Red Hat Enterprise Linux 7

カーネル管理ガイド

RHEL で Linux カーネルを管理するためにモジュール、kpatch、または kdump を使用

Last Updated: 2021-09-29

Red Hat Enterprise Linux 7 カーネル管理ガイド

RHEL で Linux カーネルを管理するためにモジュール、kpatch、または kdump を使用

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

カーネル管理ガイドは、Red Hat Enterprise Linux 7 カーネルのメンテナンスタスクについて説明します。今回のリリースでは、kpatch の使用、カーネルモジュールの管理、カーネルの手動更新などの情報が含まれます。

目次

前書き

カーネル管理ガイド では、カーネルの活用方法を説明し、複数の実用的なタスクを紹介します。本書では、カーネルモジュールを使用する情報から始まり、**sysfs** ファシリティーとの対話、カーネルの手動アップグレード、および kpatch の使用を説明します。また、クラッシュダンプメカニズムも紹介します。 これは、カーネルに障害が発生した場合に、**vmcore** コレクションを設定してテストする手順を説明します。

また、**カーネル管理ガイド** では、カーネル管理のユースケースも説明し、コマンドラインオプションの参考資料や、カーネルの設定項目 (スイッチとも呼ばれる)、カーネル機能の概要も紹介します。

第1章 カーネルモジュールの使用

本章では、以下を説明します。

- カーネルモジュールの概要
- **kmod** ユーティリティーを使用して、モジュールとその依存関係を管理する方法
- モジュールパラメーターを設定して、カーネルモジュールの動作を制御する方法
- 起動時に、モジュールを読み込む方法

注記

本章で説明するカーネルモジュールのユーティリティーを使用するには、最初に root で以下を実行して、ご使用のシステムに **kmod** パッケージがインストールされていることを確認します。

```
# yum install kmod
```

1.1. カーネルモジュールの概要

Linux カーネルは、モノリシックとして設計されています。しかし、各ユースケースで必要とされる追加またはオプションのモジュールでコンパイルされています。つまり、動的に読み込まれる **カーネルモジュール** を使用してカーネル機能を拡張することができます。カーネルモジュールは、以下のものを提供できます。

- 新しいハードウェアへのサポートを強化するデバイスドライバー
- **GFS2** や **NFS** などのファイルシステムのサポート

モジュールは、カーネル自体と同様に、その動作をカスタマイズするパラメーターを受けることができます。いずれのパラメーターも、ほとんどの場合に正しく動作します。カーネルモジュールに関連して、ユーザー空間ツールは以下の操作を行うことができます。

- 実行中のカーネルに現在読み込まれているモジュールの一覧表示
- 利用できるパラメーターとモジュール固有の情報に対して、利用可能なすべてのモジュールのクエリーを行います。
- 実行中のカーネルに動的に、または実行中のカーネルからモジュールの読み込みまたはアンロード (削除)。

kmod パッケージにより提供される、このようなユーティリティーの多くは、動作の実行時にモジュールの依存関係を考慮します。そのため、手動で依存関係を追跡する必要はほとんどありません。

最新のシステムでは、必要に応じて、さまざまなメカニズムによりカーネルモジュールが自動的に読み込まれます。ただし、モジュールを手動で読み込むか、削除しないといけない場合もあります。たとえば、どちらのモジュールも基本的な機能は提供できるものの、いずれかのモジュールが好まれる場合か、モジュールが、予期しない動作をしている場合などです。

1.2. カーネルモジュールの依存関係

特定のカーネルモジュールは、複数の他のカーネルモジュールに依存する場合があります。**/lib/modules/<KERNEL_VERSION>/modules.dep** ファイルには、各カーネルバージョンに対するカーネルモジュールの依存関係の完全な一覧が含まれます。

依存関係ファイルは、**kmod** パッケージの一部である **depmod** プログラムにより生成されます。**kmod** によるユーティリティーの多くは、操作を実行する際にモジュールの依存関係を考慮に入れるため、**手動** で依存関係を追跡する必要はほとんどありません。

警告

カーネルモジュールのコードは、制限のないモードのカーネルスペースで実行されます。そのため、読み込むモジュールに注意してください。

関連情報

- **/lib/modules/<KERNEL_VERSION>/modules.dep** の詳細は、**modules.dep(5)** の man ページを参照してください。
- **depmod** の概要やオプションの詳細は、**depmod(8)** の man ページを参照してください。

1.3. 読み込み済みモジュールの一覧表示

lsmod コマンドを実行すると、現在カーネルに読み込み済みの全カーネルモジュールを一覧表示できます。 以下に例を示します。

```
# lsmod
Module                  Size  Used by
tcp_lp                 12663  0
bnep                   19704  2
bluetooth             372662  7 bnep
rfkill                 26536  3 bluetooth
fuse                   87661  3
ebtable_broute         12731  0
bridge                110196  1 ebtable_broute
stp                    12976  1 bridge
llc                    14552  2 stp,bridge
ebtable_filter         12827  0
ebtables               30913  3 ebtable_broute,ebtable_nat,ebtable_filter
ip6table_nat           13015  1
nf_nat_ipv6            13279  1 ip6table_nat
iptable_nat            13011  1
nf_conntrack_ipv4      14862  4
nf_defrag_ipv4         12729  1 nf_conntrack_ipv4
nf_nat_ipv4            13263  1 iptable_nat
nf_nat                 21798  4 nf_nat_ipv4,nf_nat_ipv6,ip6table_nat,iptable_nat
[output truncated]
```

lsmod 出力では、3 つのコラムを表示します。

- **モジュール**

- メモリーに現在読み込まれているカーネルモジュールの名前

- **Size (サイズ)**
  - カーネルモジュールが使用するメモリー量 (キロバイト単位)

- **Used by (使用フィールド)**
  - **Module** フィールドにある依存関係の数を表す小数点
  - 依存する **Module** の名前をコンマ区切りにした文字列。この一覧を使用して、アンロードしたいモジュールに依存しているモジュールすべてを最初にアンロードできます。

最後に、**lsmod** 出力は **/proc/modules** 擬似ファイルの内容ほど詳細ではないので、はるかに読み取りやすくなっている点に留意してください。

1.4. モジュール情報の表示

カーネルモジュールに関する詳しい情報は、**modinfo <MODULE_NAME>** コマンドを使用して表示できます。

注記

カーネルモジュール名を **kmod** ユーティリティーのいずれかの引数として指定する場合には、その名前の末尾に拡張子 **.ko** を付けないでください。カーネルモジュール名には拡張子はありません。ただし、対応するファイルには拡張子があります。

例1.1 lsmod を使用したカーネルモジュール情報の一覧表示

Intel PRO/1000 ネットワークドライバーである **e1000e** モジュールに関する情報を表示するには、**root** で以下のコマンドを入力します。

```
# modinfo e1000e
filename:       /lib/modules/3.10.0-
121.el7.x86_64/kernel/drivers/net/ethernet/intel/e1000e/e1000e.ko
version:        2.3.2-k
license:        GPL
description:    Intel(R) PRO/1000 Network Driver
author:         Intel Corporation,
```

1.5. システムランタイム時のカーネルモジュールの読み込み

Linux カーネルの機能を拡張する最適な方法は、カーネルモジュールを読み込むことです。以下の手順では、**modprobe** コマンドを使用して、カーネルモジュールを検出し、現在実行しているカーネルに読み込む方法を説明します。

