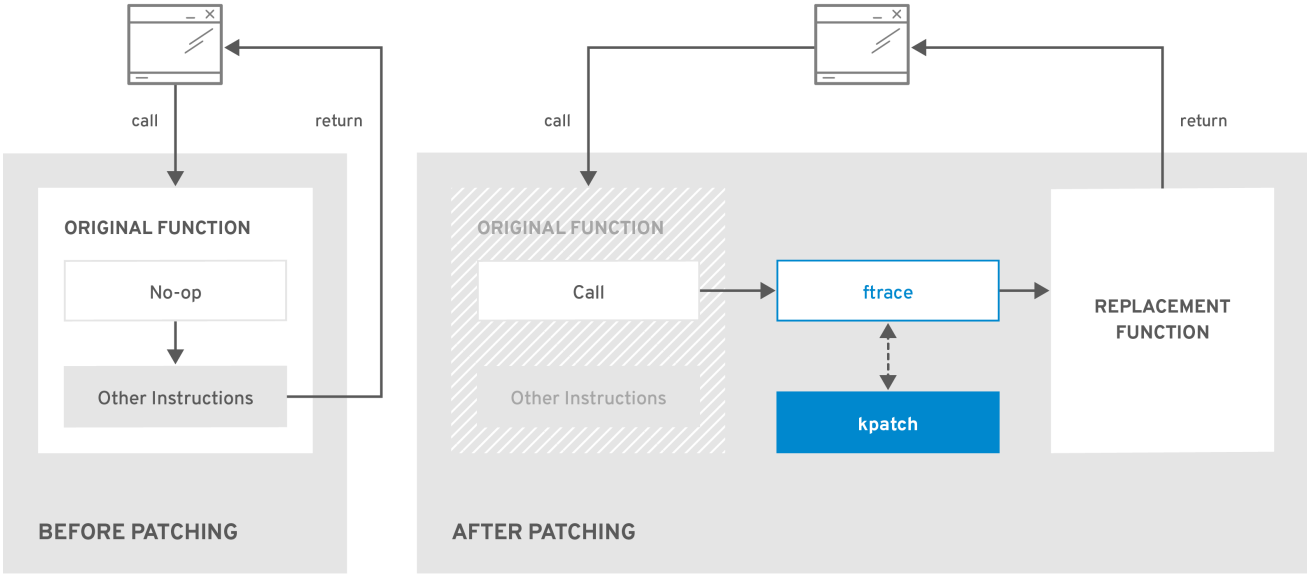
La solution de correction du noyau **kpatch** utilise le sous-système du noyau **livepatch** pour rediriger les anciennes fonctions vers les nouvelles. Lorsqu'un correctif du noyau est appliqué à un système, les choses suivantes se produisent :

1. Le module de correction du noyau est copié dans le répertoire **/var/lib/kpatch/** et enregistré pour être réappliqué au noyau par **systemd** lors du prochain démarrage.

2. Le module kpatch est chargé dans le noyau en cours d'exécution et les nouvelles fonctions sont enregistrées dans le mécanisme **ftrace** avec un pointeur sur l'emplacement en mémoire du nouveau code.

3. Lorsque le noyau accède à la fonction corrigée, il est redirigé par le mécanisme **ftrace** qui contourne les fonctions d'origine et redirige le noyau vers la version corrigée de la fonction.

Figure 16.1. Comment fonctionne le live patching du noyau



16.6. ABONNEMENT DES NOYAUX ACTUELLEMENT INSTALLÉS AU FLUX DE CORRECTIFS EN DIRECT

Un module de correction du noyau est livré dans un paquet RPM, spécifique à la version du noyau faisant l'objet de la correction. Chaque paquet RPM sera mis à jour de manière cumulative au fil du temps.

La procédure suivante explique comment s'abonner à toutes les futures mises à jour cumulatives des correctifs en direct pour un noyau donné. Les correctifs en direct étant cumulatifs, vous ne pouvez pas sélectionner les correctifs individuels déployés pour un noyau donné.

AVERTISSEMENT

Red Hat ne prend pas en charge les correctifs de tiers appliqués à un système pris en charge par Red Hat.

Conditions préalables

- Autorisations de la racine

Procédure

1. Si vous le souhaitez, vérifiez la version de votre noyau :

```
# uname -r
5.14.0-1.el9.x86_64
```

2. Recherchez un paquet de correctifs correspondant à la version de votre noyau :

```
# dnf search $(uname -r)
```

3. Installer le paquet de correctifs en direct :

```
# dnf install "kpatch-patch = $(uname -r)"
```

La commande ci-dessus installe et applique les derniers correctifs cumulatifs pour ce noyau spécifique uniquement.

Si la version d'un paquet de correctifs en direct est 1-1 ou supérieure, le paquet contiendra un module de correctifs. Dans ce cas, le noyau sera automatiquement corrigé lors de l'installation du paquet de correctifs.

Le module de correction du noyau est également installé dans le répertoire **/var/lib/kpatch/** afin d'être chargé par le système **systemd** et le gestionnaire de services lors des prochains redémarrages.

NOTE

Un paquet de live patching vide sera installé lorsqu'il n'y a pas de live patches disponibles pour un noyau donné. Un paquet de live patching vide aura un *kpatch_version-kpatch_release* de 0-0, par exemple **kpatch-patch-5_14_0-1-0-0.x86_64.rpm**. L'installation du RPM vide abonne le système à tous les futurs live patches pour le noyau donné.

4. Optionnellement, vérifiez que le noyau est corrigé :

```
# kpatch list
Loaded patch modules:
kpatch_5_14_0_1_0_1 [enabled]

Installed patch modules:
kpatch_5_14_0_1_0_1 (5.14.0-1.el9.x86_64)
…
```

La sortie montre que le module de correction du noyau a été chargé dans le noyau, qui est maintenant corrigé avec les derniers correctifs du paquet **kpatch-patch-5_14_0-1-0-1.el9.x86_64.rpm**.

Ressources supplémentaires

- **kpatch(1)** page du manuel
- *Configuring basic system settings* dans RHEL

16.7. INSCRIPTION AUTOMATIQUE DE TOUT FUTUR NOYAU AU FLUX DE CORRECTIFS EN DIRECT

Vous pouvez utiliser le plugin **kpatch-dnf** DNF pour abonner votre système aux correctifs fournis par le module de correctifs du noyau, également connus sous le nom de "kernel live patches". Le plugin permet l'abonnement à **automatic** pour tout noyau que le système utilise actuellement, ainsi que pour les noyaux **to-be-installed in the future**

Conditions préalables

- Vous disposez des droits d'accès à la racine.

Procédure

1. Optionnellement, vérifiez tous les noyaux installés et le noyau que vous utilisez actuellement :

```
# dnf list installed | grep kernel
Updating Subscription Management repositories.
Installed Packages
...
kernel-core.x86_64          5.14.0-1.el9          @beaker-BaseOS
kernel-core.x86_64          5.14.0-2.el9          @@commandline
...

# uname -r
5.14.0-2.el9.x86_64
```

2. Installez le plugin **kpatch-dnf**:

```
# dnf install kpatch-dnf
```

3. Activer l'abonnement automatique aux correctifs du noyau :

```
# dnf kpatch auto
Updating Subscription Management repositories.
Last metadata expiration check: 1:38:21 ago on Fri 17 Sep 2021 07:29:53 AM EDT.
Dependencies resolved.
================================================
 Package                    Architecture
================================================
Installing:
 kpatch-patch-5_14_0-1      x86_64
 kpatch-patch-5_14_0-2      x86_64

Transaction Summary
================================================
Install  2 Packages
…
```

Cette commande permet d'abonner tous les noyaux actuellement installés à la réception des correctifs en direct du noyau. Elle installe et applique également les derniers correctifs cumulatifs, le cas échéant, pour tous les noyaux installés.

À l'avenir, lorsque vous mettrez à jour le noyau, les correctifs seront automatiquement installés au cours de la procédure d'installation du nouveau noyau.

Le module de correction du noyau est également installé dans le répertoire **/var/lib/kpatch/** afin d'être chargé par le système **systemd** et le gestionnaire de services lors des prochains redémarrages.

NOTE

Un paquet de live patching vide sera installé lorsqu'il n'y a pas de live patches disponibles pour un noyau donné. Un paquet de live patching vide aura un *kpatch_version-kpatch_release* de 0-0, par exemple **kpatch-patch-5_14_0-1-0-0.el9.x86_64.rpm** L'installation du RPM vide abonne le système à tous les futurs live patches pour le noyau donné.

Étape de vérification

- Vérifiez que tous les noyaux installés ont été corrigés :

```
# kpatch list
Loaded patch modules:
kpatch_5_14_0_2_0_1 [enabled]

Installed patch modules:
kpatch_5_14_0_1_0_1 (5.14.0-1.el9.x86_64)
kpatch_5_14_0_2_0_1 (5.14.0-2.el9.x86_64)
```

La sortie montre que le noyau que vous utilisez et l'autre noyau installé ont été corrigés avec des correctifs provenant respectivement des paquets **kpatch-patch-5_14_0-1-0-1.el9.x86_64.rpm** et **kpatch-patch-5_14_0-2-0-1.el9.x86_64.rpm**.

Ressources supplémentaires

- **kpatch(1)** et **dnf-kpatch(8)** pages de manuel
- *Configuring basic system settings* dans RHEL

16.8. DÉSACTIVATION DE L'ABONNEMENT AUTOMATIQUE AU FLUX DE CORRECTIFS EN DIRECT

Lorsque vous abonnez votre système aux correctifs fournis par le module de correctifs du noyau, votre abonnement est **automatic**. Vous pouvez désactiver cette fonction, et donc l'installation automatique des paquets **kpatch-patch**.

Conditions préalables

- Vous disposez des droits d'accès à la racine.

Procédure

1. Optionnellement, vérifiez tous les noyaux installés et le noyau que vous utilisez actuellement :

```
# dnf list installed | grep kernel
Updating Subscription Management repositories.
Installed Packages
...
kernel-core.x86_64          5.14.0-1.el9          @beaker-BaseOS
kernel-core.x86_64          5.14.0-2.el9          @@commandline
...

# uname -r
5.14.0-2.el9.x86_64
```

2. Désactiver l'abonnement automatique aux correctifs du noyau :

```
# dnf kpatch manual
Updating Subscription Management repositories.
```

Étape de vérification

- Vous pouvez vérifier si le résultat est satisfaisant :

```
# yum kpatch status
...
Updating Subscription Management repositories.
Last metadata expiration check: 0:30:41 ago on Tue Jun 14 15:59:26 2022.
Kpatch update setting: manual
```

Ressources supplémentaires

- **kpatch(1)** et **dnf-kpatch(8)** pages de manuel

16.9. MISE À JOUR DES MODULES DE CORRECTION DU NOYAU

Étant donné que les modules de correctifs du noyau sont livrés et appliqués par le biais de paquets RPM, la mise à jour d'un module de correctifs cumulatifs du noyau est identique à celle de n'importe quel autre paquet RPM.

Conditions préalables

- Le système est abonné au flux de correctifs en direct, comme décrit dans la section Abonnement des noyaux actuellement installés au flux de correctifs en direct .

Procédure

- Mise à jour vers une nouvelle version cumulative pour le noyau actuel :

```
# dnf update "kpatch-patch = $(uname -r)"
```

La commande ci-dessus installe et applique automatiquement toutes les mises à jour disponibles pour le noyau en cours d'exécution. Y compris les correctifs cumulatifs live qui seront publiés ultérieurement.

- Il est également possible de mettre à jour tous les modules de correction du noyau installés :

```
# dnf update "kpatch-patch"
```

NOTE

Lorsque le système redémarre avec le même noyau, celui-ci est automatiquement corrigé par le service **kpatch.service** systemd.

Ressources supplémentaires

- *Configuring basic system settings* dans RHEL

16.10. SUPPRESSION DU PAQUET DE CORRECTIFS EN DIRECT

Désactivez la solution de correctifs en direct du noyau de Red Hat Enterprise Linux en supprimant le paquetage de correctifs en direct.

Conditions préalables

- Autorisations de la racine
- Le paquet de correctifs en direct est installé.

Procédure

1. Sélectionnez le paquet de correctifs en direct.

```
# dnf list installed | grep kpatch-patch
kpatch-patch-5_14_0-1.x86_64        0-1.el9        @@commandline
…
```

   L'exemple ci-dessus présente la liste des paquets de correctifs que vous avez installés.

2. Supprimez le paquet de correctifs en direct.

```
# dnf remove kpatch-patch-5_14_0-1.x86_64
```

   Lorsqu'un paquet de correctifs en direct est supprimé, le noyau reste corrigé jusqu'au prochain redémarrage, mais le module de correctifs du noyau est supprimé du disque. Au prochain redémarrage, le noyau correspondant ne sera plus patché.

3. Redémarrez votre système.

4. Vérifiez que le paquet de correctifs en direct a été supprimé.

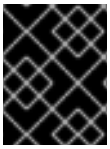
```
# dnf list installed | grep kpatch-patch
```

   La commande n'affiche aucun résultat si le paquet a été supprimé avec succès.

5. En option, vérifiez que la solution de correctifs en direct du noyau est désactivée.

```
# kpatch list
Loaded patch modules:
```

L'exemple de sortie montre que le noyau n'est pas patché et que la solution de patching en direct n'est pas active parce qu'il n'y a pas de modules de patching actuellement chargés.



IMPORTANT

Actuellement, Red Hat ne prend pas en charge la réversion des correctifs en direct sans redémarrer votre système. En cas de problème, contactez notre équipe d'assistance.

Ressources supplémentaires

- La page du manuel **kpatch(1)**
- *Configuring basic system settings* dans RHEL

16.11. DÉSINSTALLATION DU MODULE DE CORRECTION DU NOYAU

Empêcher la solution de correction en direct du noyau de Red Hat Enterprise Linux d'appliquer un module de correction du noyau lors des démarrages suivants.

Conditions préalables

- Autorisations de la racine
- Un paquet de correctifs en direct est installé.
- Un module de correction du noyau est installé et chargé.

Procédure

1. Sélectionnez un module de correction du noyau :

```
# kpatch list
Loaded patch modules:
kpatch_5_14_0_1_0_1 [enabled]

Installed patch modules:
kpatch_5_14_0_1_0_1 (5.14.0-1.el9.x86_64)
…
```

2. Désinstaller le module de correction du noyau sélectionné.

```
# kpatch uninstall kpatch_5_14_0_1_0_1
uninstalling kpatch_5_14_0_1_0_1 (5.14.0-1.el9.x86_64)
```

- Notez que le module de correction du noyau désinstallé est toujours chargé :

```
# kpatch list
Loaded patch modules:
kpatch_5_14_0_1_0_1 [enabled]
```

```
Installed patch modules:
<NO_RESULT>
```

Lorsque le module sélectionné est désinstallé, le noyau reste corrigé jusqu'au prochain redémarrage, mais le module de correction du noyau est supprimé du disque.

3. Redémarrez votre système.
4. En option, vérifiez que le module de correction du noyau a été désinstallé.

```
# kpatch list
Loaded patch modules:
…
```

L'exemple ci-dessus ne montre aucun module de correction du noyau chargé ou installé. Le noyau n'est donc pas corrigé et la solution de correction en direct du noyau n'est pas active.

IMPORTANT

Actuellement, Red Hat ne prend pas en charge la réversion des correctifs en direct sans redémarrer votre système. En cas de problème, contactez notre équipe d'assistance.

Ressources supplémentaires

- La page du manuel **kpatch(1)**

16.12. DÉSACTIVATION DE KPATCH.SERVICE

Empêcher la solution de correctifs en direct du noyau de Red Hat Enterprise Linux d'appliquer globalement tous les modules de correctifs du noyau lors des démarrages suivants.

Conditions préalables

- Autorisations de la racine
- Un paquet de correctifs en direct est installé.
- Un module de correction du noyau est installé et chargé.

Procédure

1. Vérifiez que **kpatch.service** est activé.

```
# systemctl is-enabled kpatch.service
enabled
```

2. Désactiver **kpatch.service**:

```
# systemctl disable kpatch.service
Removed /etc/systemd/system/multi-user.target.wants/kpatch.service.
```

- Notez que le module de correction du noyau appliqué est toujours chargé :

```
# kpatch list
Loaded patch modules:
kpatch_5_14_0_1_0_1 [enabled]

Installed patch modules:
kpatch_5_14_0_1_0_1 (5.14.0-1.el9.x86_64)
```

3. Redémarrez votre système.

4. Optionnellement, vérifiez l'état de **kpatch.service**.

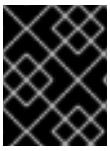
```
# systemctl status kpatch.service
● kpatch.service - "Apply kpatch kernel patches"
  Loaded: loaded (/usr/lib/systemd/system/kpatch.service; disabled; vendor preset: disabled)
  Active: inactive (dead)
```

L'exemple de sortie indique que **kpatch.service** a été désactivé et n'est pas en cours d'exécution. Par conséquent, la solution de correctifs en direct du noyau n'est pas active.