前提条件

- root 権限
- **kmod** パッケージがインストールされている。

- 関連のカーネルモジュールが読み込まれていない。これを確認するには、「現在のロード済みモジュールの一覧表示」を参照してください。

手順

1. 読み込むカーネルモジュールを選択します。
   モジュールは **/lib/modules/$(uname -r)/kernel/<SUBSYSTEM>/** ディレクトリーにあります。
2. 関連するカーネルモジュールを読み込みます。

```
# modprobe <MODULE_NAME>
```

注記

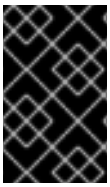
カーネルモジュールの名前を入力する際には、**.ko.xz** 拡張子は名前の末尾に追加しないでください。カーネルモジュール名には拡張子はありません。ただし、対応するファイルには拡張子があります。

3. 必要に応じて、関連モジュールが読み込まれたことを確認します。

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

モジュールが正しく読み込まれた場合、このコマンドは関連するカーネルモジュールを表示します。以下に例を示します。

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



重要

この手順で説明されている変更は、システムを再起動は**維持されません**。システムの再起動後にも**設定を維持する**ようにカーネルモジュールを読み込む方法は、「システムの起動時に自動的にカーネルモジュールを読み込む」を参照してください。

関連情報

- **modprobe** の詳細は、**modprobe(8)** の man ページを参照してください。

1.6. システムランタイム時のカーネルモジュールのアンロード

時折、実行中のカーネルから特定のカーネルモジュールをアンロードする必要性に駆られることがあります。以下の手順では、**modprobe** コマンドを使用して、現在読み込まれているカーネルから、システムの実行時にカーネルモジュールを見つけてアンロードする方法を説明します。

前提条件

- root 権限
- **kmod** パッケージがインストールされている。

手順

1. **lsmod** コマンドを実行して、アンロードするカーネルモジュールを選択します。
カーネルモジュールに依存関係がある場合は、カーネルモジュールをアンロードする前に、これらをアンロードします。依存関係のあるモジュールを特定する方法は、「現在読み込まれているモジュールの一覧表示」および「カーネルモジュールの依存関係」を参照してください。

2. 関連するカーネルモジュールをアンロードします。

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

カーネルモジュールの名前を入力する際には、**.ko.xz** 拡張子は名前の末尾に追加しないでください。カーネルモジュール名には拡張子はありません。ただし、対応するファイルには拡張子があります。

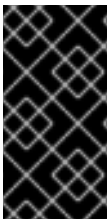
警告

実行中のシステムで使用される場合は、カーネルモジュールをアンロードしないでください。これを行うと、システムが不安定になったり、動作しなくなったりすることがあります。

3. 必要に応じて、関連モジュールがアンロードされたことを確認します。

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

モジュールが正常にアンロードされた場合、このコマンドは出力を表示しません。



重要

この手順を終了すると、システムの起動時に自動的に読み込まれるように定義したカーネルモジュールは、システムを再起動しても**アンロードされません**。この結果を追跡する方法は、「システムの起動時にカーネルモジュールが自動的にロードされないようにする」を参照してください。

関連情報

- **modprobe** の詳細は、**modprobe(8)** の man ページを参照してください。

1.7. システムの起動時に自動的にカーネルモジュールを読み込む

以下の手順では、ブートプロセス中に自動的に読み込まれるようにカーネルモジュールを設定する方法を説明します。

前提条件

- root 権限
- **kmod** パッケージがインストールされている。

手順

1. 起動プロセス中に読み込むカーネルモジュールを選択します。
   モジュールは **/lib/modules/$(uname -r)/kernel/<SUBSYSTEM>/** ディレクトリーにあります。

2. モジュールの設定ファイルを作成します。

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

注記

*カーネルモジュールの名前を入力する際には、**.ko.xz** 拡張子は名前の末尾に追加しないでください。カーネルモジュール名には拡張子はありません。ただし、対応するファイルには拡張子があります。*

3. *必要に応じて、関連モジュールが読み込まれたことを確認します。*

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

上記のコマンド例は成功し、関連するカーネルモジュールを表示します。

重要

*この手順で説明している変更は、システムを再起動しても**持続されます**。*

関連情報

- *システムの起動プロセス中のカーネルモジュールの読み込みの詳細は、**modules-load.d (5)** の man ページを参照してください。*

1.8. システムの起動時にカーネルモジュールが自動的にロードされないようにする

以下の手順では、システムの起動プロセス中にカーネルモジュールが自動的に読み込まれないように拒否リストに追加する方法を説明します。

前提条件

- *root 権限*
- ***kmod** パッケージがインストールされている。*
- *拒否リストに指定したカーネルモジュールが現在のシステム設定に重要でないことを確認する。*

手順

1. *拒否リストに追加するカーネルモジュールを選択します。*

```
$ lsmod

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

```
uinput              20480  1
xt_conntrack        16384  1
…
```

lsmod コマンドは、現在実行中のカーネルに読み込まれているモジュールの一覧を表示します。

- もしくは、読み込みを阻止する、アンロードしたカーネルモジュールを特定します。すべてのカーネルモジュールは、**/lib/modules/<KERNEL_VERSION>/kernel/<subsystem>/** ディレクトリーにあります。

2. 拒否リスト用の設定ファイルを作成します。

```
# 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
…
```

この例では、**vim** エディターで編集した **blacklist.conf** ファイルの内容を示しています。**blacklist** の行では、システムの起動プロセス中に関連のカーネルモジュールが自動的に読み込まれないように指定します。ただし、**blacklist** コマンドは、拒否リストに入っていない他のカーネルモジュールの依存関係としてのモジュールの読み込みを阻止することはありません。したがって、**install** の行では、モジュールのインストールの代わりに、**/bin/false** が実行されます。

ハッシュ記号で始まる行は、ファイルがより読みやすいコメントです。

注記

カーネルモジュールの名前を入力する際には、**.ko.xz** 拡張子は名前の末尾に追加しないでください。カーネルモジュール名には拡張子はありません。ただし、対応するファイルには拡張子があります。

3. 再構築を行う前に、現在の初期 ramdisk イメージのバックアップコピーを作成します。

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

上記のコマンドは、新しいバージョンに予期しない問題が発生したときに、バックアップの **initramfs** イメージを作成します。

- または、カーネルモジュールを拒否リストに指定するカーネルバージョンに対応する、その他の初期 ramdisk イメージのバックアップコピーを作成します。

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

4. 変更を反映するために新しい初期 ramdisk イメージを生成します。

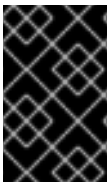
```
# dracut -f -v
```

- 現在起動しているものとは異なるカーネルバージョンの初期 ramdisk イメージを構築する場合は、ターゲット **initramfs** とカーネルバージョンの両方を指定します。

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

5. システムを再起動します。