5. Vérifiez que le module de correction du noyau a été déchargé.

```
# kpatch list
Loaded patch modules:

Installed patch modules:
kpatch_5_14_0_1_0_1 (5.14.0-1.el9.x86_64)
```

L'exemple ci-dessus montre qu'un module de correction du noyau est toujours installé mais que le noyau n'est pas corrigé.



IMPORTANT

Actuellement, Red Hat ne prend pas en charge la réversion des correctifs en direct sans redémarrer votre système. En cas de problème, contactez notre équipe d'assistance.

Ressources supplémentaires

- La page du manuel **kpatch(1)**
- *Configuring basic system settings* dans RHEL

CHAPITRE 17. UTILISER SYSTEMD POUR GÉRER LES RESSOURCES UTILISÉES PAR LES APPLICATIONS

RHEL 9 déplace les paramètres de gestion des ressources du niveau du processus au niveau de l'application en liant le système de hiérarchies **cgroup** à l'arborescence d'unités **systemd**. Par conséquent, vous pouvez gérer les ressources du système à l'aide de la commande **systemctl** ou en modifiant les fichiers d'unité **systemd**.

Pour ce faire, **systemd** prend diverses options de configuration dans les fichiers unitaires ou directement via la commande **systemctl**. Ensuite, **systemd** applique ces options à des groupes de processus spécifiques en utilisant les appels système du noyau Linux et des fonctions telles que **cgroups** et **namespaces**.

NOTE

Vous pouvez consulter l'ensemble des options de configuration pour **systemd** dans les pages suivantes du manuel :

- **systemd.resource-control(5)**
- **systemd.exec(5)**

17.1. ALLOCATION DES RESSOURCES SYSTÈME À L'AIDE DE SYSTEMD

Pour modifier la distribution des ressources du système, vous pouvez appliquer un ou plusieurs des modèles de distribution suivants :

Poids

Vous pouvez distribuer la ressource en additionnant les poids de tous les sous-groupes et en donnant à chaque sous-groupe la fraction correspondant à son ratio par rapport à la somme. Par exemple, si vous avez 10 cgroups, chacun avec un poids de valeur 100, la somme est de 1000. Chaque cgroup reçoit un dixième de la ressource.

Le poids est généralement utilisé pour distribuer des ressources sans état. Par exemple, l'option *CPUWeight=* est une implémentation de ce modèle de distribution des ressources.

Limites

Un cgroup peut consommer jusqu'à la quantité configurée de la ressource. La somme des limites des sous-groupes peut dépasser la limite du cgroupe parent. Il est donc possible de surcharger les ressources dans ce modèle.
Par exemple, l'option *MemoryMax=* est une mise en œuvre de ce modèle de distribution des ressources.

Protections

Vous pouvez définir une quantité protégée d'une ressource pour un cgroup. Si l'utilisation de la ressource est inférieure à la limite de protection, le noyau essaiera de ne pas pénaliser ce cgroup en faveur d'autres cgroups qui sont en concurrence pour la même ressource. Un surengagement est également possible.
Par exemple, l'option *MemoryLow=* est une mise en œuvre de ce modèle de distribution des ressources.

Allocations

Allocations exclusives d'une quantité absolue d'une ressource finie. Un surengagement n'est pas possible. Un exemple de ce type de ressource sous Linux est le budget temps réel.

option de fichier d'unité

Paramètre de configuration du contrôle des ressources.
Par exemple, vous pouvez configurer la ressource CPU avec des options telles que *CPUAccounting=* ou *CPUQuota=*. De même, vous pouvez configurer la mémoire ou les ressources E/S avec des options telles que *AllowedMemoryNodes=* et *IOAccounting=*.

Procédure

Pour modifier la valeur requise de l'option du fichier d'unités de votre service, vous pouvez ajuster la valeur dans le fichier d'unités ou utiliser la commande **systemctl**:

1. Vérifiez les valeurs attribuées pour le service de votre choix.

```
# systemctl show --propriété <unit file option> <service name>
```

2. Définir la valeur requise de l'option de politique d'allocation du temps CPU :

```
# systemctl set-property <service name> <unit file option> =<value>
```

Verification steps

- Vérifiez les valeurs nouvellement attribuées pour le service de votre choix.

```
# systemctl show --propriété <unit file option> <service name>
```

Ressources supplémentaires

- **systemd.resource-control(5)**, **systemd.exec(5)** pages de manuel

17.2. RÔLE DE SYSTEMD DANS LA GESTION DES RESSOURCES

La fonction principale de **systemd** est la gestion et la supervision des services. Le gestionnaire du système et des services **systemd** veille à ce que les services gérés démarrent au bon moment et dans le bon ordre pendant le processus de démarrage. Les services doivent fonctionner sans heurts pour utiliser de manière optimale la plate-forme matérielle sous-jacente. C'est pourquoi **systemd** fournit également des fonctionnalités permettant de définir des politiques de gestion des ressources et de régler diverses options susceptibles d'améliorer les performances du service.

IMPORTANT

En général, Red Hat vous recommande d'utiliser **systemd** pour contrôler l'utilisation des ressources du système. Vous ne devez configurer manuellement le système de fichiers virtuel **cgroups** que dans des cas particuliers. Par exemple, lorsque vous devez utiliser des contrôleurs **cgroup-v1** qui n'ont pas d'équivalents dans la hiérarchie **cgroup-v2**.

17.3. VUE D'ENSEMBLE DE LA HIÉRARCHIE DE SYSTEMD POUR LES CGROUPS

En arrière-plan, le gestionnaire de systèmes et de services **systemd** utilise les unités **slice**, **scope** et **service** pour organiser et structurer les processus dans les groupes de contrôle. Vous pouvez modifier cette hiérarchie en créant des fichiers d'unités personnalisés ou en utilisant la commande **systemctl**. En outre, **systemd** monte automatiquement les hiérarchies pour les contrôleurs de ressources importants du noyau dans le répertoire /**sys**/**fs**/**cgroup**/.

Trois types d'unités **systemd** sont utilisés pour le contrôle des ressources :

- **Service** - Un processus ou un groupe de processus, qui **systemd** démarré selon un fichier de configuration d'unité. Les services encapsulent les processus spécifiés afin qu'ils puissent être démarrés et arrêtés en tant qu'ensemble. Les services sont nommés de la manière suivante :

  ```
  <name>.service
  ```

- **Scope** - Un groupe de processus créés de l'extérieur. Les portées encapsulent les processus qui sont démarrés et arrêtés par les processus arbitraires via la fonction **fork()**, puis enregistrés par **systemd** au moment de l'exécution. Par exemple, les sessions utilisateur, les conteneurs et les machines virtuelles sont traités comme des portées. Les champs d'application sont nommés comme suit :

  ```
  <name>.scope
  ```

- **Slice** - Un groupe d'unités organisées hiérarchiquement. Les tranches organisent une hiérarchie dans laquelle sont placés les champs d'application et les services. Les processus réels sont contenus dans les scopes ou dans les services. Chaque nom d'une unité de tranche correspond au chemin d'accès à un emplacement dans la hiérarchie. Le tiret ("-") sert de séparateur entre les composants du chemin d'accès à une tranche et la tranche racine **-.slice**. Dans l'exemple suivant :

  ```
  <parent-name>.slice
  ```

  parent-name.slice est une sous-tranche de **parent.slice**, qui est une sous-tranche de la tranche racine **-.slice**. **parent-name.slice** peut avoir sa propre sous-tranche nommée **parent-name-name2.slice**, et ainsi de suite.

Les unités **service**, **scope** et **slice** sont directement associées à des objets dans la hiérarchie du groupe de contrôle. Lorsque ces unités sont activées, elles correspondent directement aux chemins des groupes de contrôle construits à partir des noms des unités.

Voici un exemple abrégé de la hiérarchie d'un groupe de contrôle :

```
Control group /:
-.slice
├─user.slice
│ ├─user-42.slice
│ │ ├─session-c1.scope
│ │ │ ├─ 967 gdm-session-worker [pam/gdm-launch-environment]
│ │ │ ├─1035 /usr/libexec/gdm-x-session gnome-session --autostart
/usr/share/gdm/greeter/autostart
│ │ │ ├─1054 /usr/libexec/Xorg vt1 -displayfd 3 -auth /run/user/42/gdm/Xauthority -background none
-noreset -keeptty -verbose 3
│ │ │ ├─1212 /usr/libexec/gnome-session-binary --autostart /usr/share/gdm/greeter/autostart
│ │ │ ├─1369 /usr/bin/gnome-shell
│ │ │ ├─1732 ibus-daemon --xim --panel disable
│ │ │ ├─1752 /usr/libexec/ibus-dconf
```

```
  │ │ │ ├─1762 /usr/libexec/ibus-x11 --kill-daemon
  │ │ │ ├─1912 /usr/libexec/gsd-xsettings
  │ │ │ ├─1917 /usr/libexec/gsd-a11y-settings
  │ │ │ ├─1920 /usr/libexec/gsd-clipboard
  …
  ├─init.scope
  │ └─1 /usr/lib/systemd/systemd --switched-root --system --deserialize 18
  └─system.slice
    ├─rngd.service
    │ └─800 /sbin/rngd -f
    ├─systemd-udevd.service
    │ └─659 /usr/lib/systemd/systemd-udevd
    ├─chronyd.service
    │ └─823 /usr/sbin/chronyd
    ├─auditd.service
    │ ├─761 /sbin/auditd
    │ └─763 /usr/sbin/sedispatch
    ├─accounts-daemon.service
    │ └─876 /usr/libexec/accounts-daemon
    ├─example.service
    │ ├─ 929 /bin/bash /home/jdoe/example.sh
    │ └─4902 sleep 1
    …
```

L'exemple ci-dessus montre que les services et les champs d'application contiennent des processus et sont placés dans des tranches qui ne contiennent pas de processus propres.

Ressources supplémentaires

- *Configuring basic system settings* dans Red Hat Enterprise Linux
- Que sont les contrôleurs de ressources du noyau ?
- **systemd.resource-control(5)**pages du manuel, **systemd.exec(5)**, **cgroups(7)**, **fork()**, **fork(2)**
- Comprendre les cgroups

17.4. LISTE DES UNITÉS SYSTEMD

Utilisez le système **systemd** et le gestionnaire de services pour dresser la liste de ses unités.

Procédure

- Dressez la liste de toutes les unités actives du système à l'aide de la commande **# systemctl**. Le terminal renverra une sortie similaire à l'exemple suivant :

```
# systemctl
UNIT                                LOAD   ACTIVE SUB       DESCRIPTION
…
init.scope                          loaded active running   System and Service Manager
session-2.scope                     loaded active running   Session 2 of user jdoe
abrt-ccpp.service                   loaded active exited    Install ABRT coredump hook
abrt-oops.service                   loaded active running   ABRT kernel log watcher
abrt-vmcore.service                 loaded active exited    Harvest vmcores for ABRT
abrt-xorg.service                   loaded active running   ABRT Xorg log watcher
```

```
…
-.slice                                    loaded active active    Root Slice
machine.slice                              loaded active active    Virtual Machine and Container
Slice system-getty.slice                                           loaded active active
system-getty.slice
system-lvm2\x2dpvscan.slice                loaded active active    system-
lvm2\x2dpvscan.slice
system-sshd\x2dkeygen.slice                loaded active active    system-
sshd\x2dkeygen.slice
system-systemd\x2dhibernate\x2dresume.slice        loaded active active    system-
systemd\x2dhibernate\x2dresume>
system-user\x2druntime\x2ddir.slice            loaded active active    system-
user\x2druntime\x2ddir.slice
system.slice                               loaded active active    System Slice
user-1000.slice                            loaded active active    User Slice of UID 1000
user-42.slice                              loaded active active    User Slice of UID 42
user.slice                                 loaded active active    User and Session Slice
…
```

- **UNIT** - un nom d'unité qui reflète également la position de l'unité dans la hiérarchie d'un groupe de contrôle. Les unités pertinentes pour le contrôle des ressources sont *slice*, *scope* et *service*.
- **LOAD** - indique si le fichier de configuration de l'unité a été correctement chargé. Si le fichier de l'unité n'a pas été chargé, le champ contient l'état *error* au lieu de *loaded*. Les autres états de chargement de l'unité sont les suivants : *stub*, *merged*, et *masked*.
- **ACTIVE** - l'état d'activation de l'unité de haut niveau, qui est une généralisation de **SUB**.
- **SUB** - l'état d'activation de l'unité de bas niveau. Les valeurs possibles dépendent du type d'unité.
- **DESCRIPTION** - la description du contenu et de la fonctionnalité de l'unité.

- Liste des unités inactives.

```
# systemctl --all
```

- Limiter la quantité d'informations dans le résultat.

```
# systemctl --type service,masked
```

L'option **--type** requiert une liste de types d'unités séparés par des virgules, tels que *service* et *slice*, ou d'états de charge des unités, tels que *loaded* et *masked*.

Ressources supplémentaires

- *Configuring basic system settings* dans RHEL
- Les pages du manuel **systemd.resource-control(5)**, **systemd.exec(5)**

17.5. VISUALISATION DE LA HIÉRARCHIE DES GROUPES DE CONTRÔLE DE SYSTEMD

Afficher la hiérarchie des groupes de contrôle (**cgroups**) et les processus en cours d'exécution dans un site spécifique **cgroups**.

Procédure

- Affichez l'ensemble de la hiérarchie **cgroups** sur votre système à l'aide de la commande **systemd-cgls**.

```
# systemd-cgls
Control group /:
-.slice
├─user.slice
│ ├─user-42.slice
│ │ ├─session-c1.scope
│ │ │ ├─ 965 gdm-session-worker [pam/gdm-launch-environment]
│ │ │ ├─1040 /usr/libexec/gdm-x-session gnome-session --autostart
/usr/share/gdm/greeter/autostart
…
├─init.scope
│ └─1 /usr/lib/systemd/systemd --switched-root --system --deserialize 18
└─system.slice
  …
  ├─example.service
  │ ├─6882 /bin/bash /home/jdoe/example.sh
  │ └─6902 sleep 1
  ├─systemd-journald.service
  │ └─629 /usr/lib/systemd/systemd-journald
  …
```

L'exemple de sortie renvoie l'ensemble de la hiérarchie **cgroups**, dont le niveau le plus élevé est formé par *slices*.

- Affichez la hiérarchie **cgroups** filtrée par un contrôleur de ressources avec la commande **systemd-cgls <*resource_controller*>** commande.

```
# systemd-cgls memory
Controller memory; Control group /:
├─1 /usr/lib/systemd/systemd --switched-root --system --deserialize 18
├─user.slice
│ ├─user-42.slice
│ │ ├─session-c1.scope
│ │ │ ├─ 965 gdm-session-worker [pam/gdm-launch-environment]
…
└─system.slice
  |
  …
  ├─chronyd.service
  │ └─844 /usr/sbin/chronyd
  ├─example.service
  │ ├─8914 /bin/bash /home/jdoe/example.sh
  │ └─8916 sleep 1
  …
```

L'exemple de sortie de la commande ci-dessus liste les services qui interagissent avec le contrôleur sélectionné.

- Affichez des informations détaillées sur une unité donnée et sur sa partie de la hiérarchie **cgroups** à l'aide de la commande **systemctl status <*system_unit*>** commande.

```
# systemctl status example.service
● example.service - My example service
   Loaded: loaded (/usr/lib/systemd/system/example.service; enabled; vendor preset:
disabled)
   Active: active (running) since Tue 2019-04-16 12:12:39 CEST; 3s ago
 Main PID: 17737 (bash)
    Tasks: 2 (limit: 11522)
   Memory: 496.0K (limit: 1.5M)
   CGroup: /system.slice/example.service
           ├─17737 /bin/bash /home/jdoe/example.sh
           └─17743 sleep 1
Apr 16 12:12:39 redhat systemd[1]: Started My example service.
Apr 16 12:12:39 redhat bash[17737]: The current time is Tue Apr 16 12:12:39 CEST 2019
Apr 16 12:12:40 redhat bash[17737]: The current time is Tue Apr 16 12:12:40 CEST 2019
```

Ressources supplémentaires

- Que sont les contrôleurs de ressources du noyau ?
- Les pages du manuel **systemd.resource-control(5)**, **cgroups(7)**

17.6. VISUALISATION DES GROUPES DE PROCESSUS

La procédure suivante décrit comment savoir à quel site *control group* (**cgroup**) appartient un processus. Vous pouvez ensuite consulter le site **cgroup** pour connaître les contrôleurs et les configurations spécifiques qu'il utilise.

Procédure

1. Pour savoir à quel site **cgroup** un processus appartient, exécutez la commande suivante **# cat proc/<*PID*>/cgroup** commande :

```
# cat /proc/2467/cgroup
0::/system.slice/example.service
```

   L'exemple de sortie se rapporte à un processus d'intérêt. Dans ce cas, il s'agit d'un processus identifié par **PID 2467**, qui appartient à l'unité **example.service**. Vous pouvez déterminer si le processus a été placé dans un groupe de contrôle correct, tel que défini par les spécifications du fichier de l'unité **systemd**.

2. Pour afficher les contrôleurs utilisés par le site **cgroup** et les fichiers de configuration correspondants, consultez le répertoire **cgroup**:

```
# cat /sys/fs/cgroup/system.slice/example.service/cgroup.controllers
memory pids

# ls /sys/fs/cgroup/system.slice/example.service/
cgroup.controllers
cgroup.events
…
cpu.pressure
```

```
cpu.stat
io.pressure
memory.current
memory.events
…
pids.current
pids.events
pids.max
```

NOTE

La hiérarchie de la version 1 de **cgroups** utilise un modèle par contrôleur. Par conséquent, la sortie du fichier **/proc/*PID*/cgroup** indique à quel **cgroups** de chaque contrôleur le PID appartient. Vous pouvez trouver les **cgroups** correspondants dans les répertoires des contrôleurs à l'adresse suivante **/sys/fs/cgroup/*<controller_name>*/**.

Ressources supplémentaires

- **cgroups(7)** page du manuel
- Que sont les contrôleurs de ressources du noyau ?
- Documentation dans le fichier **/usr/share/doc/kernel-doc-<kernel_version>/Documentation/admin-guide/cgroup-v2.rst** (après l'installation du paquet **kernel-doc**)

17.7. CONTRÔLE DE LA CONSOMMATION DES RESSOURCES

Affichez une liste des groupes de contrôle en cours d'exécution (**cgroups**) et leur consommation de ressources en temps réel.

Procédure

1. La commande **systemd-cgtop** permet d'afficher un compte dynamique des sites en cours d'exécution (**cgroups**).

```
# systemd-cgtop
Control Group                          Tasks   %CPU   Memory  Input/s Output/s
/                                        607   29.8     1.5G        -        -
/system.slice                            125      -   428.7M        -        -
/system.slice/ModemManager.service         3      -     8.6M        -        -
/system.slice/NetworkManager.service       3      -    12.8M        -        -
/system.slice/accounts-daemon.service      3      -     1.8M        -        -
/system.slice/boot.mount                   -      -    48.0K        -        -
/system.slice/chronyd.service              1      -     2.0M        -        -
/system.slice/cockpit.socket               -      -     1.3M        -        -
/system.slice/colord.service               3      -     3.5M        -        -
/system.slice/crond.service                1      -     1.8M        -        -
/system.slice/cups.service                 1      -     3.1M        -        -
/system.slice/dev-hugepages.mount          -      -   244.0K        -        -
/system.slice/dev-mapper-rhel\x2dswap.swap -      -   912.0K        -        -
/system.slice/dev-mqueue.mount             -      -    48.0K        -        -
```

```
/system.slice/example.service            2      -    2.0M        -        -
/system.slice/firewalld.service          2      -   28.8M        -        -
...
```

L'exemple suivant affiche les sites **cgroups** en cours d'exécution, classés en fonction de leur utilisation des ressources (CPU, mémoire, charge d'E/S sur disque). La liste est actualisée toutes les secondes par défaut. Elle offre donc un aperçu dynamique de l'utilisation réelle des ressources de chaque groupe de contrôle.

Ressources supplémentaires

- La page du manuel **systemd-cgtop(1)**

17.8. UTILISATION DES FICHIERS UNITAIRES DE SYSTEMD POUR FIXER DES LIMITES AUX APPLICATIONS

Chaque unité existante ou en cours d'exécution est supervisée par **systemd**, qui crée également des groupes de contrôle pour ces unités. Les unités ont des fichiers de configuration dans le répertoire **/usr/lib/systemd/system/**. Vous pouvez modifier manuellement les fichiers d'unité pour fixer des limites, établir des priorités ou contrôler l'accès aux ressources matérielles pour des groupes de processus.

Conditions préalables

- Vous disposez des privilèges **root**.

Procédure

1. Modifier le fichier **/usr/lib/systemd/system/example.service** pour limiter l'utilisation de la mémoire d'un service :

```
…
[Service]
MemoryMax=1500K
…
```

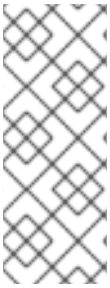
La configuration ci-dessus impose une limite de mémoire maximale que les processus d'un groupe de contrôle ne peuvent pas dépasser. Le service **example.service** fait partie d'un tel groupe de contrôle auquel des limites ont été imposées. Vous pouvez utiliser les suffixes K, M, G ou T pour identifier le kilo-octet, le méga-octet, le giga-octet ou le téra-octet comme unité de mesure.

2. Recharger tous les fichiers de configuration de l'unité :

```
# systemctl daemon-reload
```

3. Redémarrer le service :

```
# systemctl restart example.service
```



NOTE

Vous pouvez consulter l'ensemble des options de configuration pour **systemd** dans les pages suivantes du manuel :

- **systemd.resource-control(5)**
- **systemd.exec(5)**

Vérification

1. Vérifiez que les modifications ont bien été prises en compte :

```
# cat /sys/fs/cgroup/system.slice/example.service/memory.max
1536000
```

L'exemple montre que la consommation de mémoire a été limitée à environ 1 500 Ko.

Ressources supplémentaires

- Comprendre les cgroups
- *Configuring basic system settings* dans Red Hat Enterprise Linux
- **systemd.resource-control(5) systemd.exec(5)**, pages de manuel **cgroups(7)**

17.9. UTILISATION DE LA COMMANDE SYSTEMCTL POUR FIXER DES LIMITES AUX APPLICATIONS

Les paramètres d'affinité de l'unité centrale permettent de limiter l'accès d'un processus particulier à certaines unités centrales. En effet, le planificateur de CPU ne planifie jamais l'exécution d'un processus sur une unité centrale qui ne figure pas dans le masque d'affinité du processus.

Le masque d'affinité CPU par défaut s'applique à tous les services gérés par **systemd**.

Pour configurer le masque d'affinité CPU pour un service particulier **systemd**, **systemd** propose **CPUAffinity=** à la fois comme option de fichier d'unité et comme option de configuration de gestionnaire dans le fichier **/etc/systemd/system.conf**.

L'option **CPUAffinity= unit file option** définit une liste d'unités centrales ou de plages d'unités centrales qui sont fusionnées et utilisées comme masque d'affinité.

Après avoir configuré le masque d'affinité CPU pour un service **systemd** particulier, vous devez redémarrer le service pour appliquer les modifications.

Procédure

Pour définir le masque d'affinité CPU pour un service **systemd** particulier en utilisant l'option **CPUAffinity unit file** option :

1. Vérifiez les valeurs de l'option de fichier de l'unité **CPUAffinity** dans le service de votre choix :

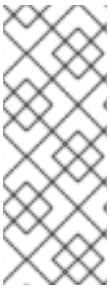
```
systemctl show --property <CPU affinity configuration option> <service name>
```

2. En tant que root, définissez la valeur requise de l'option **CPUAffinity** unit file pour les plages de CPU utilisées comme masque d'affinité :

```
# systemctl set-property <service name> CPUAffinity=<value>
```

3. Redémarrez le service pour appliquer les modifications.

```
# systemctl restart <service name>
```



NOTE

Vous pouvez consulter l'ensemble des options de configuration pour **systemd** dans les pages suivantes du manuel :

- **systemd.resource-control(5)**
- **systemd.exec(5)**

17.10. DÉFINITION DE L'AFFINITÉ PAR DÉFAUT DE L'UNITÉ CENTRALE PAR LE BIAIS DE LA CONFIGURATION DU GESTIONNAIRE

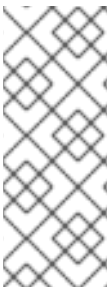
Le fichier **CPUAffinity option** dans le fichier **/etc/systemd/system.conf** définit un masque d'affinité pour le numéro d'identification de processus (PID) 1 et tous les processus dérivés du PID1. Vous pouvez ensuite remplacer le fichier **CPUAffinity** pour chaque service.

Pour définir le masque d'affinité CPU par défaut pour tous les services systemd en utilisant l'option **manager configuration**:

1. Définissez les numéros de CPU pour l'option **CPUAffinity=** dans le fichier **/etc/systemd/system.conf**.
2. Enregistrez le fichier modifié et rechargez le service **systemd**:

```
# systemctl daemon-reload
```

3. Redémarrez le serveur pour appliquer les modifications.



NOTE

Vous pouvez consulter l'ensemble des options de configuration pour **systemd** dans les pages suivantes du manuel :

- **systemd.resource-control(5)**
- **systemd.exec(5)**

17.11. CONFIGURATION DES POLITIQUES NUMA À L'AIDE DE SYSTEMD

L'accès non uniforme à la mémoire (NUMA) est une conception de sous-système de mémoire d'ordinateur, dans laquelle le temps d'accès à la mémoire dépend de l'emplacement physique de la mémoire par rapport au processeur.

La mémoire proche de l'unité centrale a un temps de latence plus faible (mémoire locale) que la mémoire locale d'une autre unité centrale (mémoire étrangère) ou partagée entre plusieurs unités centrales.

En ce qui concerne le noyau Linux, la politique NUMA régit où (par exemple, sur quels nœuds NUMA) le noyau alloue des pages de mémoire physique pour le processus.

systemd fournit les options de fichier d'unité **NUMAPolicy** et **NUMAMask** pour contrôler les politiques d'allocation de mémoire pour les services.

Procédure

Pour définir la politique de mémoire NUMA à l'aide de l'option **NUMAPolicy unit file** option :

1. Vérifiez les valeurs de l'option de fichier de l'unité **NUMAPolicy** dans le service de votre choix :

```
$ systemctl show --property <NUMA policy configuration option> <service name>
```

2. En tant qu'utilisateur principal, définissez le type de stratégie requis pour l'option de fichier d'unité **NUMAPolicy**:

```
# systemctl set-property <service name> NUMAPolicy=<value>
```

3. Redémarrez le service pour appliquer les modifications.

```
# systemctl restart <service name>
```

Pour définir un paramètre global **NUMAPolicy** via l'option **manager configuration**:

1. Recherchez l'option **NUMAPolicy** dans le fichier **/etc/systemd/system.conf**.
2. Modifiez le type de politique et enregistrez le fichier.
3. Recharger la configuration de **systemd**:

```
# systemd daemon-reload
```

4. Redémarrer le serveur.

IMPORTANT

Lorsque vous configurez une politique NUMA stricte, par exemple **bind**, veillez à définir également l'option de fichier d'unité **CPUAffinity=**.

Ressources supplémentaires

- Utilisation de la commande systemctl pour fixer des limites aux applications
- Les pages du manuel **systemd.resource-control(5)**, **systemd.exec(5)**, **set_mempolicy(2)**.

17.12. OPTIONS DE CONFIGURATION DE LA POLITIQUE NUMA POUR SYSTEMD

Systemd propose les options suivantes pour configurer la politique NUMA :

NUMAPolicy

Contrôle la politique de mémoire NUMA des processus exécutés. Les types de politique suivants sont possibles :

- par défaut
- préférée
- lier
- entrelacement
- local

NUMAMask

Contrôle la liste des nœuds NUMA associée à la politique NUMA sélectionnée.
Notez que l'option **NUMAMask** ne doit pas être spécifiée pour les politiques suivantes :

- par défaut
- local

Pour la stratégie préférée, la liste ne spécifie qu'un seul nœud NUMA.

Ressources supplémentaires

- **systemd.resource-control(5)**, **systemd.exec(5)**, et **set_mempolicy(2)** pages de manuel

17.13. CRÉATION DE CGROUPS TRANSITOIRES À L'AIDE DE LA COMMANDE SYSTEMD-RUN

Le site transitoire **cgroups** fixe des limites aux ressources consommées par une unité (service ou champ d'application) pendant sa durée d'exécution.

Procédure

- Pour créer un groupe de contrôle transitoire, utilisez la commande **systemd-run** dans le format suivant :

```
# systemd-run --unit=<name> --slice=<name>.slice <command>
```

Cette commande crée et démarre un service transitoire ou une unité d'étendue et exécute une commande personnalisée dans cette unité.

  - L'option **--unit=<name>** donne un nom à l'unité. Si **--unit** n'est pas spécifié, le nom est généré automatiquement.
  - L'option **--slice=<*name*>.slice** fait de votre service ou de votre unité de portée un membre d'une tranche spécifiée. Remplacez **<*name*>.slice** par le nom d'une tranche existante (comme indiqué dans la sortie de **systemctl -t slice**), ou créez une nouvelle tranche en indiquant un nom unique. Par défaut, les services et les champs d'application sont créés en tant que membres de la tranche **system.slice**.

- Remplacez ***<command>*** par la commande que vous souhaitez exécuter dans le service ou l'unité de portée.
  Le message suivant s'affiche pour confirmer que vous avez créé et démarré le service ou l'étendue avec succès :

  ```
  # Exécution en tant qu'unité <name>.service
  ```

- Il est possible de laisser l'unité fonctionner après la fin de ses processus afin de collecter des informations sur l'exécution :

  ```
  # systemd-run --unit=<name> --slice=<name>.slice --remain-after-exit <command>
  ```

  La commande crée et démarre une unité de service transitoire et exécute une commande personnalisée dans cette unité. L'option **--remain-after-exit** permet de s'assurer que le service continue de fonctionner après la fin de ses processus.

Ressources supplémentaires

- Comprendre les groupes de contrôle
- *Configuring basic system settings* dans RHEL
- la page du manuel **systemd-run(1)**

17.14. SUPPRESSION DES GROUPES DE CONTRÔLE TRANSITOIRES

Vous pouvez utiliser le gestionnaire de systèmes et de services **systemd** pour supprimer les groupes de contrôle transitoires (**cgroups**) si vous n'avez plus besoin de limiter, de hiérarchiser ou de contrôler l'accès aux ressources matérielles pour des groupes de processus.

Les sites transitoires **cgroups** sont automatiquement libérés lorsque tous les processus contenus dans un service ou une unité d'étendue sont terminés.

Procédure

- Pour arrêter l'unité de service avec tous ses processus, exécutez :

  ```
  # systemctl stop name .service
  ```

- Pour mettre fin à un ou plusieurs processus de l'unité, exécutez :

  ```
  # systemctl kill name .service --kill-who=PID,... --signal=<signal>
  ```

  La commande ci-dessus utilise l'option **--kill-who** pour sélectionner le(s) processus du groupe de contrôle que vous souhaitez arrêter. Pour tuer plusieurs processus en même temps, passez une liste de PIDs séparés par des virgules. L'option **--signal** détermine le type de signal POSIX à envoyer aux processus spécifiés. Le signal par défaut est *SIGTERM*.

Ressources supplémentaires

- Comprendre les groupes de contrôle
- Que sont les contrôleurs de ressources du noyau ?

- **systemd.resource-control(5)**, **cgroups(7)** pages de manuel
- *Configuring basic system settings* dans RHEL

CHAPITRE 18. COMPRENDRE LES CGROUPS

Vous pouvez utiliser la fonctionnalité du noyau *control groups* (**cgroups**) pour fixer des limites, établir des priorités ou isoler les ressources matérielles des processus. Cela vous permet de contrôler granulairement l'utilisation des ressources des applications afin de les utiliser plus efficacement.

18.1. COMPRENDRE LES GROUPES DE CONTRÔLE

Control groups est une fonctionnalité du noyau Linux qui vous permet d'organiser les processus en groupes hiérarchiquement ordonnés - **cgroups**. La hiérarchie (arbre des groupes de contrôle) est définie en fournissant une structure au système de fichiers virtuel **cgroups**, monté par défaut sur le répertoire /**sys**/**fs**/**cgroup**/. Le gestionnaire de systèmes et de services **systemd** utilise **cgroups** pour organiser toutes les unités et tous les services qu'il régit. Vous pouvez également gérer manuellement les hiérarchies de **cgroups** en créant et en supprimant des sous-répertoires dans le répertoire /**sys**/**fs**/**cgroup**/.