```
$ reboot
```



重要

この手順で説明している変更は、システムを再起動しても**持続されます**。主要なカーネルモジュールを拒否リストに誤って追加した場合には、システムが不安定になったり、稼働しなくなったりする可能性があります。

関連情報

- **dracut** ユーティリティの詳細は **dracut(8)** man ページを参照してください。
- Red Hat Enterprise Linux 8 以前のバージョンで、システムの起動時にカーネルモジュールの自動読み込みを阻止する方法は、「カーネルモジュールが自動的にロードしないようにする」を参照してください。

1.9. セキュアブート用のカーネルモジュールの署名

Red Hat Enterprise Linux 7 には UEFI セキュアブート機能が含まれているので、Red Hat Enterprise Linux 7 は UEFI セキュアブートが有効になっているシステム上でインストールし、実行できます。Red Hat Enterprise Linux 7 では、UEFI システムでセキュアブートを使用する必要がないことにご留意ください。

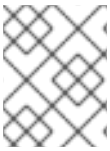
セキュアブートが有効な場合は、UEFI オペレーティングシステムのブートローダー、Red Hat Enterprise Linux カーネル、およびすべてのカーネルモジュールを秘密鍵で署名し、それに対応する公開鍵で認証する必要があります。それらが署名・認証されてなければ、システムは起動プロセスを終了できません。

Red Hat Enterprise Linux 7 ディストリビューションには、以下が含まれます。

- 署名付きブートローダー
- 署名付きカーネル
- 署名付きカーネルモジュール

また、署名された第1ステージのブートローダーと署名されたカーネルには、組み込み Red Hat 公開鍵が含まれています。これらの署名済みバイナリーおよび組み込み鍵により、Red Hat Enterprise Linux 7 は UEFI セキュアブート対応のシステムで、UEFI ファームウェアが提供する Microsoft UEFI セキュア

ブート認証局キーを使用してインストール、ブート、および実行できます。



注記

セキュアブートのサポートは、すべての UEFI ベースのシステムに含まれるわけではありません。

以下のセクションでは、セキュアブートが有効になっている UEFI ベースのビルドシステム上で Red Hat Enterprise Linux 7 に使用する、プライベートで構築されたカーネルモジュールへの自己署名に必要な手順を説明しています。また、カーネルモジュールの展開を希望するターゲットシステムに公開鍵をインポートするのに利用可能なオプションについても説明しています。

カーネルモジュールに署名して読み込むには、以下を行う必要があります。

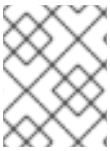
1. *システムに関連のユーティリティをインストール。*
2. *カーネルモジュールを認証。*
3. *公開鍵と秘密鍵を生成。*
4. *ターゲットシステムに公開鍵をインポート。*
5. *秘密鍵でカーネルモジュールを署名。*
6. *署名付きカーネルモジュールを読み込み。*

1.9.1. 前提条件

外部でビルドされたカーネルモジュールに署名できるようにするには、次の表にリストされているユーティリティをビルドシステムにインストールします。

表1.1 必要なユーティリティー

ユーティリティー	提供するパッケージ	使用対象	目的
openssl	**openssl**	ビルドシステム	公開および秘密 X.509 鍵のペアを生成
sign-file	**kernel-devel**	ビルドシステム	カーネルモジュールの署名に使用する Perl スクリプト
perl	**perl**	ビルドシステム	署名スクリプトの実行に使用する Perl インタープリター
mokutil	**mokutil**	ターゲットシステム	公開鍵を手動で登録する際に使用するオプションのユーティリティ
keyctl	**keyutils**	ターゲットシステム	システムキーリングに公開鍵を表示する際に使用するオプションのツール



注記

カーネルモジュールを構築、署名するビルドシステムは、UEFI セキュアブートを有効にする必要がなく、UEFI ベースのシステムである必要すらありません。

1.9.2. カーネルモジュールの認証

Red Hat Enterprise Linux 7 では、カーネルモジュールの読み込み時に、カーネルのシステムキーリング上の公開 X.509 鍵を使ってモジュールの署名をチェックします。使用される鍵は、カーネルのシステムブラックリストのキーリングにあるものを除きます。次のセクションでは、キー/キーリングのソースの概要、システム内のさまざまなソースからロードされたキーの例を示します。また、ユーザーは、カーネルモジュールの認証に必要なものを確認することができます。

1.9.2.1. カーネルモジュール認証に使用する公開鍵のソース

ブート中にカーネルは、以下の表にある永続的キーストアー式から X.509 鍵をシステムキーリングまたはシステムのブラックリストに読み込みます。

表1.2 システムキーリングのソース

X.509 鍵のソース	キー追加に関するユーザー能力	UEFI セキュアブートの状態	ブート中に読み込まれる鍵
カーネルに埋め込み	×	-	**.system_keyring**
UEFI セキュアブート "db"	限定的	有効でない	いいえ
		有効	**.system_keyring**
UEFI セキュアブート "dbx"	限定的	有効でない	いいえ
		有効	**.system_keyring**
shim.efi ブートローダーに埋め込み	いいえ	有効でない	いいえ
		有効	**.system_keyring**
Machine Owner Key (MOK) リスト	はい	有効でない	いいえ
		有効	**.system_keyring**

システムが UEFI ベースでない場合、または UEFI セキュアブートが有効になっていない場合は、カーネルに組み込まれた鍵のみがシステムのキーリングに読み込まれます。この場合、カーネルの再構築なしでキーセットを拡張することはできません。

システムのブラックリストキーリングは、無効にされた X.509 鍵の一覧です。ブラックリストにある鍵でモジュールが署名されていると、公開鍵がシステムのキーリングにあったとしても、モジュールは認証に失敗します。

*システムのキーリング上にある鍵についての情報は、**keyctl** ユーティリティーを使うと表示できます。以下は、UEFI セキュアブートが有効になっていない Red Hat Enterprise Linux 7 システムからの短い出力例です。*

```
# keyctl list %:.system_keyring
3 keys in keyring:
...asymmetric: Red Hat Enterprise Linux Driver Update Program (key 3): bf57f3e87...
...asymmetric: Red Hat Enterprise Linux kernel signing key: 4249689eefc77e95880b...
...asymmetric: Red Hat Enterprise Linux kpatch signing key: 4d38fd864ebe18c5f0b7...
```

以下は、UEFI セキュアブートが有効になっている Red Hat Enterprise Linux 7 システムからの短い出力例です。

```
# keyctl list %:.system_keyring
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...
```

*上記の出力では、UEFI セキュアブート "db" 鍵から加わった 2 つの鍵と、**shim.efi** ブートローダーに組み込まれている **Red Hat Secure Boot (CA key 1)** が示されています。UEFI セキュアブート関連のソースを使用してキーを識別するカーネルコンソールメッセージを検索することもできます。これには、UEFI セキュアブート db、組み込み shim、および MOK リストが含まれます。*

```
# dmesg | grep 'EFI: Loaded cert'
[5.160660] EFI: Loaded cert 'Microsoft Windows Production PCA 2011: a9290239...
[5.160674] EFI: Loaded cert 'Microsoft Corporation UEFI CA 2011: 13adbf4309b...
[5.165794] EFI: Loaded cert 'Red Hat Secure Boot (CA key 1): 4016841644ce3a8...
```

1.9.2.2. カーネルモジュール認証の要件

本セクションでは、UEFI セキュアブート機能が有効なシステムでカーネルモジュールを読み込むために必要な条件を説明します。

*UEFI セキュアブートが有効な場合、または **module.sig_enforce** カーネルパラメーターが指定されている場合は、システムのキーリングの鍵を使用して認証された署名済みカーネルモジュールのみを読み込むことができます。また、公開鍵は、システムのブラックリストキーリング上に存在すべきではありません。*

*UEFI セキュアブートが無効で **module.sig_enforce** カーネルパラメーターが指定されていない場合は、公開鍵なしで、未署名カーネルモジュールと署名済みカーネルモジュールを読み込むことができます。これは、以下の表で説明されています。*

表1.3 カーネルモジュールの読み込み認証要件

モジュールの署名	公開鍵ありおよび署名が有効	UEFI セキュアブートの状態	sig_enforce	モジュールの読み込み	カーネルの汚染
署名なし	-	有効でない	有効でない	成功	はい

モジュールの署名	公開鍵ありおよび署名が有効	UEFI セキュアブートの状態	sig_enforce	モジュールの読み込み	カーネルの汚染
		有効でない	有効	失敗	-
		有効	-	失敗	-
署名あり	いいえ	有効でない	有効でない	成功	はい
		有効でない	有効	失敗	-
		有効	-	失敗	-
署名あり	はい	有効でない	有効でない	成功	いいえ
		有効でない	有効	成功	いいえ
		有効	-	成功	いいえ

1.9.3. 公開および秘密 X.509 鍵のペアの生成

セキュアブートを有効化したシステム上でカーネルモジュールを使用する作業を正常に行うには、公開および秘密 X.509 鍵ペアを生成する必要があります。後で秘密鍵を使用してカーネルモジュールに署名します。セキュアブートで署名済みモジュールを検証するには、適切な公開鍵を Machine Owner Key (MOK) に追加する必要があります。手順は、「システム管理者が手動で公開鍵を MOK リストに追加する」 を参照してください。

このキーペア生成のパラメーターの一部は、設定ファイルで指定するのが最適です。

1. *キーペア生成のパラメーターで設定ファイルを作成します。*

```
# cat << EOF > configuration_file.config
[ req ]
default_bits = 4096
distinguished_name = req_distinguished_name
prompt = no
string_mask = utf8only
x509_extensions = myexts

[ req_distinguished_name ]
O = Organization
CN = Organization signing key
emailAddress = E-mail address

[ myexts ]
basicConstraints=critical,CA:FALSE
keyUsage=digitalSignature
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid
EOF
```

2. X.509 公開鍵と秘密鍵のペアを以下の例のように作成します。

```
# openssl req -x509 -new -nodes -utf8 -sha256 -days 36500 \
-batch -config configuration_file.config -outform DER \
-out my_signing_key_pub.der \
-keyout my_signing_key.priv
```

公開鍵は **my_signing_key_pub.*der*** ファイルに書き込まれます。この秘密鍵は **my_signing_key.*priv*** ファイルに書き込まれます。

3. カーネルモジュールを認証、読み込むすべてのシステムに公開鍵を登録します。詳細は 「公開鍵のターゲットシステムでの登録」 を参照してください。

警告

強力なセキュリティー対策とアクセスポリシーを適用して、秘密鍵の内容を保護します。悪用すれば、この鍵は、一致する公開鍵で認証されるシステムのセキュリティに危害を与えるために使用できます。

1.9.4. 公開鍵のターゲットシステムでの登録

セキュアブートが有効になっている UFEI ベースのマシンで Red Hat Enterprise Linux 7 を起動すると、無効化したキーの dbx データベースにあるものを除いて、セキュアブート db キーデータベースにあるすべてのパブリックキーのシステムキーにカーネルがロードされます。以下のセクションでは、システムキーリングが公開鍵を使用してカーネルモジュールを認証できるように、ターゲットシステムで公開鍵をインポートする方法を説明します。

1.9.4.1. 公開鍵を含むファクトリーファームウェアイメージ

システムでカーネルモジュールの認証を実現するために、ファクトリーファームウェアイメージで公開鍵を UEFI セキュアブート鍵データベースに組み入れるようシステムベンダーに要求することを検討します。

1.9.4.2. システム管理者が手動で公開鍵を MOK リストに追加する

Machine Owner Key (MOK) 機能を使用して、UEFI セキュアブートキーデータベースを拡張することができます。セキュアブートが有効になっている UEFI ベースのシステムで Red Hat Enterprise Linux 7 が起動すると、鍵データベースの鍵に加えて、MOK リストの鍵もシステムキーリングに追加されます。MOK リストの鍵は、セキュアブートデータベースの鍵と同様に永続的かつ安全な方法で保存されますが、これらは別個の機能です。MOK 機能は、**shim.efi**、**MokManager.efi**、**grubx64.efi** および Red Hat Enterprise Linux 7 **mokutil** ユーティリティーでサポートされています。

MOK 鍵の登録は、各ターゲットシステムの UEFI システムコンソールでユーザーが物理的に手動で対応する必要があります。それにもかかわらず、MOK 機能は、新規生成された鍵ペアのテストとこれで署名されたカーネルモジュールのテストにおいて便利な方法を提供します。

公開鍵を MOK リストに追加するには、以下に従います。

1. 公開鍵を MOK リストに追加するようリクエストします。

```
# mokutil --import my_signing_key_pub.der
```

このMOK 登録リクエストに関するパスワードの入力と確認が求められます。

2. マシンを再起動します。
このMOK 鍵登録リクエストは **shim.efi** が発見し、**MokManager.efi** を起動してUEFI コンソールからの登録が完了できるようになります。

3. このリクエストに関連付けたパスワードを入力し、登録を確認します。
公開鍵がMOK リストに永続的に追加されます。

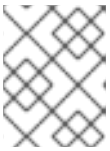
鍵がMOK リストに追加されると、UEFI セキュアブートが有効になっているシステムの起動時に毎回、この鍵はシステムのキーリングに自動的に追加されます。

1.9.5. 秘密鍵を使用したカーネルモジュールの署名

カーネルモジュールの準備が完了していることを前提とします。

- Perl スクリプトを使用して、秘密鍵でカーネルモジュールに署名します。

```
# perl /usr/src/kernels/$(uname -r)/scripts/sign-file \
sha256 \
my_signing_key.priv\
my_signing_key_pub.der\
my_module.ko
```



注記

Perl スクリプトを使用するには、署名するカーネルモジュールファイルの他に、秘密および公開鍵の両方を含むファイルを提供する必要があります。

カーネルモジュールはELF イメージ形式で、Perl スクリプトは署名を計算して、カーネルモジュールファイルのELF イメージに直接追加します。**modinfo** ユーティリティーを使うと、カーネルモジュールの署名がある場合は、それについての情報を表示できます。**modinfo** の使用方法は、「モジュール情報の表示」 を参照してください。

この追加された署名はELF イメージセクションには含まれず、またELF イメージの正式な一部ではありません。このため、**readelf** のようなユーティリティは、この署名をカーネルモジュールに表示することができません。

これでカーネルモジュールの読み込み準備が完了しました。署名済みのカーネルモジュールは、UEFI セキュアブートが無効となっているシステムまたはUEFI 以外のシステムでも読み込み可能であることに注意してください。つまり、署名済みのカーネルモジュールと署名なしのカーネルモジュールの両方を提供する必要はないことになります。

1.9.6. 署名済みカーネルモジュールの読み込み

公開鍵が登録されてシステムキーリングに格納されたら、**mokutil** を使って公開鍵をMOK リストに追加します。次に、**modprobe** コマンドを使用して、カーネルモジュールを手動で読み込みます。

1. オプションで、公開鍵の登録前にカーネルモジュールを読み込まないことを確認します。
現在読み込み済みのカーネルモジュールを一覧表示する方法は、「読み込み済みモジュールの一覧表示」 を参照してください。

2. *現在のブートで、システムキーリングに追加されている鍵を確認します。*

```
# keyctl list %:.system_keyring
```

公開鍵はまだ登録されていないので、このコマンドの出力には表示されません。

3. *公開鍵の登録をリクエストします。*

```
# mokutil --import my_signing_key_pub.der
```

4. *再起動して、UEFI コンソールでの登録を完了します。*

```
# reboot
```

5. *システムキーリングの鍵を再度確認します。*

```
# keyctl list %:.system_keyring
```

6. *任意のカーネルの /**extra**/ ディレクトリーにモジュールをコピーします。*

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

7. *モジュールの依存関係の一覧を更新します。*