Les contrôleurs de ressources (un composant du noyau) modifient alors le comportement des processus dans **cgroups** en limitant, en priorisant ou en allouant les ressources du système (telles que le temps de l'unité centrale, la mémoire, la largeur de bande du réseau ou diverses combinaisons) de ces processus.

La valeur ajoutée de **cgroups** est l'agrégation de processus qui permet de répartir les ressources matérielles entre les applications et les utilisateurs. Il est ainsi possible d'accroître l'efficacité globale, la stabilité et la sécurité de l'environnement des utilisateurs.

Groupes de contrôle version 1

Control groups version 1 (**cgroups-v1**) fournissent une hiérarchie de contrôleurs par ressource. Cela signifie que chaque ressource, telle que l'unité centrale, la mémoire, les E/S, etc., possède sa propre hiérarchie de groupes de contrôle. Il est possible de combiner différentes hiérarchies de groupes de contrôle de manière à ce qu'un contrôleur puisse coordonner avec un autre la gestion de leurs ressources respectives. Toutefois, les deux contrôleurs peuvent appartenir à des hiérarchies de processus différentes, ce qui ne permet pas une bonne coordination.
Les contrôleurs **cgroups-v1** ont été développés sur une longue période et, par conséquent, le comportement et la dénomination de leurs fichiers de contrôle ne sont pas uniformes.

Groupes de contrôle version 2

Les problèmes de coordination des contrôleurs, qui découlaient de la flexibilité de la hiérarchie, ont conduit au développement de *control groups version 2*.
Control groups version 2 (**cgroups-v2**) fournit une hiérarchie de groupe de contrôle unique par rapport à laquelle tous les contrôleurs de ressources sont montés.

Le comportement et la dénomination des fichiers de contrôle sont cohérents d'un contrôleur à l'autre.

IMPORTANT

Par défaut, RHEL 9 monte et utilise **cgroups-v2**.

Cette sous-section est basée sur une présentation de Devconf.cz 2019.[1]

Ressources supplémentaires

- Que sont les contrôleurs de ressources du noyau ?
- **cgroups(7)** page du manuel
- cgroups-v1
- cgroups-v2

18.2. QUE SONT LES CONTRÔLEURS DE RESSOURCES DU NOYAU ?

La fonctionnalité des groupes de contrôle est activée par les contrôleurs de ressources du noyau. RHEL 9 prend en charge différents contrôleurs pour *control groups version 1* (**cgroups-v1**) et *control groups version 2* (**cgroups-v2**).

Un contrôleur de ressources, également appelé sous-système de groupe de contrôle, est un sous-système du noyau qui représente une ressource unique, telle que le temps de l'unité centrale, la mémoire, la bande passante du réseau ou les entrées/sorties du disque. Le noyau Linux fournit une gamme de contrôleurs de ressources qui sont montés automatiquement par le système **systemd** et le gestionnaire de services. La liste des contrôleurs de ressources actuellement montés se trouve dans le fichier /**proc**/**cgroups**.

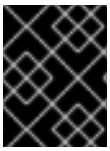
Les contrôleurs suivants sont disponibles pour **cgroups-v1**:

- **blkio** - peut fixer des limites à l'accès aux entrées/sorties vers et depuis les périphériques de bloc.
- **cpu** - peut ajuster les paramètres de l'ordonnanceur Completely Fair Scheduler (CFS) pour les tâches du groupe de contrôle. Il est monté avec le contrôleur **cpuacct** sur le même support.
- **cpuacct** - crée des rapports automatiques sur les ressources CPU utilisées par les tâches d'un groupe de contrôle. Il est monté avec le contrôleur **cpu** sur le même support.
- **cpuset** - peut être utilisé pour limiter l'exécution des tâches du groupe de contrôle à un sous-ensemble spécifié de CPU et pour ordonner aux tâches d'utiliser la mémoire uniquement sur les nœuds de mémoire spécifiés.
- **devices** - peut contrôler l'accès aux appareils pour les tâches d'un groupe de contrôle.
- **freezer** - peut être utilisé pour suspendre ou reprendre des tâches dans un groupe de contrôle.
- **memory** - peut être utilisé pour fixer des limites à l'utilisation de la mémoire par les tâches d'un groupe de contrôle et génère des rapports automatiques sur les ressources mémoire utilisées par ces tâches.
- **net_cls** - marque les paquets réseau avec un identifiant de classe (**classid**) qui permet au contrôleur de trafic Linux (la commande **tc**) d'identifier les paquets qui proviennent d'une tâche de groupe de contrôle particulière. Un sous-système de **net_cls**, **net_filter** (iptables), peut également utiliser cette étiquette pour effectuer des actions sur ces paquets. Le **net_filter** marque les sockets réseau avec un identifiant de pare-feu (**fwid**) qui permet au pare-feu Linux (via la commande **iptables**) d'identifier les paquets provenant d'une tâche particulière du groupe de contrôle.
- **net_prio** - définit la priorité du trafic réseau.
- **pids** - peut fixer des limites pour un certain nombre de processus et leurs enfants dans un groupe de contrôle.

- **perf_event** - peut regrouper les tâches à surveiller par l'utilitaire de surveillance des performances et de création de rapports **perf**.
- **rdma** - peut fixer des limites aux ressources spécifiques Remote Direct Memory Access/InfiniBand dans un groupe de contrôle.
- **hugetlb** - peut être utilisé pour limiter l'utilisation de pages de mémoire virtuelle de grande taille par les tâches d'un groupe de contrôle.

Les contrôleurs suivants sont disponibles pour **cgroups-v2**:

- **io** - Un suivi de **blkio** of **cgroups-v1**.
- **memory** - Un suivi de **memory** of **cgroups-v1**.
- **pids** - Identique à **pids** dans **cgroups-v1**.
- **rdma** - Identique à **rdma** dans **cgroups-v1**.
- **cpu** - Un suivi de **cpu** et **cpuacct** de **cgroups-v1**.
- **cpuset** - Ne prend en charge que la fonctionnalité de base (**cpus{,.effective}**, **mems{,.effective}**) avec une nouvelle fonction de partition.
- **perf_event** - La prise en charge est inhérente, il n'y a pas de fichier de contrôle explicite. Vous pouvez spécifier une adresse **v2 cgroup** en tant que paramètre de la commande **perf** qui établira le profil de toutes les tâches contenues dans cette adresse **cgroup**.



IMPORTANT

Un contrôleur de ressources peut être utilisé soit dans une hiérarchie **cgroups-v1**, soit dans une hiérarchie **cgroups-v2**, mais pas simultanément dans les deux.

Ressources supplémentaires

- **cgroups(7)** page du manuel
- Documentation dans le répertoire **/usr/share/doc/kernel-doc-<kernel_version>/Documentation/cgroups-v1/** (après avoir installé le paquetage **kernel-doc**).

18.3. QU'EST-CE QU'UN ESPACE DE NOMS ?

Les espaces de noms sont l'une des méthodes les plus importantes pour organiser et identifier les objets logiciels.

Un espace de noms enveloppe une ressource système globale (par exemple un point de montage, un périphérique réseau ou un nom d'hôte) dans une abstraction qui donne l'impression aux processus de l'espace de noms qu'ils ont leur propre instance isolée de la ressource globale. Les conteneurs sont l'une des technologies les plus courantes qui utilisent les espaces de noms.

Les modifications apportées à une ressource globale particulière ne sont visibles que par les processus de cet espace de noms et n'affectent pas le reste du système ou d'autres espaces de noms.

Pour savoir de quels espaces de noms un processus est membre, vous pouvez vérifier les liens symboliques dans le répertoire **/proc/*<PID>*/ns/** dans le répertoire

Le tableau suivant présente les espaces de noms pris en charge et les ressources qu'ils isolent :

Espace de noms	Isolats
Mount	Points de montage
UTS	Nom d'hôte et nom de domaine NIS
IPC	System V IPC, files d'attente de messages POSIX
PID	ID de processus
Network	Dispositifs de réseau, piles, ports, etc
User	ID d'utilisateur et de groupe
Control groups	Répertoire racine du groupe de contrôle

Ressources supplémentaires

- **namespaces(7)** et **cgroup_namespaces(7)** pages de manuel
- Comprendre les groupes de contrôle

[1] Linux Control Group v2 - An Introduction, Devconf.cz 2019 présentation par Waiman Long

CHAPITRE 19. AMÉLIORER LES PERFORMANCES DU SYSTÈME AVEC ZSWAP

Vous pouvez améliorer les performances du système en activant la fonction du noyau **zswap**.

19.1. QU'EST-CE QUE ZSWAP ?

zswap est une fonctionnalité du noyau qui fournit un cache RAM compressé pour les pages d'échange, ce qui peut améliorer les performances du système.

Le mécanisme fonctionne comme suit : **zswap** prend les pages qui sont en train d'être échangées et tente de les compresser dans un pool de mémoire RAM alloué dynamiquement. Lorsque le pool est plein ou que la RAM est épuisée, **zswap** expulse les pages de la mémoire cache compressée sur une base LRU (la moins récemment utilisée) vers le périphérique d'échange de sauvegarde. Une fois la page décompressée dans le cache d'échange, **zswap** libère la version compressée dans le pool.

Les avantages de lazswap

- réduction significative des E/S
- amélioration significative de la performance de la charge de travail

Dans Red Hat Enterprise Linux 9, **zswap** est activé par défaut.

Ressources supplémentaires

- Qu'est-ce que Zswap ?

19.2. ACTIVATION DE ZSWAP AU MOMENT DE L'EXÉCUTION

Vous pouvez activer la fonction **zswap** lors de l'exécution du système à l'aide de l'interface **sysfs**.

Conditions préalables

- Vous disposez des droits d'accès à la racine.

Procédure

- Activer **zswap**:

```
# echo 1 > /sys/module/zswap/parameters/enabled
```

Étape de vérification

- Vérifiez que **zswap** est activé :

```
# grep -r . /sys/kernel/debug/zswap

duplicate_entry:0
pool_limit_hit:13422200
pool_total_size:6184960 (pool size in total in pages)
reject_alloc_fail:5
```

```
reject_compress_poor:0
reject_kmemcache_fail:0
reject_reclaim_fail:13422200
stored_pages:4251 (pool size after compression)
written_back_pages:0
```

Ressources supplémentaires

- Comment activer la fonction Zswap ?

19.3. ACTIVATION PERMANENTE DE ZSWAP

Vous pouvez activer la fonction **zswap** de manière permanente en fournissant le paramètre de ligne de commande **zswap.enabled=1** kernel.

Conditions préalables

- Vous disposez des droits d'accès à la racine.
- L'utilitaire **grubby** ou **zipl** est installé sur votre système.

Procédure

1. Activer **zswap** de façon permanente :

```
# grubby --update-kernel=/boot/vmlinuz-$(uname -r) --args="zswap.enabled=1"
```

2. Redémarrez le système pour que les modifications soient prises en compte.

Verification steps

- Vérifiez que **zswap** est activé :

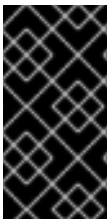
```
# cat /proc/cmdline

BOOT_IMAGE=(hd0,msdos1)/vmlinuz-5.14.0-70.5.1.el9_0.x86_64
root=/dev/mapper/rhel-root ro crashkernel=1G-4G:192M,4G-64G:256M,64G-:512M
resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root
rd.lvm.lv=rhel/swap rhgb quiet
zswap.enabled=1
```

Ressources supplémentaires

- Comment activer la fonction Zswap ?
- Configuration des paramètres de la ligne de commande du noyau

CHAPITRE 20. UTILISATION DE CGROUPFS POUR GÉRER MANUELLEMENT LES CGROUPS

Vous pouvez gérer les hiérarchies **cgroup** sur votre système en créant des répertoires sur le système de fichiers virtuel **cgroupfs**. Le système de fichiers est monté par défaut sur le répertoire **/sys/fs/cgroup/** et vous pouvez spécifier les configurations souhaitées dans des fichiers de contrôle dédiés.



IMPORTANT

En général, Red Hat vous recommande d'utiliser **systemd** pour contrôler l'utilisation des ressources du système. Vous ne devez configurer manuellement le système de fichiers virtuel **cgroups** que dans des cas particuliers. Par exemple, lorsque vous devez utiliser des contrôleurs **cgroup-v1** qui n'ont pas d'équivalents dans la hiérarchie **cgroup-v2**.

20.1. CRÉATION DE CGROUPS ET ACTIVATION DE CONTRÔLEURS DANS LE SYSTÈME DE FICHIERS CGROUPS-V2

Vous pouvez gérer le répertoire *control groups* (**cgroups**) en créant ou en supprimant des répertoires et en écrivant sur les fichiers du système de fichiers virtuel **cgroups**. Le système de fichiers est monté par défaut sur le répertoire **/sys/fs/cgroup/**. Pour utiliser les paramètres des contrôleurs **cgroups**, vous devez également activer les contrôleurs souhaités pour l'enfant **cgroups**. La racine **cgroup** a, par défaut, activé les contrôleurs **memory** et **pids** pour son enfant **cgroups**. Par conséquent, Red Hat recommande de créer au moins deux niveaux d'enfants **cgroups** à l'intérieur de la racine **/sys/fs/cgroup/ cgroup** . De cette manière, vous pouvez éventuellement supprimer les contrôleurs **memory** et **pids** de l'enfant **cgroups** et maintenir une meilleure clarté organisationnelle des fichiers **cgroup**.

Conditions préalables

- Vous disposez des droits d'accès à la racine.

Procédure

1. Créez le répertoire **/sys/fs/cgroup/Example/**:

```
# mkdir /sys/fs/cgroup/Example/
```

   Le répertoire **/sys/fs/cgroup/Example/** définit un groupe enfant. Lorsque vous créez le répertoire **/sys/fs/cgroup/Example/**, certains fichiers d'interface **cgroups-v2** sont automatiquement créés dans le répertoire. Le répertoire **/sys/fs/cgroup/Example/** contient également des fichiers spécifiques aux contrôleurs **memory** et **pids**.

2. Il est possible d'inspecter le groupe de contrôle enfant nouvellement créé :

```
# ll /sys/fs/cgroup/Example/
-r—r—r--. 1 root root 0 Jun  1 10:33 cgroup.controllers
-r—r—r--. 1 root root 0 Jun  1 10:33 cgroup.events
-rw-r—r--. 1 root root 0 Jun  1 10:33 cgroup.freeze
-rw-r—r--. 1 root root 0 Jun  1 10:33 cgroup.procs
…
-rw-r—r--. 1 root root 0 Jun  1 10:33 cgroup.subtree_control
-r—r—r--. 1 root root 0 Jun  1 10:33 memory.events.local
-rw-r—r--. 1 root root 0 Jun  1 10:33 memory.high
```

```
-rw-r—r--. 1 root root 0 Jun  1 10:33 memory.low
…
-r—r—r--. 1 root root 0 Jun  1 10:33 pids.current
-r—r—r--. 1 root root 0 Jun  1 10:33 pids.events
-rw-r—r--. 1 root root 0 Jun  1 10:33 pids.max
```

L'exemple de sortie montre les fichiers généraux de l'interface de contrôle **cgroup** tels que **cgroup.procs** ou **cgroup.controllers**. Ces fichiers sont communs à tous les groupes de contrôle, quels que soient les contrôleurs activés.

Les fichiers tels que **memory.high** et **pids.max** se rapportent aux contrôleurs **memory** et **pids**, qui se trouvent dans le groupe de contrôle racine (/**sys**/**fs**/**cgroup**/), et sont activés par défaut par **systemd**.

Par défaut, le groupe enfant nouvellement créé hérite de tous les paramètres du parent **cgroup**. Dans ce cas, aucune limite n'est imposée par la racine **cgroup**.

3. Vérifiez que les contrôleurs souhaités sont disponibles dans le fichier **/sys/fs/cgroup/cgroup.controllers**:

```
# cat /sys/fs/cgroup/cgroup.controllers
cpuset cpu io memory hugetlb pids rdma
```

4. Activez les contrôleurs souhaités. Dans cet exemple, il s'agit des contrôleurs **cpu** et **cpuset**:

```
# echo "+cpu" >> /sys/fs/cgroup/cgroup.subtree_control
# echo "+cpuset" >> /sys/fs/cgroup/cgroup.subtree_control
```

Ces commandes activent les contrôleurs **cpu** et **cpuset** pour les groupes enfants immédiats du groupe de contrôle racine /**sys**/**fs**/**cgroup**/. Y compris le groupe de contrôle **Example** nouvellement créé. Un site *child group* est l'endroit où vous pouvez spécifier des processus et appliquer des contrôles à chacun des processus en fonction de vos critères.