```
# depmod -a
```

8. *カーネルモジュールを読み込み、正常にロードされたことを確認します。*

```
# modprobe -v my_module
# lsmod | grep my_module
```

   a. *必要に応じて、起動時にモジュールを読み込むには、**/etc/modules-loaded.d/my_module.conf** ファイルに追加します。*

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

第2章 SYSCTL およびカーネルの設定項目の使用

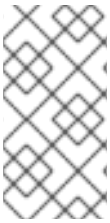
2.1. カーネルの設定項目の概要

カーネルの設定項目を使用することで、Red Hat Enterprise Linux の起動時に、またはシステムを実行中にオンデマンドで動作をカスタマイズできます。ハードウェアのパラメーターによっては、起動時のみに指定され、システムが実行し始めると変更できないものもありますが、ほとんどの場合は必要に応じて変更でき、次の起動に向けてパーマネント設定ができます。

2.2. カーネルの設定項目の使用方法

カーネルの設定項目を修正するには、以下の 3 つの方法があります。

1. **sysctl** コマンドを使用する方法
2. **/etc/sysctl.d/** ディレクトリーで設定ファイルを手動で修正する方法
3. シェルを経由して、**/proc/sys** にマウントされた仮想ファイルと相互作用する方法



注記

すべての起動時パラメーターが sysfs サブシステムの制御下にあるわけではありません。いくつかのハードウェア固有のオプションは、カーネルのコマンドライン上に設定する必要がありますが、これらのオプションについては、本ガイドの Kernel Parameters セクションを参照してください。

2.2.1. sysctl コマンドの使用

sysctl コマンドを使用して、カーネルの設定項目を表示、読み取り、および設定します。表示または読み取りの際のカーネルの設定項目のフィルター、そしてカーネルの設定項目の一時的または永続的な設定が可能となります。

1. 変数の表示

```
# sysctl -a
```

2. 変数の読み取り

```
# sysctl kernel.version
kernel.version = #1 SMP Fri Jan 19 13:19:54 UTC 2018
```

3. 変数の一時的な書き込み

```
# sysctl <tunable class>.<tunable>=<value>
```

4. 変数の永続的な書き込み

```
# sysctl -w <tunable class>.<tunable>=<value> >> /etc/sysctl.conf
```

2.2.2. /etc/sysctl.d 内のファイルの変更

*起動時にデフォルトをオーバーライドするには、手動で **/etc/sysctl.d** にファイルを追加することも可能です。*

1. ***/etc/sysctl.d** に新しいファイルを作成します。*

```
# vim /etc/sysctl.d/99-custom.conf
```

2. *以下の形式で、設定したい変数を 1 行に 1 つずつ入れていきます。*

```
<tunable class>.<tunable> = <value> +
<tunable class>.<tunable> = <value>
```

3. *ファイルを保存します。*

4. *変更を有効にするためにマシンを再起動するか、*
   または
   ***sysctl -p /etc/sysctl.d/99-custom.conf** を実行して再起動せずに変更を適用します。*

2.3. 制御可能な設定項目について

設定項目は、カーネルのサブシステムによっていくつかのグループに分けられます。Red Hat Enterprise Linux システムにおける設定項目には、以下のクラスがあります。

表2.1 sysctl インターフェースの表

クラス	サブシステム
abi	実行ドメインおよびパーソナリティー
crypto	暗号化インターフェース
debug	カーネルのデバッグインターフェース
dev	デバイスの詳細
fs	グローバルおよび固有のファイルシステムの設定項目
kernel	グローバルなカーネルの設定項目
net	ネットワークの設定項目
sunrpc	Sun Remote Procedure Call (NFS)
user	ユーザー名前空間の制限
vm	メモリー、バッファー、およびキャッシュのチューニングと管理

2.3.1. ネットワークインターフェースのパラメーター

システム管理者は、ネットワークの設定項目を経由して実行中のシステムのネットワーク設定を変更できます。

ネットワークの設定項目は、**/proc/sys/net** ディレクトリーにあります。このディレクトリーには、さまざまなネットワークトピックに関する複数のサブディレクトリーが格納されています。システム管理者はこれらのサブディレクトリー内のファイルを変更し、ネットワーク設定を調整する必要があります。

以下は、最も頻繁に使用されるディレクトリーです。

1. **/proc/sys/net/core/**

2. **/proc/sys/net/ipv4/**

/proc/sys/net/core/ ディレクトリーには、カーネルとネットワーク層との間の相互作用を制御するさまざまな設定が格納されています。これらの設定項目のをいくつか調整することで、たとえば、受信キューのサイズを増加したり、ネットワークインターフェース専用の接続およびメモリーの最大値を上げたりすることで、システムのパフォーマンスを向上させることができます。システムのパフォーマンスは、個々の問題のさまざまな側面に左右されることに留意してください。

/proc/sys/net/ipv4/ ディレクトリーには、追加のネットワーク設定が格納されています。これは、システムに対する攻撃を防止したり、システムをルーターとして機能させるように使用している場合に役立ちます。ディレクトリーには、IP および TCP の両方の変数が格納されています。これらの変数に関する詳細は **/usr/share/doc/kernel-doc-<version>/Documentation/networking/ip-sysctl.txt** を参照してください。

/proc/sys/net/ipv4/ ディレクトリー内のその他のディレクトリーは、ネットワークスタックのさまざまな側面を扱っています。

1. **/proc/sys/net/ipv4/conf/** - 各システムインターフェースを異なる方法で設定できるようにします。 これには、未設定のデバイス用のデフォルト設定の使用や、すべての特殊設定をオーバーライドする設定が含まれます。

2. **/proc/sys/net/ipv4/neigh/** - システムに直接接続されたホストとの通信のための設定や、1 ステップ以上離れたシステム用の異なる設定も含まれています。

3. **/proc/sys/net/ipv4/route/** - システム上の任意のインターフェースとのルーティングに適用される指定値が格納されています。

ネットワークの設定項目のリストは IPv4 インターフェースと関係があり、**/proc/sys/net/ipv4/{all, <interface_name>}/** ディレクトリーからアクセスできます。

以下のパラメーターの説明は、カーネルドキュメントのサイトを参考にしています。[1]

log_martians

カーネルログに無効なアドレスを持つパケットをログします。

Type	デフォルト
ブール値	0

1 つ以上の **conf/{all,interface}/log_martians** が TRUE に設定されている場合に有効です。

Further Resources

- 「カーネルパラメーター ipv4.conf.all.log_martians はどんなことに使用されますか?」

- 「メッセージファイルに "martian source" と記録される理由」

accept_redirects

ICMP のリダイレクトメッセージを受信します。

Type	デフォルト
ブール値	1

インターフェース用の **accept_redirects** が、以下の条件下で有効となります。

- **conf/{all,interface}/accept_redirects** が両方とも TRUE の場合 (インターフェースへの転送が有効な場合)
- **conf/{all,interface}/accept_redirects** のうち少なくとも 1 つが TRUE の場合 (インターフェースへの転送は無効)

「ICMP リダイレクトを無効にする方法」を参照してください。

転送

インターフェース上で IP 転送を有効にします。

Type	デフォルト
ブール値	0

Further Resources

- 「パケット転送および非ローカルバインディングの有効化」

mc_forwarding

マルチキャストルーティングを実施します。

Type	デフォルト
ブール値	0

- 読み取り専用値
- マルチキャストルーティングのデーモンが必要です。
- インターフェース用にマルチキャストルーティングを有効にするためには **conf/all/mc_forwarding** も TRUE に設定する必要があります。

Further Resources

- 読み取り専用の動作に関する詳細は、「Why system reports "permission denied on key" while setting the kernel parameter "net.ipv4.conf.all.mc_forwarding"?」を参照してください。

medium_id

接続しているメディアがデバイスを区別するために使用する任意の値です。

Type	デフォルト
整数	0

備考

- *同じメディア上の 2 つのデバイスにおいて、このうちの 1 つでしかブロードキャストパケットが 受信されない場合、2 つのデバイスの id 値が異なる可能性があります。*
- *デフォルト値の 0 は、メディアに対してそのデバイスが唯一のインターフェースであることを意味します。*
- *値が -1 の場合、メディアが不明であることを示します。*
- *現在は、proxy_arp の動作を変更するために使用されています。*
- *proxy_arp の機能は、異なるメディアに接続された 2 つのデバイスの間で転送されたパケット用に有効となります。*

Further Resources *- たとえば、「Using the "medium_id" feature in Linux 2.2 and 2.4」 を参照してください。*

proxy_arp

Proxy ARP の実行

Type	デフォルト
ブール値	0

conf/{all,interface}/proxy_arp *のうち少なくとも 1 つが TRUE に設定されている場合、インターフェース用の* ***proxy_arp*** *は有効です。そうでない場合は無効です。*

proxy_arp_pvlan

プライベート VLAN Proxy ARP

Type	デフォルト
ブール値	0

RFC 3069 などの機能をサポートするために、Proxy ARP が同じインターフェースに応答できるようにします。

shared_media

共有メディアのリダイレクト RFC1620 を送信 (ルーター) または受信 (ホスト) します。

Type	デフォルト
ブール値	1

注記

- *secure_redirects をオーバーライドします。*
- ***conf/{all,interface}/shared_media** のうち少なくとも 1 つが TRUE に設定されている場合、インターフェース用の **shared_media** が有効となります。*

secure_redirects

インターフェースの現在のゲートウェイリストにリストされているゲートウェイに対する ICMP リダイレクトメッセージのみを受信します。

Type	デフォルト
ブール値	1

備考

- *無効となった場合でも、RFC1122 リダイレクトルールが引き続き適用されます。*
- *shared_media によってオーバーライドされます。*
- ***conf/{all,interface}/secure_redirects** のうち少なくとも 1 つが TRUE に設定されている場合、インターフェース用の secure_redirects は有効となります。*

send_redirects

ルーターの場合、リダイレクトを送信します。

Type	デフォルト
ブール値	1

注記
*インターフェースの**send_redirects** は、**conf/{all,interface}/send_redirects** のうち少なくとも 1 つが TRUE に設定されている場合に有効にされます。*

bootp_relay

ローカル用としてこのホストに予定されていないソースアドレス 0.b.c.d のパケットを受信します。

Type	デフォルト
ブール値	0

注記

- これらのパケットを管理するために BOOTP デーモンを有効にする必要があります。
- インターフェース用に BOOTP リレーを有効にするために **conf/all/bootp_relay** も TRUE に設定する必要があります。
- 実装されていない場合は、『Red Hat Enterprise Linux ネットワークガイド』の「DHCP リレーエージェント」を参照してください。

accept_source_route

SRR オプションのあるパケットを受信します。

Type	デフォルト
ブール値	1

注記

- インターフェース上の SRR オプションのあるパケットを受信するために、**conf/all/accept_source_route** も TRUE に設定する必要があります。

accept_local

ローカルソースアドレスのあるパケットを受信します。

Type	デフォルト
ブール値	0

注記

- これは、適切なルーティングとのコンビネーションにて、2 つのローカルインターフェース間のパケットをワイヤー上で移動し、適切に受信させるために使用することができます。
- accept_local に効果をもたらすためには、**rp_filter** をゼロ以外の値に設定する必要があります。

route_localnet

ルーティング中は、ループバックアドレスを martian ソースまたは宛先として考慮しません。

Type	デフォルト
ブール値	0

注記

- これにより、ローカルルーティング目的での 127/8 の使用が有効となります。

rp_filter

ソースの検証を有効化

Type	デフォルト
整数	0

値	効果
0	ソースの検証はありません。
1	RFC3704 厳密な逆方向パスで定義された厳密モード
2	RFC3704 緩やかな逆方向パスで定義された緩やかなモード

注記

- *RFC3704 における現在の推奨プラクティスは、DDos 攻撃による IP スプーフィングを回避するために厳密モードを有効にすることです。*
- *非対称のルーティングまたは別の複雑なルーティングを使用する場合は、緩やかなモードが推奨されます。*
- *{interface} でソースの検証を行う際、**conf/{all,interface}/rp_filter** の中の最大値が使用されます。*

arp_filter

Type	デフォルト
ブール値	0

値	効果
0	(デフォルト) カーネルは、別のインターフェースからのアドレスの ARP 要求に対応できます。正常な通信の可能性を向上させるので、通常は理にかなっています。
1	同じサブネットで複数のネットワークインターフェースを持つことを可能にします。また、カーネルがインターフェースから ARP 要求 の IP パケットをルーティングするかどうかに基づいて、各インターフェースの ARP が応答できるようにします (したがって、これを機能させるためにはソースベースのルーティングを使用する必要があります)。つまり、ARP 要求に応答するカード (通常は 1) の制御が可能となります。

注記

- *IP アドレスは、特定のインターフェースではなく、Linux の完全なホストが所有します。この動作が問題を起こすのは、負荷分散などのより複雑なセットアップの場合だけです。*
- ***conf/{all,interface}/arp_filter** のうち少なくとも 1 つが TRUE に設定されている場合、インターフェース用の **arp_filter** は有効となります。*

arp_announce

インターフェース上に送信された ARP 要求の IP パケットからローカルソースの IP アドレスを発表するための異なる制限レベルを定義します。

Type	デフォルト
整数	0

値	効果