Les utilisateurs peuvent lire le contenu du fichier **cgroup.subtree_control** à n'importe quel niveau pour avoir une idée des contrôleurs qui seront disponibles pour l'activation dans le groupe enfant immédiat.

NOTE

Par défaut, le fichier /**sys**/**fs**/**cgroup**/**cgroup.subtree_control** du groupe de contrôle racine contient les contrôleurs **memory** et **pids**.

5. Activez les contrôleurs souhaités pour l'enfant **cgroups** du groupe de contrôle **Example**:

```
# echo " cpu cpuset" >> /sys/fs/cgroup/Example/cgroup.subtree_control
```

Cette commande garantit que le groupe de contrôle enfant immédiat *only* aura des contrôleurs pertinents pour réguler la distribution du temps CPU - et non des contrôleurs **memory** ou **pids**.

6. Créez le répertoire /**sys**/**fs**/**cgroup**/**Example**/**tasks**/:

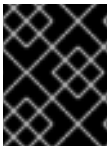
```
# mkdir /sys/fs/cgroup/Example/tasks/
```

Le répertoire /**sys**/**fs**/**cgroup**/**Example**/**tasks**/ définit un groupe enfant avec des fichiers qui se

rapportent uniquement aux contrôleurs **cpu** et **cpuset**. Vous pouvez maintenant affecter des processus à ce groupe de contrôle et utiliser les options des contrôleurs **cpu** et **cpuset** pour vos processus.

7. Optionnellement, inspecter le groupe de contrôle de l'enfant :

```
# ll /sys/fs/cgroup/Example/tasks
-r—r—r--. 1 root root 0 Jun  1 11:45 cgroup.controllers
-r—r—r--. 1 root root 0 Jun  1 11:45 cgroup.events
-rw-r—r--. 1 root root 0 Jun  1 11:45 cgroup.freeze
-rw-r—r--. 1 root root 0 Jun  1 11:45 cgroup.max.depth
-rw-r—r--. 1 root root 0 Jun  1 11:45 cgroup.max.descendants
-rw-r—r--. 1 root root 0 Jun  1 11:45 cgroup.procs
-r—r—r--. 1 root root 0 Jun  1 11:45 cgroup.stat
-rw-r—r--. 1 root root 0 Jun  1 11:45 cgroup.subtree_control
-rw-r—r--. 1 root root 0 Jun  1 11:45 cgroup.threads
-rw-r—r--. 1 root root 0 Jun  1 11:45 cgroup.type
-rw-r—r--. 1 root root 0 Jun  1 11:45 cpu.max
-rw-r—r--. 1 root root 0 Jun  1 11:45 cpu.pressure
-rw-r—r--. 1 root root 0 Jun  1 11:45 cpuset.cpus
-r—r—r--. 1 root root 0 Jun  1 11:45 cpuset.cpus.effective
-rw-r—r--. 1 root root 0 Jun  1 11:45 cpuset.cpus.partition
-rw-r—r--. 1 root root 0 Jun  1 11:45 cpuset.mems
-r—r—r--. 1 root root 0 Jun  1 11:45 cpuset.mems.effective
-r—r—r--. 1 root root 0 Jun  1 11:45 cpu.stat
-rw-r—r--. 1 root root 0 Jun  1 11:45 cpu.weight
-rw-r—r--. 1 root root 0 Jun  1 11:45 cpu.weight.nice
-rw-r—r--. 1 root root 0 Jun  1 11:45 io.pressure
-rw-r—r--. 1 root root 0 Jun  1 11:45 memory.pressure
```



IMPORTANT

Le contrôleur **cpu** n'est activé que si le groupe de contrôle enfant concerné comporte au moins deux processus qui se disputent le temps d'une seule unité centrale.

Verification steps

- Facultatif : confirmez que vous avez créé un nouveau site **cgroup** avec uniquement les contrôleurs souhaités actifs :

```
# cat /sys/fs/cgroup/Example/tasks/cgroup.controllers
cpuset cpu
```

Ressources supplémentaires

- Comprendre les groupes de contrôle
- Que sont les contrôleurs de ressources du noyau ?
- Montage de cgroups-v1
- **cgroups(7)**, **sysfs(5)** pages de manuel

20.2. CONTRÔLE DE LA RÉPARTITION DU TEMPS D'UTILISATION DE L'UNITÉ CENTRALE POUR LES APPLICATIONS EN AJUSTANT LE POIDS DE L'UNITÉ CENTRALE

Vous devez attribuer des valeurs aux fichiers pertinents du contrôleur **cpu** pour réguler la distribution du temps de CPU aux applications sous l'arborescence spécifique du cgroup.

Conditions préalables

- Vous disposez des droits d'accès à la racine.
- Vous disposez d'applications pour lesquelles vous souhaitez contrôler la répartition du temps de l'unité centrale.
- Vous avez créé une hiérarchie à deux niveaux de *child control groups* à l'intérieur de /**sys**/**fs**/**cgroup**/ *root control group* comme dans l'exemple suivant :

```
...
├── Example
│   ├── g1
│   ├── g2
│   └── g3
...
```

- Vous avez activé le contrôleur **cpu** dans le groupe de contrôle parent et dans les groupes de contrôle enfants de la même manière que celle décrite dans la section Création de cgroups et activation de contrôleurs dans le système de fichiers cgroups-v2.

Procédure

1. Configurez les poids CPU souhaités afin de respecter les restrictions de ressources au sein des groupes de contrôle :

```
# echo "150" > /sys/fs/cgroup/Example/g1/cpu.weight
# echo "100" > /sys/fs/cgroup/Example/g2/cpu.weight
# echo "50" > /sys/fs/cgroup/Example/g3/cpu.weight
```

2. Ajoutez les PID des applications aux groupes enfants **g1**, **g2**, et **g3**:

```
# echo "33373" > /sys/fs/cgroup/Example/g1/cgroup.procs
# echo "33374" > /sys/fs/cgroup/Example/g2/cgroup.procs
# echo "33377" > /sys/fs/cgroup/Example/g3/cgroup.procs
```

Les commandes de l'exemple garantissent que les applications souhaitées deviennent membres des cgroups enfants **Example**/**g***/ et que leur temps d'utilisation de l'unité centrale est réparti conformément à la configuration de ces cgroups.

Les poids des cgroups enfants (**g1**, **g2**, **g3**) qui ont des processus en cours sont additionnés au niveau du cgroup parent (**Example**). Les ressources de l'unité centrale sont ensuite réparties proportionnellement en fonction des poids respectifs.

Par conséquent, lorsque tous les processus s'exécutent en même temps, le noyau alloue à chacun d'entre eux un temps d'utilisation proportionnel basé sur le fichier **cpu.weight** de leur cgroupe respectif :

Enfant cgroup	**cpu.weight** fichier	Attribution du temps de l'unité centrale
g1	150	~50% (150/300)
g2	100	~33% (100/300)
g3	50	~16% (50/300)

La valeur du fichier du contrôleur **cpu.weight** n'est pas un pourcentage.

Si un processus cessait de fonctionner, laissant le cgroup **g2** sans aucun processus en cours, le calcul ne tiendrait pas compte du cgroup **g2** et ne prendrait en compte que les poids des cgroups **g1** et **g3**:

Enfant cgroup	**cpu.weight** fichier	Attribution du temps de l'unité centrale
g1	150	~75% (150/200)
g3	50	~25% (50/200)

IMPORTANT

Si un cgroup enfant a plusieurs processus en cours d'exécution, le temps CPU alloué au cgroup respectif sera distribué de manière égale aux processus membres de ce cgroup.

Vérification

1. Vérifiez que les applications s'exécutent dans les groupes de contrôle spécifiés :

```
# cat /proc/33373/cgroup /proc/33374/cgroup /proc/33377/cgroup
0::/Example/g1
0::/Example/g2
0::/Example/g3
```

La sortie de la commande montre les processus des applications spécifiées qui s'exécutent dans les cgroups enfants **Example/g***/.

2. Examinez la consommation actuelle de l'unité centrale des applications limitées :

```
# top
top - 05:17:18 up 1 day, 18:25,  1 user,  load average: 3.03, 3.03, 3.00
Tasks:  95 total,   4 running,  91 sleeping,   0 stopped,   0 zombie
%Cpu(s): 18.1 us, 81.6 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.3 hi,  0.0 si,  0.0 st
MiB Mem :   3737.0 total,   3233.7 free,    132.8 used,    370.5 buff/cache
MiB Swap:   4060.0 total,   4060.0 free,      0.0 used.   3373.1 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
33373 root      20   0   18720   1748   1460 R  49.5   0.0 415:05.87 sha1sum
```

```
 33374 root     20   0   18720   1756   1464 R  32.9   0.0 412:58.33 sha1sum
 33377 root     20   0   18720   1860   1568 R  16.3   0.0 411:03.12 sha1sum
   760 root     20   0  416620  28540  15296 S   0.3   0.7   0:10.23 tuned
     1 root     20   0  186328  14108   9484 S   0.0   0.4   0:02.00 systemd
     2 root     20   0       0      0      0 S   0.0   0.0   0:00.01 kthread
...
```

NOTE

Nous avons forcé tous les processus de l'exemple à s'exécuter sur une seule unité centrale pour une illustration plus claire. Le poids de l'unité centrale applique les mêmes principes lorsqu'il est utilisé sur plusieurs unités centrales.

Notez que la ressource CPU pour les applications **PID 33373**, **PID 33374** et **PID 33377** a été allouée en fonction des poids, 150, 100, 50, que vous avez attribués aux groupes enfants respectifs. Ces poids correspondent à environ 50 %, 33 % et 16 % du temps d'utilisation de l'unité centrale pour chaque application.

Ressources supplémentaires

- Comprendre les groupes de contrôle
- Que sont les contrôleurs de ressources du noyau ?
- Création de cgroups et activation de contrôleurs dans le système de fichiers cgroups-v2
- Modèles de distribution des ressources
- **cgroups(7)**, **sysfs(5)** pages de manuel

20.3. MONTAGE DE CGROUPS-V1

Au cours du processus de démarrage, RHEL 9 monte par défaut le système de fichiers virtuel **cgroup-v2**. Pour utiliser la fonctionnalité **cgroup-v1** en limitant les ressources pour vos applications, configurez manuellement le système.

NOTE

cgroup-v1 et **cgroup-v2** sont tous deux pleinement activés dans le noyau. Il n'y a pas de version de groupe de contrôle par défaut du point de vue du noyau, et c'est **systemd** qui décide du montage au démarrage.

Conditions préalables

- Vous disposez des droits d'accès à la racine.

Procédure

1. Configurer le système pour qu'il monte **cgroups-v1** par défaut lors du démarrage du système par le système **systemd** et le gestionnaire de services :

```
# grubby --update-kernel=/boot/vmlinuz-$(uname -r) --
args="systemd.unified_cgroup_hierarchy=0
systemd.legacy_systemd_cgroup_controller"
```

Cette opération ajoute les paramètres de ligne de commande du noyau nécessaires à l'entrée de démarrage actuelle.

Pour ajouter les mêmes paramètres à toutes les entrées de démarrage du noyau :

```
# grubby --update-kernel=ALL --args="systemd.unified_cgroup_hierarchy=0
systemd.legacy_systemd_cgroup_controller"
```

2. Redémarrez le système pour que les modifications soient prises en compte.

Vérification

1. Optionnellement, vérifiez que le système de fichiers **cgroups-v1** a été monté :

```
# mount -l | grep cgroup
tmpfs on /sys/fs/cgroup type tmpfs
(ro,nosuid,nodev,noexec,seclabel,size=4096k,nr_inodes=1024,mode=755,inode64)
cgroup on /sys/fs/cgroup/systemd type cgroup
(rw,nosuid,nodev,noexec,relatime,seclabel,xattr,release_agent=/usr/lib/systemd/systemd-
cgroups-agent,name=systemd)
cgroup on /sys/fs/cgroup/perf_event type cgroup
(rw,nosuid,nodev,noexec,relatime,seclabel,perf_event)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup
(rw,nosuid,nodev,noexec,relatime,seclabel,cpu,cpuacct)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,pids)
cgroup on /sys/fs/cgroup/cpuset type cgroup
(rw,nosuid,nodev,noexec,relatime,seclabel,cpuset)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup
(rw,nosuid,nodev,noexec,relatime,seclabel,net_cls,net_prio)
cgroup on /sys/fs/cgroup/hugetlb type cgroup
(rw,nosuid,nodev,noexec,relatime,seclabel,hugetlb)
cgroup on /sys/fs/cgroup/memory type cgroup
(rw,nosuid,nodev,noexec,relatime,seclabel,memory)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,blkio)
cgroup on /sys/fs/cgroup/devices type cgroup
(rw,nosuid,nodev,noexec,relatime,seclabel,devices)
cgroup on /sys/fs/cgroup/misc type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,misc)
cgroup on /sys/fs/cgroup/freezer type cgroup
(rw,nosuid,nodev,noexec,relatime,seclabel,freezer)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,rdma)
```

Les systèmes de fichiers **cgroups-v1** correspondant aux différents contrôleurs **cgroup-v1** ont été montés avec succès dans le répertoire **/sys/fs/cgroup/**.

2. Il est possible d'inspecter le contenu du répertoire **/sys/fs/cgroup/**:

```
# ll /sys/fs/cgroup/
dr-xr-xr-x. 10 root root  0 Mar 16 09:34 blkio
lrwxrwxrwx.  1 root root 11 Mar 16 09:34 cpu → cpu,cpuacct
lrwxrwxrwx.  1 root root 11 Mar 16 09:34 cpuacct → cpu,cpuacct
dr-xr-xr-x. 10 root root  0 Mar 16 09:34 cpu,cpuacct
dr-xr-xr-x.  2 root root  0 Mar 16 09:34 cpuset
dr-xr-xr-x. 10 root root  0 Mar 16 09:34 devices
dr-xr-xr-x.  2 root root  0 Mar 16 09:34 freezer
```

```
dr-xr-xr-x.  2 root root  0 Mar 16 09:34 hugetlb
dr-xr-xr-x. 10 root root  0 Mar 16 09:34 memory
dr-xr-xr-x.  2 root root  0 Mar 16 09:34 misc
lrwxrwxrwx.  1 root root 16 Mar 16 09:34 net_cls → net_cls,net_prio
dr-xr-xr-x.  2 root root  0 Mar 16 09:34 net_cls,net_prio
lrwxrwxrwx.  1 root root 16 Mar 16 09:34 net_prio → net_cls,net_prio
dr-xr-xr-x.  2 root root  0 Mar 16 09:34 perf_event
dr-xr-xr-x. 10 root root  0 Mar 16 09:34 pids
dr-xr-xr-x.  2 root root  0 Mar 16 09:34 rdma
dr-xr-xr-x. 11 root root  0 Mar 16 09:34 systemd
```

Le répertoire **/sys/fs/cgroup/**, également appelé *root control group*, par défaut, contient des répertoires spécifiques aux contrôleurs, tels que **cpuset**. En outre, il existe des répertoires liés à **systemd**.

Ressources supplémentaires

- Comprendre les groupes de contrôle
- Que sont les contrôleurs de ressources du noyau ?
- **cgroups(7)**, **sysfs(5)** pages de manuel
- cgroup-v2 activé par défaut dans RHEL 9

20.4. FIXER DES LIMITES DE CPU AUX APPLICATIONS EN UTILISANT CGROUPS-V1

Il arrive qu'une application consomme beaucoup de temps processeur, ce qui peut avoir un impact négatif sur la santé globale de votre environnement. Utilisez le système de fichiers virtuel **/sys/fs/** pour configurer des limites de CPU pour une application utilisant *control groups version 1* (**cgroups-v1**).

Conditions préalables

- Vous disposez des droits d'accès à la racine.
- Vous disposez d'une application dont vous souhaitez limiter la consommation de l'unité centrale.
- Vous avez configuré le système pour qu'il monte **cgroups-v1** par défaut lors du démarrage du système par le système **systemd** et le gestionnaire de services :

```
# grubby --update-kernel=/boot/vmlinuz-$(uname -r) --
args="systemd.unified_cgroup_hierarchy=0
systemd.legacy_systemd_cgroup_controller"
```

Cette opération ajoute les paramètres de ligne de commande du noyau nécessaires à l'entrée de démarrage actuelle.