0	(デフォルト) 任意のインターフェース上に設定された任意のローカルアドレスを使用します。
1	このインターフェースでは、出力先のサブネットにないローカルアドレスは使用しないようにします。このインターフェースを経由してアクセス可能な出力先ホストが、ARP 要求のソース IP アドレスが受信側インターフェース上に設定されるロジカルなネットワークの一環となるよう要求した場合、このモードは役立ちます。要求を生成する際、出力先 IP を含むすべてのサブネットを確認し、そのようなサブネットからのソースアドレスである場合は保持します。そのようなサブネットがない場合は、レベル 2 のルールにしたがってソースアドレスを選択します。
2	この出力先には常に最適のローカルアドレスを使用します。このモードでは、IP パケットのソースアドレスを無視し、出力先ホストとの対話には好みのローカルアドレスを選択するようにします。このようなローカルアドレスは、出力先 IP アドレスを含む発信インターフェース上のすべてのサブネット上にある主要な IP アドレスを探すことで選択されます。適切なローカルアドレスが見つからない場合は、発信インターフェース上またはその他すべてのインターフェース上にある最初のローカルアドレスを選択します。この時、アナウンスするソース IP アドレスに関係なく、要求に対する応答があることを期待します。

備考

- ***conf/{all,interface}/arp_announce** の中の最大値が使用されます。*
- *制限レベルを上げると、解決済み出力先から応答がある可能性が高くなり、制限レベルを下げると、より有効な送信者情報をアナウンスします。*

arp_ignore

受信した ARP 要求に対して応答するさまざまなモードを定義します。 この ARP 要求は、ローカル出力先 IP アドレスを解決するものです。

Type	デフォルト
整数	0

値	効果

値	効果
0	(デフォルト): 任意のインターフェース上に設定された任意のローカル出力先 IP アドレスに応答します。
1	出力先 IP アドレスが受信インターフェース上で設定されたローカルアドレスの場合にのみ応答します。
2	出力先 IP アドレスが受信インターフェース上で設定されたローカルアドレスで、送信者の IP アドレスと出力先 IP アドレスの両方がこのインターフェース上の同じサブネットの一部である場合にのみ応答します。
3	スコープホストで設定されたローカルアドレスには応答しません。グローバルおよびリンク用のアドレス解決のみに応答します。
4-7	予備
8	ローカルアドレスの場合はすべて、応答しません。 {interface} 上で ARP 要求が受信された際に conf/{all,interface}/arp_ignore の最大値が使用されます。

備考

arp_notify

アドレスおよびデバイスの変更を通知するモードを定義します。

Type	デフォルト
ブール値	0

値	Effect
0	何もしません。
1	デバイスの停止またはハードウェアのアドレス変更の際、余計な ARP 要求を生成します。

注記

arp_accept

IP がまだ ARP テーブルに存在しない余計な ARP フレームの動作を定義します。

Type	デフォルト
ブール値	0

値	Effect
0	ARP テーブルに新しいエントリーを作成しません。
1	ARP テーブルに新しいエントリーを作成します。

注記
この設定がオンの場合、応答および要求の両タイプの余計なARP が、ARP テーブルをアップデートするようトリガーします。ARP テーブルが余計なARP フレームのIP アドレスをすでに格納している場合、この設定がオンまたはオフであることに関係なく、ARP テーブルがアップデートされます。

app_solicit

マルチキャストプローブへドロップバックする前にネットリンクを経由してユーザー空間のARP デーモンに送信するプローブの最大数 (mcast_solicit を参照してください)。

Type	デフォルト
整数	0

注記
***mcast_solicit** を参照してください。*

disable_policy

このインターフェースのIPSEC ポリシー (SPD) を無効にします。

Type	デフォルト
ブール値	0

needinfo

disable_xfrm

いかなるポリシーであろうと、このインターフェースのIPSEC 暗号化を無効にします。

Type	デフォルト
ブール値	0

needinfo

igmpv2_unsolicited_report_interval

次の未承諾のIGMPv1 またはIGMPv2 レポートの再送信が実行されるミリ秒単位の間隔。

Type	デフォルト
整数	10000

注記
ミリ秒

igmpv3_unsolicited_report_interval

次の未承諾の IGMPv3 レポートの再送信が実行されるミリ秒単位の間隔。

Type	デフォルト
整数	1000

注記
ミリ秒

tag

必要に応じて使用可能な数字の書き込みが可能です。

Type	デフォルト
整数	0

xfrm4_gc_thresh

IPv4 宛先キャッシュエントリー用のガベージコレクションを開始するしきい値。

Type	デフォルト
整数	1

注記
この値が 2 倍になると、システムは新しい割り当てを拒否します。

2.3.2. グローバルなカーネルの設定項目

システム管理者は、グローバルカーネルパラメーターを経由して実行中のシステムの一般的な設定を構成および変更できます。

*グローバルカーネルパラメーターは、**/proc/sys/kernel/** ディレクトリーにあり、さまざまな設定内容向けに、名前付きの制御ファイル、またはサブディレクトリーでグループ分けされています。グローバルカーネルパラメーターを調整するには、システム管理者が制御ファイルを修正する必要があります。*

以下のパラメーターの説明は、カーネルドキュメントのサイトを参考にしています。[2]

dmesg_restrict

*非特権ユーザーが、**dmesg** コマンドを実行して、カーネルのログバッファーのメッセージを表示しないようにするかどうかを示します。*
詳細は「Kernel sysctl documentation」を参照してください。

core_pattern

コアダンプファイルのパターン名を指定します。

最大長	デフォルト
128 文字	"core"

詳細は「Kernel sysctl documentation」を参照してください。

hardlockup_panic

ハードロックアップが検出された場合にカーネルパニックを制御します。

Type	値	Effect
整数	0	カーネルが、ハードロックアップでパニックを発生させません。
整数	1	ハードロックアップでのカーネルパニック

パニックを発生させるためには、システムで、最初にハードロックアップを検出する必要があります。検出は、nmi_watchdog パラメーターで制御されます。

Further Resources

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

softlockup_panic

ソフトロックアップが検出されるとカーネルパニックを制御します。

Type	値	Effect
整数	0	カーネルが、ソフトロックアップでパニックにならない
整数	1	カーネルが、ソフトロックアップでパニックになる

RHEL 7 では、この値のデフォルトは 0 です。

***softlockup_panic** の詳細は「kernel_parameters」を参照してください。*

kptr_restrict

*制限は、/**proc** またはその他のインターフェースを介して、カーネルアドレスを露出させる制限が配置されているかどうかを示します。*

Type	デフォルト
整数	0

値	Effect
0	出力前にカーネルアドレスのハッシュ値を計算します。
1	出力されたカーネルポインターを、特定の条件下で、0 に置き換えます。
2	出力されたカーネルポインターを、無条件に 0 に置き換えます。

詳細は「Kernel sysctl documentation」を参照してください。

nmi_watchdog

x86 システムで、ハードロックアップ検出を制御します。

Type	デフォルト
整数	0

値	Effect
0	ロックアップ検出を無効にします。
1	ロックアップ検出を有効にします。

ハードロックアップ検出は、各 CPU で割り込みに応答する機能を監視します。

詳細は「Kernel sysctl documentation」を参照してください。

watchdog_thresh

*ウォッチドッグの **hrtimer**、NMI イベント、およびソフトロックアップまたはハードロックアップのしきい値を制御します。*

デフォルトのしきい値	ソフトロックアップのしきい値
10 秒	2 * **watchdog_thresh**

このパラメーターを 0 (ゼロ) に設定すると、ロックアップ検出を無効にします。

詳細は「Kernel sysctl documentation」を参照してください。

panic, panic_on_oops, panic_on_stackoverflow, panic_on_unrecovered_nmi, panic_on_warn, panic_on_rcu_stall, hung_task_panic

このパラメーターは、カーネルがパニックを発生する状況を指定します。
*一連の **panic** パラメーターの詳細は、「Kernel sysctl documentation」を参照してください。*

printk, printk_delay, printk_ratelimit, printk_ratelimit_burst, printk_devkmsg

このようなパラメーターは、カーネルエラーメッセージのログへの記録または出力を制御します。
*一連の **printk** パラメーターの詳細は、「Kernel sysctl documentation」を参照してください。*

shmall, shmmax, shm_rmid_forced

このパラメーターは、共有メモリーの制限を制御します。
*一連の **shm** パラメーターの詳細は、「Kernel sysctl documentation」を参照してください。*

threads-max

***fork()** システムコールが作成するスレッドの最大数を制御します。*

最小値	最大値
20	FUTEX_TID_MASK (0x3fffffff) で指定

***threads-max** 値は、利用可能な RAM ページに対して確認されます。スレッド構造が、利用可能な RAM ページを使用しすぎている場合は、それに応じて **threads-max** が削減します。*

詳細は「Kernel sysctl documentation」を参照してください。

pid_max

PID 割り当てラップ値。
詳細は「Kernel sysctl documentation」を参照してください。

numa_balancing

このパラメーターは、NUMA メモリーの自動分散を有効または無効にします。NUMA マシンでは、CPU がリモートメモリーにアクセスするかどうかについてパフォーマンスのペナルティーがあります。
詳細は「Kernel sysctl documentation」を参照してください。

numa_balancing_scan_period_min_ms, numa_balancing_scan_delay_ms, numa_balancing_scan_period_max_ms, numa_balancing_scan_size_mb

このパラメーターは、そのタスクが実行しているローカルのメモリーノードへデータが移行する必要がある場合に、ページが適切に配置されているかどうかを検出します。
*一連の **numa_balancing_scan** パラメーターの詳細は、「Kernel sysctl documentation」を参照してください。*

[1] https://www.kernel.org/doc/Documentation/

[2] https://www.kernel.org/doc/Documentation/

第3章 カーネルパラメーターおよび値の表示

3.1. カーネルコマンドラインパラメーター

カーネル引数の別名を持つカーネルコマンドラインパラメーターは、Red Hat Enterprise Linux の動作をカスタマイズするために起動時のみに使用されます。

3.1.1. カーネルコマンドラインパラメーターの設定

本セクションは、**GRUB2** ブートローダーを使用して、AMD64 システムおよび Intel 64 システム、および **zipl** を使用する IBM Z で、カーネルコマンドラインパラメーターを変更する方法を説明します。

カーネルコマンドラインパラメーターは、**boot/grub/grub.cfg** 設定ファイルに保存されています。この設定ファイルは、**GRUB2** ブートローダーにより生成されます。この設定ファイルは編集しません。このファイルへの変更ができるのは、設定スクリプトのみです。

AMD64 および Intel 64 システムならびに IBM Power Systems ハードウェア向けに、GRUB2 のカーネルコマンドラインパラメーターを変更します。

1. **vim** または **Gedit** などのプレーンテキストエディターを使って **/etc/default/grub** 設定ファイルを **root** として開きます。
2. このファイル内で、以下のように **GRUB_CMDLINE_LINUX** で始まるラインを探します。