Procédure

1. Identifiez l'ID du processus (PID) de l'application dont vous souhaitez limiter la consommation de CPU :

```
# top
```

```
top - 11:34:09 up 11 min,  1 user,  load average: 0.51, 0.27, 0.22
Tasks: 267 total,   3 running, 264 sleeping,   0 stopped,   0 zombie
%Cpu(s): 49.0 us,  3.3 sy,  0.0 ni, 47.5 id,  0.0 wa,  0.2 hi,  0.0 si,  0.0 st
MiB Mem :   1826.8 total,    303.4 free,   1046.8 used,    476.5 buff/cache
MiB Swap:   1536.0 total,   1396.0 free,    140.0 used.    616.4 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 6955 root      20   0  228440   1752   1472 R  99.3   0.1   0:32.71 sha1sum
 5760 jdoe      20   0 3603868 205188  64196 S   3.7  11.0   0:17.19 gnome-shell
 6448 jdoe      20   0  743648  30640  19488 S   0.7   1.6   0:02.73 gnome-terminal-
    1 root      20   0  245300   6568   4116 S   0.3   0.4   0:01.87 systemd
  505 root      20   0       0      0      0 I   0.3   0.0   0:00.75 kworker/u4:4-events_unbound
...
```

L'exemple de sortie du programme **top** révèle que **PID 6955** (application illustrative **sha1sum**) consomme beaucoup de ressources de l'unité centrale.

2. Créez un sous-répertoire dans le répertoire du contrôleur de ressources **cpu**:

```
# mkdir /sys/fs/cgroup/cpu/Example/
```

Le répertoire ci-dessus représente un groupe de contrôle, dans lequel vous pouvez placer des processus spécifiques et leur appliquer certaines limites de CPU. En même temps, certains fichiers d'interface **cgroups-v1** et des fichiers spécifiques au contrôleur **cpu** seront créés dans le répertoire.

3. Il est possible d'inspecter le groupe de contrôle nouvellement créé :

```
# ll /sys/fs/cgroup/cpu/Example/
-rw-r—r--. 1 root root 0 Mar 11 11:42 cgroup.clone_children
-rw-r—r--. 1 root root 0 Mar 11 11:42 cgroup.procs
-r—r—r--. 1 root root 0 Mar 11 11:42 cpuacct.stat
-rw-r—r--. 1 root root 0 Mar 11 11:42 cpuacct.usage
-r—r—r--. 1 root root 0 Mar 11 11:42 cpuacct.usage_all
-r—r—r--. 1 root root 0 Mar 11 11:42 cpuacct.usage_percpu
-r—r—r--. 1 root root 0 Mar 11 11:42 cpuacct.usage_percpu_sys
-r—r—r--. 1 root root 0 Mar 11 11:42 cpuacct.usage_percpu_user
-r—r—r--. 1 root root 0 Mar 11 11:42 cpuacct.usage_sys
-r—r—r--. 1 root root 0 Mar 11 11:42 cpuacct.usage_user
-rw-r—r--. 1 root root 0 Mar 11 11:42 cpu.cfs_period_us
-rw-r—r--. 1 root root 0 Mar 11 11:42 cpu.cfs_quota_us
-rw-r—r--. 1 root root 0 Mar 11 11:42 cpu.rt_period_us
-rw-r—r--. 1 root root 0 Mar 11 11:42 cpu.rt_runtime_us
-rw-r—r--. 1 root root 0 Mar 11 11:42 cpu.shares
-r—r—r--. 1 root root 0 Mar 11 11:42 cpu.stat
-rw-r—r--. 1 root root 0 Mar 11 11:42 notify_on_release
-rw-r—r--. 1 root root 0 Mar 11 11:42 tasks
```

L'exemple de sortie montre des fichiers, tels que **cpuacct.usage**, **cpu.cfs._period_us**, qui représentent des configurations et/ou des limites spécifiques, qui peuvent être définies pour les processus dans le groupe de contrôle **Example**. Notez que les noms de fichiers respectifs sont précédés du nom du contrôleur du groupe de contrôle auquel ils appartiennent.

Par défaut, le groupe de contrôle nouvellement créé hérite de l'accès à l'ensemble des ressources de l'unité centrale du système, sans limite.

4. Configurer les limites de CPU pour le groupe de contrôle :

```
# echo "1000000" > /sys/fs/cgroup/cpu/Example/cpu.cfs_period_us
# echo "200000" > /sys/fs/cgroup/cpu/Example/cpu.cfs_quota_us
```

Le fichier **cpu.cfs_period_us** représente une période de temps en microsecondes (µs, représentée ici par "us") pour la fréquence à laquelle l'accès d'un groupe de contrôle aux ressources de l'unité centrale doit être réattribué. La limite supérieure est de 1 seconde et la limite inférieure de 1000 microsecondes.

Le fichier **cpu.cfs_quota_us** représente la durée totale en microsecondes pendant laquelle tous les processus d'un groupe de contrôle peuvent s'exécuter au cours d'une période (telle que définie par **cpu.cfs_period_us**). Dès que les processus d'un groupe de contrôle, au cours d'une période unique, utilisent la totalité du temps spécifié par le quota, ils sont bridés pour le reste de la période et ne sont plus autorisés à s'exécuter jusqu'à la période suivante. La limite inférieure est de 1000 microsecondes.

Les exemples de commandes ci-dessus définissent les limites de temps de l'unité centrale de sorte que tous les processus du groupe de contrôle **Example** ne puissent s'exécuter que pendant 0,2 seconde (définie par **cpu.cfs_quota_us**) sur 1 seconde (définie par **cpu.cfs_period_us**).

5. Il est possible de vérifier les limites :

```
# cat /sys/fs/cgroup/cpu/Example/cpu.cfs_period_us
/sys/fs/cgroup/cpu/Example/cpu.cfs_quota_us
1000000
200000
```

6. Ajouter le PID de l'application au groupe de contrôle **Example**:

```
# echo "6955" > /sys/fs/cgroup/cpu/Example/cgroup.procs

or

# echo "6955" > /sys/fs/cgroup/cpu/Example/tasks
```

La commande précédente garantit qu'une application souhaitée devient membre du groupe de contrôle **Example** et ne dépasse donc pas les limites de CPU configurées pour le groupe de contrôle **Example**. Le PID doit représenter un processus existant dans le système. L'adresse **PID 6955** a été attribuée au processus **sha1sum /dev/zero &**, utilisé pour illustrer le cas d'utilisation du contrôleur **cpu**.

7. Vérifiez que l'application s'exécute dans le groupe de contrôle spécifié :

```
# cat /proc/6955/cgroup
12:cpuset:/
11:hugetlb:/
10:net_cls,net_prio:/
9:memory:/user.slice/user-1000.slice/user@1000.service
8:devices:/user.slice
7:blkio:/
6:freezer:/
5:rdma:/
4:pids:/user.slice/user-1000.slice/user@1000.service
```

```
3:perf_event:/
2:cpu,cpuacct:/Example
1:name=systemd:/user.slice/user-1000.slice/user@1000.service/gnome-terminal-
server.service
```

L'exemple ci-dessus montre que le processus de l'application souhaitée s'exécute dans le groupe de contrôle **Example**, qui applique des limites de CPU au processus de l'application.

8. Identifiez la consommation actuelle de l'unité centrale de votre application limitée :

```
# top
top - 12:28:42 up  1:06,  1 user,  load average: 1.02, 1.02, 1.00
Tasks: 266 total,   6 running, 260 sleeping,   0 stopped,   0 zombie
%Cpu(s): 11.0 us,  1.2 sy,  0.0 ni, 87.5 id,  0.0 wa,  0.2 hi,  0.0 si,  0.2 st
MiB Mem :   1826.8 total,    287.1 free,   1054.4 used,    485.3 buff/cache
MiB Swap:   1536.0 total,   1396.7 free,    139.2 used.    608.3 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 6955 root      20   0  228440   1752   1472 R  20.6   0.1  47:11.43 sha1sum
 5760 jdoe      20   0 3604956 208832  65316 R   2.3  11.2   0:43.50 gnome-shell
 6448 jdoe      20   0  743836  31736  19488 S   0.7   1.7   0:08.25 gnome-terminal-
  505 root      20   0       0      0      0 I   0.3   0.0   0:03.39 kworker/u4:4-events_unbound
 4217 root      20   0   74192   1612   1320 S   0.3   0.1   0:01.19 spice-vdagentd
...
```

Remarquez que la consommation de l'unité centrale du site **PID 6955** est passée de 99 % à 20 %.

IMPORTANT

La contrepartie de **cgroups-v2** pour **cpu.cfs_period_us** et **cpu.cfs_quota_us** est le fichier **cpu.max**. Le fichier **cpu.max** est disponible via le contrôleur **cpu**.

Ressources supplémentaires

- Comprendre les groupes de contrôle
- Ce que sont les contrôleurs de ressources du noyau
- **cgroups(7)**, **sysfs(5)** pages de manuel

CHAPITRE 21. ANALYSE DES PERFORMANCES DU SYSTÈME AVEC BPF COMPILER COLLECTION

En tant qu'administrateur système, vous pouvez utiliser la bibliothèque BPF Compiler Collection (BCC) pour créer des outils d'analyse des performances de votre système d'exploitation Linux et recueillir des informations qui pourraient être difficiles à obtenir par d'autres interfaces.

21.1. INSTALLATION DU PAQUETAGE BCC-TOOLS

Installez le paquetage **bcc-tools**, qui installe également la bibliothèque BPF Compiler Collection (BCC) en tant que dépendance.

Procédure

1. Installer **bcc-tools**.

   ```
   # dnf install bcc-tools
   ```

   Les outils BCC sont installés dans le répertoire **/usr/share/bcc/tools/**.

2. Optionnellement, inspecter les outils :

   ```
   # ll /usr/share/bcc/tools/
   ...
   -rwxr-xr-x. 1 root root  4198 Dec 14 17:53 dcsnoop
   -rwxr-xr-x. 1 root root  3931 Dec 14 17:53 dcstat
   -rwxr-xr-x. 1 root root 20040 Dec 14 17:53 deadlock_detector
   -rw-r--r--. 1 root root  7105 Dec 14 17:53 deadlock_detector.c
   drwxr-xr-x. 3 root root  8192 Mar 11 10:28 doc
   -rwxr-xr-x. 1 root root  7588 Dec 14 17:53 execsnoop
   -rwxr-xr-x. 1 root root  6373 Dec 14 17:53 ext4dist
   -rwxr-xr-x. 1 root root 10401 Dec 14 17:53 ext4slower
   ...
   ```

   Le répertoire **doc** de la liste ci-dessus contient la documentation de chaque outil.

21.2. UTILISATION DE CERTAINS OUTILS BCC POUR L'ANALYSE DES PERFORMANCES

Utilisez certains programmes prédéfinis de la bibliothèque BPF Compiler Collection (BCC) pour analyser efficacement et en toute sécurité les performances du système au cas par cas. L'ensemble des programmes pré-créés de la bibliothèque BCC peut servir d'exemple pour la création de programmes supplémentaires.

Conditions préalables

- Installation du paquet bcc-tools
- Autorisations de la racine

Utilisation de execsnoop pour examiner les processus du système

1. Exécutez le programme **execsnoop** dans un terminal :

```
# /usr/share/bcc/tools/execsnoop
```

2. Dans un autre terminal, par exemple :

```
$ ls /usr/share/bcc/tools/doc/
```

Ce qui précède crée un processus éphémère de la commande **ls**.

3. Le terminal exécutant **execsnoop** affiche une sortie similaire à la suivante :

```
PCOMM PID    PPID   RET ARGS
ls    8382   8287     0 /usr/bin/ls --color=auto /usr/share/bcc/tools/doc/
...
```

Le programme **execsnoop** imprime une ligne de sortie pour chaque nouveau processus, ce qui consomme des ressources système. Il détecte même les processus de programmes qui s'exécutent très brièvement, tels que **ls**, et que la plupart des outils de surveillance n'enregistreraient pas.

Le site **execsnoop** affiche les champs suivants :

- **PCOMM** - Le nom du processus parent. (**ls**)
- **PID** - L'ID du processus. (**8382**)
- **PPID** - L'ID du processus parent. (**8287**)
- **RET** - La valeur de retour de l'appel système **exec()** (**0**), qui charge le code du programme dans de nouveaux processus.
- **ARGS** - Emplacement du programme lancé avec les arguments.

Pour plus de détails, d'exemples et d'options concernant **execsnoop**, consultez le fichier **/usr/share/bcc/tools/doc/execsnoop_example.txt**.

Pour plus d'informations sur **exec()**, voir les pages du manuel **exec(3)**.

Utiliser opensnoop pour savoir quels fichiers une commande ouvre

1. Exécutez le programme **opensnoop** dans un terminal :

```
# /usr/share/bcc/tools/opensnoop -n uname
```

Ce qui précède imprime la sortie des fichiers qui ne sont ouverts que par le processus de la commande **uname**.

2. Dans un autre terminal, entrez :

```
$ uname
```

La commande ci-dessus ouvre certains fichiers, qui sont capturés à l'étape suivante.

3. Le terminal exécutant **opensnoop** affiche une sortie similaire à la suivante :

```
PID    COMM  FD ERR PATH
```

```
8596   uname  3  0   /etc/ld.so.cache
8596   uname  3  0   /lib64/libc.so.6
8596   uname  3  0   /usr/lib/locale/locale-archive
...
```

Le programme **opensnoop** surveille l'appel système **open()** sur l'ensemble du système et imprime une ligne de sortie pour chaque fichier que **uname** a essayé d'ouvrir en cours de route.

Le site **opensnoop** affiche les champs suivants :

- **PID** - L'ID du processus. (**8596**)
- **COMM** - Le nom du processus. (**uname**)
- **FD** - Le descripteur de fichier - une valeur que **open()** renvoie pour faire référence au fichier ouvert. (**3**)
- **ERR** - Erreurs éventuelles.
- **PATH** - Emplacement des fichiers que **open()** a tenté d'ouvrir.
  Si une commande tente de lire un fichier inexistant, la colonne **FD** renvoie **-1** et la colonne **ERR** imprime une valeur correspondant à l'erreur en question. Par conséquent, **opensnoop** peut vous aider à identifier une application qui ne se comporte pas correctement.

Pour plus de détails, d'exemples et d'options concernant **opensnoop**, consultez le fichier **/usr/share/bcc/tools/doc/opensnoop_example.txt**.

Pour plus d'informations sur **open()**, voir les pages du manuel **open(2)**.

Utilisation de biotop pour examiner les opérations d'E/S sur le disque

1. Exécutez le programme **biotop** dans un terminal :

```
# /usr/share/bcc/tools/biotop 30
```

Cette commande vous permet de surveiller les principaux processus qui effectuent des opérations d'entrée/sortie sur le disque. L'argument garantit que la commande produira un résumé de 30 secondes.

NOTE

Si aucun argument n'est fourni, l'écran de sortie est rafraîchi par défaut toutes les 1 secondes.

2. Dans un autre terminal, entrez, par exemple, :

```
# dd if=/dev/vda of=/dev/zero
```

La commande ci-dessus lit le contenu du disque dur local et écrit la sortie dans le fichier **/dev/zero**. Cette étape génère un certain trafic d'E/S pour illustrer **biotop**.

3. Le terminal exécutant **biotop** affiche une sortie similaire à la suivante :

```
PID    COMM             D MAJ MIN DISK       I/O  Kbytes     AVGms
9568   dd               R 252 0   vda      16294 14440636.0  3.69
```

```
48    kswapd0          W 252 0   vda       1763 120696.0    1.65
7571  gnome-shell      R 252 0   vda        834 83612.0     0.33
1891  gnome-shell      R 252 0   vda       1379 19792.0     0.15
7515  Xorg             R 252 0   vda        280  9940.0     0.28
7579  llvmpipe-1       R 252 0   vda        228  6928.0     0.19
9515  gnome-control-c  R 252 0   vda         62  6444.0     0.43
8112  gnome-terminal-  R 252 0   vda         67  2572.0     1.54
7807  gnome-software   R 252 0   vda         31  2336.0     0.73
9578  awk              R 252 0   vda         17  2228.0     0.66
7578  llvmpipe-0       R 252 0   vda        156  2204.0     0.07
9581  pgrep            R 252 0   vda         58  1748.0     0.42
7531  InputThread      R 252 0   vda         30  1200.0     0.48
7504  gdbus            R 252 0   vda          3  1164.0     0.30
1983  llvmpipe-1       R 252 0   vda         39   724.0     0.08
1982  llvmpipe-0       R 252 0   vda         36   652.0     0.06
...
```

Le site **biotop** affiche les champs suivants :

- **PID** - L'ID du processus. (**9568**)
- **COMM** - Le nom du processus. (**dd**)
- **DISK** - Le disque effectuant les opérations de lecture. (**vda**)
- **I/O** - Nombre d'opérations de lecture effectuées. (16294)
- **Kbytes** - Le nombre de Kbytes atteint par les opérations de lecture. (14,440,636)
- **AVGms** - Le temps d'E/S moyen des opérations de lecture. (3.69)

Pour plus de détails, d'exemples et d'options concernant **biotop**, consultez le fichier **/usr/share/bcc/tools/doc/biotop_example.txt**.

Pour plus d'informations sur **dd**, voir les pages du manuel **dd(1)**.

Utilisation de xfsslower pour révéler les lenteurs inattendues du système de fichiers

1. Exécutez le programme **xfsslower** dans un terminal :

```
# /usr/share/bcc/tools/xfsslower 1
```

La commande ci-dessus mesure le temps que le système de fichiers XFS passe à effectuer des opérations de lecture, d'écriture, d'ouverture ou de synchronisation (**fsync**). L'argument **1** garantit que le programme n'affiche que les opérations qui sont plus lentes que 1 ms.

NOTE

En l'absence d'arguments, **xfsslower** affiche par défaut les opérations plus lentes que 10 ms.

2. Dans un autre terminal, entrez, par exemple, ce qui suit :

```
$ vim text
```

La commande ci-dessus crée un fichier texte dans l'éditeur **vim** afin d'initier certaines interactions avec le système de fichiers XFS.

3. Le terminal qui exécute **xfsslower** affiche quelque chose de similaire après avoir sauvegardé le fichier de l'étape précédente :

```
TIME     COMM           PID    T BYTES   OFF_KB   LAT(ms) FILENAME
13:07:14 b'bash'        4754   R 256     0          7.11 b'vim'
13:07:14 b'vim'         4754   R 832     0          4.03 b'libgpm.so.2.1.0'
13:07:14 b'vim'         4754   R 32      20         1.04 b'libgpm.so.2.1.0'
13:07:14 b'vim'         4754   R 1982    0          2.30 b'vimrc'
13:07:14 b'vim'         4754   R 1393    0          2.52 b'getscriptPlugin.vim'
13:07:45 b'vim'         4754   S 0       0          6.71 b'text'
13:07:45 b'pool'        2588   R 16      0          5.58 b'text'
...
```

Chaque ligne ci-dessus représente une opération dans le système de fichiers qui a pris plus de temps qu'un certain seuil. **xfsslower** est capable d'exposer les problèmes éventuels du système de fichiers, qui peuvent prendre la forme d'opérations inopinément lentes.

Le site **xfsslower** affiche les champs suivants :

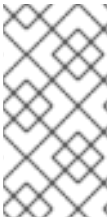
- **COMM** - Le nom du processus. (**b'bash'**)
- **T** - Le type d'opération. (**R**)
  - **R**tête
  - **W**rite
  - **S**ync
- **OFF_KB** - Le décalage du fichier en Ko. (0)
- **FILENAME** - Le fichier en cours de lecture, d'écriture ou de synchronisation.

Pour plus de détails, d'exemples et d'options concernant **xfsslower**, consultez le fichier **/usr/share/bcc/tools/doc/xfsslower_example.txt**.

Pour plus d'informations sur **fsync**, voir les pages du manuel **fsync(2)**.

CHAPITRE 22. RENFORCER LA SÉCURITÉ AVEC LE SOUS-SYSTÈME D'INTÉGRITÉ DU NOYAU

Vous pouvez améliorer la protection de votre système en utilisant les composants du sous-système d'intégrité du noyau. En savoir plus sur les composants concernés et leur configuration.



NOTE

Vous pouvez utiliser les fonctionnalités avec des signatures cryptographiques uniquement pour les produits Red Hat car le système de trousseau de clés du noyau inclut uniquement les certificats pour les clés de signature Red Hat. L'utilisation d'autres fonctions de hachage entraîne une inviolabilité incomplète.

22.1. LE SOUS-SYSTÈME D'INTÉGRITÉ DU NOYAU

Le sous-système d'intégrité est la partie du noyau qui maintient l'intégrité globale des données du système. Ce sous-système permet de conserver l'état d'un système tel qu'il était au moment de sa construction. En utilisant ce sous-système, vous pouvez empêcher toute modification indésirable de fichiers système spécifiques.

Le sous-système d'intégrité du noyau se compose de deux éléments principaux :

Architecture de mesure de l'intégrité (IMA)

- L'IMA mesure le contenu des fichiers à chaque fois qu'ils sont exécutés ou ouverts en les hachant ou en les signant à l'aide de clés cryptographiques. Les clés sont stockées dans le sous-système du trousseau du noyau.
- L'IMA place les valeurs mesurées dans l'espace mémoire du noyau. Cela empêche les utilisateurs du système de modifier les valeurs mesurées.
- L'IMA permet aux parties locales et distantes de vérifier les valeurs mesurées.
- IMA permet de valider localement le contenu actuel des fichiers par rapport aux valeurs précédemment stockées dans la liste de mesures de la mémoire du noyau. Cette extension interdit toute opération sur un fichier spécifique si les mesures actuelles et précédentes ne correspondent pas.

Module de vérification étendu (EVM)

- L'EVM protège les attributs étendus des fichiers (également connus sous le nom de *xattr*) qui sont liés à la sécurité du système, tels que les mesures IMA et les attributs SELinux. L'EVM procède au hachage cryptographique des valeurs correspondantes ou les signe à l'aide de clés cryptographiques. Les clés sont stockées dans le sous-système du trousseau de clés du noyau.

Le sous-système d'intégrité du noyau peut utiliser le Trusted Platform Module (TPM) pour renforcer encore la sécurité du système.

Un TPM est un composant matériel, micrologiciel ou virtuel avec des clés cryptographiques intégrées, qui est construit conformément à la spécification TPM du Trusted Computing Group (TCG) pour les fonctions cryptographiques importantes. Les TPM sont généralement construits comme du matériel

dédié attaché à la carte mère de la plateforme. En fournissant des fonctions cryptographiques à partir d'une zone protégée et inviolable de la puce matérielle, les TPM sont protégés contre les attaques logicielles. Les MPT présentent les caractéristiques suivantes :

- Générateur de nombres aléatoires
- Générateur et stockage sécurisé de clés cryptographiques
- Générateur de hachage
- Attestation à distance

Ressources supplémentaires

- Renforcement de la sécurité
- Configuration de base et avancée de Security-Enhanced Linux (SELinux)

22.2. DES CLÉS FIABLES ET CRYPTÉES

Trusted keys et *encrypted keys* sont des éléments importants du renforcement de la sécurité des systèmes.

Les clés de confiance et les clés cryptées sont des clés symétriques de longueur variable générées par le noyau qui utilisent le service de trousseau du noyau. L'intégrité des clés peut être vérifiée, ce qui signifie qu'elles peuvent être utilisées, par exemple, par le module de vérification étendu (EVM) pour vérifier et confirmer l'intégrité d'un système en cours d'exécution. Les programmes de niveau utilisateur ne peuvent accéder aux clés que sous la forme d'informations chiffrées (*blobs*).

Clés de confiance

Les clés de confiance nécessitent la puce du module de plate-forme de confiance (TPM), qui est utilisée pour créer et crypter (sceller) les clés. Chaque TPM possède une clé d'enveloppement principale, appelée clé racine de stockage, qui est stockée dans le TPM lui-même.

NOTE

RHEL 9 ne prend en charge que TPM 2.0. Si vous devez utiliser TPM 1.2, utilisez RHEL 8. Pour plus d'informations, consultez la solution Le Trusted Platform Module (TPM) est-il pris en charge par Red Hat ?

Vous pouvez vérifier qu'une puce TPM 2.0 a été activée en entrant la commande suivante :

```
$ cat /sys/class/tpm/tpm0/tpm_version_major
2
```

Vous pouvez également activer une puce TPM 2.0 et gérer le dispositif TPM 2.0 par le biais de paramètres dans le micrologiciel de la machine.

En outre, vous pouvez sceller les clés de confiance avec un ensemble spécifique de valeurs du TPM *platform configuration register* (PCR). Le PCR contient un ensemble de valeurs de gestion de l'intégrité qui reflètent le micrologiciel, le chargeur de démarrage et le système d'exploitation. Cela signifie que les clés scellées par PCR ne peuvent être décryptées par le TPM que sur le même système que celui sur lequel elles ont été cryptées. Toutefois, lorsqu'une clé de confiance scellée par PCR est chargée (ajoutée à un trousseau) et que les valeurs PCR qui lui sont associées sont vérifiées,

elle peut être mise à jour avec de nouvelles (ou futures) valeurs PCR, de sorte qu'un nouveau noyau, par exemple, puisse être démarré. Vous pouvez également enregistrer une seule clé sous la forme de plusieurs blobs, chacun avec une valeur PCR différente.

Clés cryptées

Les clés chiffrées ne nécessitent pas de MPT, car elles utilisent la norme de chiffrement avancé (AES) du noyau, ce qui les rend plus rapides que les clés de confiance. Les clés chiffrées sont créées à l'aide de nombres aléatoires générés par le noyau et chiffrées par *master key* lorsqu'elles sont exportées dans des blobs de l'espace utilisateur.

La clé principale est soit une clé de confiance, soit une clé d'utilisateur. Si la clé principale n'est pas de confiance, la clé cryptée n'est aussi sûre que la clé utilisateur utilisée pour la crypter.

22.3. TRAVAILLER AVEC DES CLÉS DE CONFIANCE

Vous pouvez améliorer la sécurité du système en utilisant l'utilitaire **keyctl** pour créer, exporter, charger et mettre à jour les clés de confiance.

Conditions préalables

- Le module de plate-forme de confiance (TPM) est activé et actif. Voir Sous-système d'intégrité du noyau et Clés de confiance et chiffrées .
  Vous pouvez vérifier que votre système dispose d'un TPM en entrant la commande **tpm2_pcrread**. Si la sortie de cette commande affiche plusieurs hachages, vous disposez d'un TPM.

Procédure

1. Créez une clé RSA de 2048 bits avec une clé de stockage primaire SHA-256 avec une poignée persistante, par exemple, *81000001*, en utilisant l'un des utilitaires suivants :

   a. En utilisant le paquet **tss2**:

   ```
   # TPM_DEVICE=/dev/tpm0 tsscreateprimary -hi o -st
   Handle 80000000
   # TPM_DEVICE=/dev/tpm0 tssevictcontrol -hi o -ho 80000000 -hp 81000001
   ```

   b. En utilisant le paquet **tpm2-tools**:

   ```
   # tpm2_createprimary --key-algorithm=rsa2048 --key-context=key.ctxt
   name-alg:
     value: sha256
     raw: 0xb
   …
   sym-keybits: 128
   rsa: xxxxxx…

   # tpm2_evictcontrol -c key.ctxt 0x81000001
   persistentHandle: 0x81000001
   action: persisted
   ```

2. Créez une clé de confiance en utilisant un TPM 2.0 avec la syntaxe suivante **keyctl add trusted *<NAME>* "new *<KEY_LENGTH>* keyhandle=*<PERSISTENT-HANDLE>* [options]" *<KEYRING>***. Dans cet exemple, la poignée persistante est *81000001*.

```
# keyctl add trusted kmk "new 32 keyhandle=0x81000001" @u
642500861
```

La commande crée une clé de confiance appelée **kmk** d'une longueur de **32** octets (256 bits) et la place dans le trousseau de l'utilisateur (**@u**). Les clés peuvent avoir une longueur de 32 à 128 octets (256 à 1024 bits).

3. Liste la structure actuelle des trousseaux de clés du noyau.

```
# keyctl show
Session Keyring
       -3 --alswrv    500   500  keyring: ses 97833714 --alswrv 500 -1 \ keyring: uid.1000
642500861 --alswrv 500 500 \ trusted: kmk
```

4. Exporter la clé vers un blob de l'espace utilisateur en utilisant le numéro de série de la clé de confiance.

```
# keyctl pipe 642500861 > kmk.blob
```

La commande utilise la sous-commande **pipe** et le numéro de série **kmk**.

5. Charger la clé de confiance à partir du blob de l'espace utilisateur.

```
# keyctl add trusted kmk "load `cat kmk.blob`" @u
268728824
```

6. Créez des clés cryptées sécurisées qui utilisent la clé de confiance scellée par le TPM (**kmk**). Suivez la syntaxe suivante : *keyctl add encrypted <NAME> "new [FORMAT] <KEY_TYPE>: <PRIMARY_KEY_NAME> <KEY_LENGTH>" <KEYRING>*.

```
# keyctl add encrypted encr-key "new trusted:kmk 32" @u
159771175
```

Ressources supplémentaires

- la page du manuel **keyctl(1)**
- Des clés fiables et cryptées
- Service de conservation des clés du noyau
- Le sous-système d'intégrité du noyau

22.4. TRAVAILLER AVEC DES CLÉS CRYPTÉES

Vous pouvez améliorer la sécurité du système sur les systèmes qui ne disposent pas d'un module de plate-forme de confiance (TPM) en gérant des clés cryptées.

Procédure

1. Générer une clé d'utilisateur en utilisant une séquence aléatoire de nombres.

```
# keyctl add user kmk-user "$(dd if=/dev/urandom bs=1 count=32 2>/dev/null)" @u
427069434
```

La commande génère une clé d'utilisateur appelée **kmk-user** qui fait office de *primary key* et est utilisée pour sceller les clés cryptées proprement dites.

2. Générer une clé cryptée en utilisant la clé primaire de l'étape précédente :

```
# keyctl add encrypted encr-key "new user:kmk-user 32" @u
1012412758
```

3. Optionnellement, liste de toutes les clés du trousseau de l'utilisateur spécifié :

```
# keyctl list @u
2 keys in keyring:
427069434: --alswrv  1000  1000 user: kmk-user
1012412758: --alswrv  1000  1000 encrypted: encr-key
```

IMPORTANT

Les clés chiffrées qui ne sont pas scellées par une clé primaire de confiance ne sont aussi sûres que la clé primaire de l'utilisateur (clé de nombres aléatoires) qui a été utilisée pour les chiffrer. Par conséquent, chargez la clé primaire de l'utilisateur de la manière la plus sûre possible et, de préférence, au début du processus de démarrage.

Ressources supplémentaires

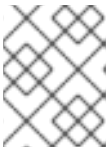
- La page du manuel **keyctl(1)**
- Service de conservation des clés du noyau

22.5. PERMETTRE L'IMA ET L'EVM

Vous pouvez activer et configurer l'architecture de mesure de l'intégrité (IMA) et le module de vérification étendu (EVM) pour améliorer la sécurité du système d'exploitation.

Conditions préalables

- Secure Boot est temporairement désactivé.



NOTE

Lorsque l'option Secure Boot est activée, le paramètre de ligne de commande **ima_appraise=fix** kernel ne fonctionne pas.

- Le système de fichiers **securityfs** est monté sur le répertoire **/sys/kernel/security/** et le répertoire **/sys/kernel/security/integrity/ima/** existe. Vous pouvez vérifier où **securityfs** est monté en utilisant la commande **mount**:

```
# mount
...
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
...
```

- Le gestionnaire de services **systemd** est corrigé pour prendre en charge l'IMA et l'EVM au démarrage. Vous pouvez le vérifier en utilisant la commande suivante :

```
# dmesg | grep -i -e EVM -e IMA
[    0.000000] Command line: BOOT_IMAGE=(hd0,msdos1)/vmlinuz-5.14.0-1.el9.x86_64
root=/dev/mapper/rhel-root ro crashkernel=1G-4G:192M,4G-64G:256M,64G-:512M
resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb quiet
[    0.000000] kvm-clock: cpu 0, msr 23601001, primary cpu clock
[    0.000000] Using crashkernel=1G-4G:192M,4G-64G:256M,64G-:512M, the size chosen is
a best effort estimation.
[    0.000000] Kernel command line: BOOT_IMAGE=(hd0,msdos1)/vmlinuz-5.14.0-
1.el9.x86_64 root=/dev/mapper/rhel-root ro crashkernel=1G-4G:192M,4G-64G:256M,64G-
:512M resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb quiet
[    0.911527] ima: No TPM chip found, activating TPM-bypass!
[    0.911538] ima: Allocated hash algorithm: sha1
[    0.911580] evm: Initialising EVM extended attributes:
[    0.911581] evm: security.selinux
[    0.911581] evm: security.ima
[    0.911582] evm: security.capability
[    0.911582] evm: HMAC attrs: 0x1
[    1.715151] systemd[1]: systemd 239 running in system mode. (+PAM +AUDIT +SELINUX
+IMA -APPARMOR +SMACK +SYSVINIT +UTMP +LIBCRYPTSETUP +GCRYPT
+GNUTLS +ACL +XZ +LZ4 +SECCOMP +BLKID +ELFUTILS +KMOD +IDN2 -IDN +PCRE2
default-hierarchy=legacy)
[    3.824198] fbcon: qxldrmfb (fb0) is primary device
[    4.673457] PM: Image not found (code -22)
[    6.549966] systemd[1]: systemd 239 running in system mode. (+PAM +AUDIT +SELINUX
+IMA -APPARMOR +SMACK +SYSVINIT +UTMP +LIBCRYPTSETUP +GCRYPT
+GNUTLS +ACL +XZ +LZ4 +SECCOMP +BLKID +ELFUTILS +KMOD +IDN2 -IDN +PCRE2
default-hierarchy=legacy)
```

Procédure

1. Activez l'IMA et l'EVM dans le mode *fix* pour l'entrée de démarrage actuelle et permettez aux utilisateurs de recueillir et de mettre à jour les mesures IMA en ajoutant les paramètres de ligne de commande du noyau suivants :

```
# grubby --update-kernel=/boot/vmlinuz-$(uname -r) --args="ima_policy=appraise_tcb
ima_appraise=fix evm=fix"
```

   La commande active l'IMA et l'EVM dans le mode *fix* pour l'entrée de démarrage actuelle et permet aux utilisateurs de rassembler et de mettre à jour les mesures IMA.

   Le paramètre de ligne de commande du noyau **ima_policy=appraise_tcb** garantit que le noyau utilise la politique de mesure par défaut de la Trusted Computing Base (TCB) et l'étape d'évaluation. L'étape d'évaluation interdit l'accès aux fichiers dont les mesures antérieures et actuelles ne correspondent pas.

2. Redémarrez pour que les modifications soient prises en compte.

3. Facultatif : Vérifiez que les paramètres ont été ajoutés à la ligne de commande du noyau :

```
# cat /proc/cmdline
BOOT_IMAGE=(hd0,msdos1)/vmlinuz-5.14.0-1.el9.x86_64 root=/dev/mapper/rhel-root ro
```

```
crashkernel=1G-4G:192M,4G-64G:256M,64G-:512M resume=/dev/mapper/rhel-swap
rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb quiet ima_policy=appraise_tcb ima_appraise=fix
evm=fix
```

4. Créer une clé maître du noyau pour protéger la clé EVM :

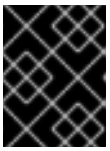
```
# keyctl add user kmk "$(dd if=/dev/urandom bs=1 count=32 2> /dev/null)" @u
748544121
```

L'adresse **kmk** est entièrement conservée dans la mémoire du noyau. La valeur de 32 octets de **kmk** est générée à partir d'octets aléatoires du fichier **/dev/urandom** et placée dans le trousseau de l'utilisateur (**@u**). Le numéro de série de la clé se trouve sur la première ligne de la sortie précédente.

5. Créer une clé EVM cryptée sur la base du site **kmk**:

```
# keyctl add encrypted evm-key "new user:kmk 64" @u
641780271
```

La commande utilise le site **kmk** pour générer et crypter une clé d'utilisateur de 64 octets (nommée **evm-key**) et la placer dans le trousseau de l'utilisateur (**@u**). Le numéro de série de la clé se trouve sur la première ligne de la sortie précédente.



IMPORTANT

Il est nécessaire de nommer la clé utilisateur **evm-key** car c'est le nom que le sous-système EVM attend et avec lequel il travaille.

6. Créer un répertoire pour les clés exportées.

```
# mkdir -p /etc/keys/
```

7. Recherchez le site **kmk** et exportez sa valeur en clair dans le nouveau répertoire.

```
# keyctl pipe $(keyctl search @u user kmk) > /etc/keys/kmk
```

8. Recherchez le site **evm-key** et exportez sa valeur cryptée dans le nouveau répertoire.

```
# keyctl pipe $(keyctl search @u encrypted evm-key) > /etc/keys/evm-key
```

Le site **evm-key** a été crypté précédemment par la clé principale du noyau.

9. Optionnel : Affichez les clés nouvellement créées.

```
# keyctl show
Session Keyring
974575405   --alswrv     0        0      keyring: ses 299489774 --alswrv 0 65534 \ keyring: uid.0
748544121 --alswrv 0 0 \ user: kmk
641780271   --alswrv     0        0           \_ encrypted: evm-key

# ls -l /etc/keys/
```

```
total 8
-rw-r--r--. 1 root root 246 Jun 24 12:44 evm-key
-rw-r--r--. 1 root root  32 Jun 24 12:43 kmk
```

10. Facultatif : si les clés ont été supprimées du trousseau, par exemple après un redémarrage du système, vous pouvez importer les clés déjà exportées **kmk** et **evm-key** au lieu d'en créer de nouvelles.

    a. Importer le site **kmk**.

    ```
    # keyctl add user kmk "$(cat /etc/keys/kmk)" @u
    451342217
    ```

    b. Importer le site **evm-key**.

    ```
    # keyctl add encrypted evm-key "load $(cat /etc/keys/evm-key)" @u
    924537557
    ```

11. Activer l'EVM.

    ```
    # echo 1 > /sys/kernel/security/evm
    ```

12. Réétiqueter l'ensemble du système.

    ```
    # find / -fstype xfs -type f -uid 0 -exec head -n 1 '{}' >/dev/null \;
    ```

AVERTISSEMENT

L'activation de l'IMA et de l'EVM sans réétiquetage du système peut rendre la majorité des fichiers du système inaccessibles.

Vérification

- Vérifiez que l'EVM a été initialisé.

```
# dmesg | tail -1
[…] evm: key initialized
```

Ressources supplémentaires

- Le sous-système d'intégrité du noyau
- Des clés fiables et cryptées.

22.6. COLLECTE DE HACHAGES DE FICHIERS AVEC UNE ARCHITECTURE DE MESURE DE L'INTÉGRITÉ

Dans la phase *measurement*, vous pouvez créer des hachages de fichiers et les stocker en tant

qu'attributs étendus (*xattrs*) de ces fichiers. Avec les hachages de fichiers, vous pouvez générer une signature numérique basée sur RSA ou un code d'authentification de message basé sur le hachage (HMAC-SHA1) et ainsi empêcher les attaques de falsification hors ligne sur les attributs étendus.

Conditions préalables

- L'IMA et l'EVM sont activés. Pour plus d'informations, voir Activation de l'architecture de mesure de l'intégrité et du module de vérification étendu.
- Une clé de confiance valide ou une clé cryptée est stockée dans le trousseau de clés du noyau.
- Les paquets **ima-evm-utils**, **attr**, et **keyutils** sont installés.

Procédure

1. Créer un fichier de test.

```
# echo <Test_text> > test_file
```

   L'IMA et l'EVM garantissent que le fichier d'exemple **test_file** a des valeurs de hachage attribuées qui sont stockées en tant qu'attributs étendus.

2. Inspecter les attributs étendus du fichier.

```
# getfattr -m . -d test_file
# file: test_file
security.evm=0sAnDIy4VPA0HArpPO/EqiutnNyBql
security.ima=0sAQOEDeuUnWzwwKYk+n66h/vby3eD
```

   L'exemple de sortie montre des attributs étendus avec les valeurs de hachage de l'IMA et de l'EVM et le contexte SELinux. EVM ajoute un attribut étendu **security.evm** lié aux autres attributs. À ce stade, vous pouvez utiliser l'utilitaire **evmctl** sur **security.evm** pour générer une signature numérique basée sur RSA ou un code d'authentification de message basé sur le hachage (HMAC-SHA1).

Ressources supplémentaires

- Renforcement de la sécurité

CHAPITRE 23. CONFIGURATION PERMANENTE DES PARAMÈTRES DU NOYAU À L'AIDE DE KERNEL_SETTINGS RHEL SYSTEM ROLE

Vous pouvez utiliser le rôle **kernel_settings** pour configurer les paramètres du noyau sur plusieurs clients à la fois. Cette solution :

- Fournit une interface conviviale avec des paramètres d'entrée efficaces.
- Conserve tous les paramètres du noyau en un seul endroit.

Après avoir exécuté le rôle **kernel_settings** à partir de la machine de contrôle, les paramètres du noyau sont appliqués immédiatement aux systèmes gérés et persistent à travers les redémarrages.

IMPORTANT

Notez que les rôles système RHEL livrés sur les canaux RHEL sont disponibles pour les clients RHEL sous forme de paquetage RPM dans le référentiel AppStream par défaut. Les rôles système RHEL sont également disponibles sous forme de collection pour les clients ayant des abonnements Ansible via Ansible Automation Hub.

23.1. INTRODUCTION AU RÔLE KERNEL_SETTINGS

RHEL System Roles est un ensemble de rôles qui fournit une interface de configuration cohérente pour gérer à distance plusieurs systèmes.

Les rôles système RHEL ont été introduits pour les configurations automatisées du noyau à l'aide du rôle système **kernel_settings**. Le paquet **rhel-system-roles** contient ce rôle système, ainsi que la documentation de référence.

Pour appliquer les paramètres du noyau sur un ou plusieurs systèmes de manière automatisée, utilisez le rôle **kernel_settings** avec une ou plusieurs de ses variables de rôle de votre choix dans un playbook. Un playbook est une liste d'un ou plusieurs jeux lisibles par l'homme et écrits au format YAML.

Le rôle **kernel_settings** vous permet de configurer :

- Les paramètres du noyau en utilisant la variable de rôle **kernel_settings_sysctl**
- Divers sous-systèmes du noyau, périphériques matériels et pilotes de périphériques utilisant la variable de rôle **kernel_settings_sysfs**
- L'affinité du processeur pour le gestionnaire de service **systemd** et les processus qui en découlent en utilisant la variable de rôle **kernel_settings_systemd_cpu_affinity**
- Le sous-système de mémoire du noyau utilise les variables de rôle **kernel_settings_transparent_hugepages** et **kernel_settings_transparent_hugepages_defrag** pour rendre les hugepages transparentes

Ressources supplémentaires

- **README.md** et **README.html** dans le répertoire **/usr/share/doc/rhel-system-roles/kernel_settings/**
- Travailler avec des playbooks

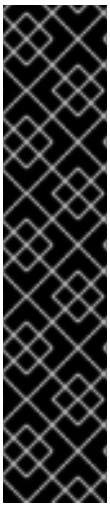
- Comment constituer votre inventaire

23.2. APPLICATION DES PARAMÈTRES SÉLECTIONNÉS DU NOYAU À L'AIDE DU RÔLE KERNEL_SETTINGS

Suivez ces étapes pour préparer et appliquer un playbook Ansible afin de configurer à distance les paramètres du noyau avec un effet persistant sur plusieurs systèmes d'exploitation gérés.

Conditions préalables

- Vous avez les autorisations **root**.
- En vertu de votre abonnement RHEL, vous avez installé les paquets **ansible-core** et **rhel-system-roles** sur la machine de contrôle.
- Un inventaire des hôtes gérés est présent sur la machine de contrôle et Ansible peut s'y connecter.



IMPORTANT

RHEL 8.0 - 8.5 donne accès à un dépôt Ansible séparé qui contient Ansible Engine 2.9 pour l'automatisation basée sur Ansible. Ansible Engine contient des utilitaires de ligne de commande tels que **ansible**, **ansible-playbook**; des connecteurs tels que **docker** et **podman**; et tout l'univers des plugins et des modules. Pour plus d'informations sur la manière d'obtenir et d'installer Ansible Engine, reportez-vous à la section Comment télécharger et installer Red Hat Ansible Engine ?

RHEL 8.6 et 9.0 a introduit Ansible Core (fourni en tant que **ansible-core** RPM), qui contient les utilitaires de ligne de commande Ansible, les commandes et un petit ensemble de plugins Ansible intégrés. Le dépôt AppStream fournit **ansible-core**, qui a une portée de support limitée. Pour en savoir plus, consultez le document Scope of support for the ansible-core package included in the RHEL 9 AppStream.

Procédure

1. Si vous le souhaitez, vous pouvez consulter le fichier **inventory** à des fins d'illustration :

```
# cat /home/jdoe/<ansible_project_name>/inventory
[testingservers]
pdoe@192.168.122.98
fdoe@192.168.122.226

[db-servers]
db1.example.com
db2.example.com

[webservers]
web1.example.com
web2.example.com
192.0.2.42
```

Ce fichier définit le groupe **[testingservers]** et d'autres groupes. Il vous permet d'exécuter Ansible plus efficacement sur un ensemble spécifique de systèmes.

2. Créer un fichier de configuration pour définir les valeurs par défaut et l'escalade des privilèges pour les opérations Ansible.

   a. Créez un nouveau fichier YAML et ouvrez-le dans un éditeur de texte, par exemple :

   ```
   # vi /home/jdoe/<ansible_project_name>/ansible.cfg
   ```

   b. Insérez le contenu suivant dans le fichier :

   ```
   [defaults]
   inventory = ./inventory

   [privilege_escalation]
   become = true
   become_method = sudo
   become_user = root
   become_ask_pass = true
   ```

   La section **[defaults]** indique un chemin d'accès au fichier d'inventaire des hôtes gérés. La section **[privilege_escalation]** définit que les privilèges de l'utilisateur doivent passer à **root** sur les hôtes gérés spécifiés. Ceci est nécessaire pour une configuration réussie des paramètres du noyau. Lorsque le playbook Ansible est exécuté, vous serez invité à saisir le mot de passe de l'utilisateur. L'utilisateur passe automatiquement à **root** par le biais de **sudo** après s'être connecté à un hôte géré.

3. Créer un playbook Ansible qui utilise le rôle **kernel_settings**.

   a. Créez un nouveau fichier YAML et ouvrez-le dans un éditeur de texte, par exemple :

   ```
   # vi /home/jdoe/<ansible_project_name>/kernel-roles.yml
   ```

   Ce fichier représente un playbook et contient généralement une liste ordonnée de tâches, également appelées *plays*, qui sont exécutées contre des hôtes gérés spécifiques sélectionnés dans votre fichier **inventory**.

   b. Insérez le contenu suivant dans le fichier :

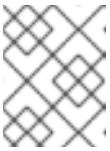
   ```
   ---
   -
     hosts: testingservers
     name: "Configure kernel settings"
     roles:
       - rhel-system-roles.kernel_settings
     vars:
       kernel_settings_sysctl:
         - name: fs.file-max
           value: 400000
         - name: kernel.threads-max
           value: 65536
       kernel_settings_sysfs:
         - name: /sys/class/net/lo/mtu
           value: 65000
       kernel_settings_transparent_hugepages: madvise
   ```

   La clé **name** est facultative. Elle associe une chaîne arbitraire à la pièce en tant qu'étiquette

et identifie l'objet de la pièce. La clé **hosts** de la pièce spécifie les hôtes contre lesquels la pièce est exécutée. La ou les valeurs de cette clé peuvent être fournies sous forme de noms individuels d'hôtes gérés ou de groupes d'hôtes tels que définis dans le fichier **inventory**.

La section **vars** représente une liste de variables contenant les noms des paramètres du noyau sélectionnés et les valeurs auxquelles ils doivent être définis.

La clé **roles** spécifie le rôle du système qui va configurer les paramètres et les valeurs mentionnés dans la section **vars**.



NOTE

Vous pouvez modifier les paramètres du noyau et leurs valeurs dans le playbook en fonction de vos besoins.

4. En option, vérifiez que la syntaxe de votre pièce est correcte.

```
# ansible-playbook --syntax-check kernel-roles.yml

playbook: kernel-roles.yml
```

Cet exemple montre la vérification réussie d'un playbook.

5. Exécutez votre cahier des charges.

```
# ansible-playbook kernel-roles.yml

...

BECOME password:

PLAY [Configure kernel settings]
*********************************************************************************



PLAY RECAP
*********************************************************************************************************************
fdoe@192.168.122.226       : ok=10   changed=4    unreachable=0    failed=0    skipped=6
rescued=0    ignored=0
pdoe@192.168.122.98        : ok=10   changed=4    unreachable=0    failed=0    skipped=6
rescued=0    ignored=0
```

Avant qu'Ansible n'exécute votre playbook, vous allez être invité à saisir votre mot de passe afin qu'un utilisateur sur les hôtes gérés puisse être basculé sur **root**, ce qui est nécessaire pour configurer les paramètres du noyau.

La section récapitulative montre que la lecture s'est terminée avec succès (**failed=0**) pour tous les hôtes gérés, et que 4 paramètres du noyau ont été appliqués (**changed=4**).

6. Redémarrez les hôtes gérés et vérifiez les paramètres du noyau concernés pour vous assurer que les changements ont été appliqués et qu'ils persistent après les redémarrages.

Ressources supplémentaires

- Préparation d'un nœud de contrôle et de nœuds gérés à l'utilisation des rôles système RHEL
- **README.html** et **README.md** dans le répertoire **/usr/share/doc/rhel-system-roles/kernel_settings/**
- Constituez votre inventaire
- Configuration d'Ansible
- Travailler avec des Playbooks
- Utilisation de variables
- Rôles