```
GRUB_CMDLINE_LINUX="rd.lvm.lv=rhel/swap crashkernel=auto rd.lvm.lv=rhel/root rhgb quiet"
```

3. 必要なカーネルコマンドラインパラメーターの値を変更します。続いてファイルを保存し、エディターを終了します。
4. 編集された **default** ファイルを使用して、**GRUB2** 設定を再生成します。BIOS ファームウェアを使用しているシステムの場合は次のコマンドを実行します。

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

UEFI ファームウェアを使用しているシステムの場合は次のコマンドを実行します。

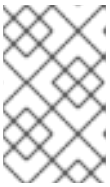
```
# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

上記の手順を完了するとブートローダーが再設定され、設定ファイルに指定したカーネルコマンドラインパラメーターが、次回の再起動後に適用されます。

IBM Z ハードウェア用に zipl のカーネルコマンドラインパラメーターの変更

1. **vim** または **Gedit** などのプレーンテキストエディターを使って **/etc/zipl.conf** 設定ファイルを **root** として開きます。
2. このファイル内で **parameters=** セクションを探し、必要なパラメーターを編集するか、見つからない場合はパラメーターを追加します。続いてファイルを保存し、エディターを終了します。
3. **zipl** 設定を再生成します。

```
# zipl
```



注記

*オプションを何も付けずに **zipl** コマンドのみを実行すると、デフォルト値が使用されます。使用できるオプションの一覧については **zipl(8)** の man ページを参照してください。*

上記の手順を完了するとブートローダーが再設定され、設定ファイルに指定したカーネルコマンドラインパラメーターが、次回の再起動後に適用されます。

3.1.2. 制御可能なカーネルコマンドラインパラメーター

カーネルコマンドラインパラメーターの全一覧は、https://www.kernel.org/doc/Documentation/admin-guide/kernel-parameters.txt を参照してください。

3.1.2.1. ハードウェア固有のカーネルコマンドラインパラメーター

pci=option[,option...]

PCI ハードウェアサブシステムの動作を指定

設定	Effect
earlydump	[X86] カーネルが何らかの変更をする前に PCI 設定領域をダンプします。
off	[X86] PCI バスを調べません。
noaer	[PCIE] PCIEAER カーネルパラメーターが有効な場合、このカーネル起動オプションを使用して PCIE Advanced Error Reporting (AER) の使用を無効にすることができます。
noacpi	[X86] Interrupt Request (IRQ) ルーティングまたは PCI スキャニング用に Advanced Configuration and Power Interface (ACPI) は使用しません。
bfsort	PCI デバイスを幅優先順に並べ替えます。この並べ替えは、古い (⇐ 2.4) カーネルに対応するデバイスの順番となります。
nobfsort	PCI デバイスを幅優先順に並べ替えません。

*さらなる PCI オプションは、**kernel-doc-<version>.noarch** パッケージのディスクドキュメントに文書化されています。ここでは、'<version>' は対応するカーネルバージョンに置き換える必要があります。*

acpi=option

Advanced Configuration and Power Interface (ACPI) の動作を指定します。

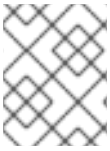
設定	Effect
acpi=off	ACPI の無効化
acpi=ht	ACPI の起動テーブル構文解析を使用しますが、ACPI インタープリターは有効にしません これにより、ハイパースレッディングで **必要ではない** 任意の ACPI 機能が無効となります。
acpi=force	ACPI サブシステムの有効化が必要
acpi=strict	ACPI 仕様と完全に準拠していないプラットフォームに対する ACPI 層の耐性を低くします。
acpi_sci=<value>	ACPI SCI の割り込みをセットアップします。 ここで <value> は、エッジ、レベル、ハイ、ローのうちの 1 つとなります。
acpi=noirq	IRQ ルーティングに ACPI は使用しません。
acpi=nocmcff	訂正されたエラーの firmware first (FF) モードを無効にします。これにより、ファームウェアが FF フラグを設定したかどうかを確認する HEST CMC エラーソースの構文解析が無効となります。この場合、訂正されたエラー報告が重複する可能性があります。

第4章 カーネル機能

本章では、多くのユーザー空間ツールを有効にするカーネル機能の目的および使用について説明します。また、これらのツールの詳細についてのリソースも紹介します。

4.1. CONTROL GROUPS

4.1.1. コントロールグループの概要



注記

コントロールグループの名前空間は、Red Hat Enterprise Linux 7.5 ではテクノロジープレビューとして提供されています。

Linux コントロールグループ (cgroup) は、システムハードウェアの使用上の制限を有効にし、**cgroup** 内で実行する個々のプロセスが **cgroup** の設定で許可された分だけ活用していることを確認します。

コントロールグループは、***名前空間*** が有効にしたリソースの使用量を制限します。たとえば、ネットワーク名前空間により、あるプロセスの特定のネットワークカードへのアクセスが可能となり、cgroup はこのプロセスのカードの使用量が 50% を超えないように確認し、他のプロセスが帯域幅を確実に利用できるようにします。

コントロールグループの名前空間は、/proc/self/ns/cgroup インターフェースを経由して個々の cgroup の仮想化ビューを提供します。

目的は、グローバルな名前空間から cgroup への特権付きデータの漏えいを回避し、コンテナーマイグレーションなどの他の機能を有効にすることです。

現在、コンテナーを単一の cgroup と関連付けることが格段に容易となっていることから、コンテナーにはより一貫した cgroup のビューがあります。また、コンテナー内部のタスクを有効にし、属している cgroup の仮想化ビューが可能となります。

4.1.2. 名前空間の概要

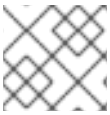
名前空間は、分離したシステムリソースの仮想化ビューを可能にするカーネル機能です。システムリソースからプロセスを分離することで、プロセスが対話できるものを指定および制御できます。名前空間は、コントロールグループにとって不可欠なものです。

4.1.3. サポートしている名前空間

以下の名前空間は、Red Hat Enterprise Linux 7.5 以降でサポートされています。

- Mount
  - マウント名前空間は、ファイルシステムのマウントポイントを分離します。これにより各プロセスは明確なファイルシステム領域を持つことができ、その領域内で操作します。
- UTS
  - ホスト名および NIS ドメイン名
- IPC
  - System V IPC、POSIX メッセージキュー

- PID
  - プロセス ID
- Network
  - ネットワークデバイス、スタック、ポートなど。
- User
  - ユーザーおよびグループ ID
- コントロールグループ
  - cgroup の分離



注記

コントロールグループの使用方法は、『リソース管理ガイド』 で説明しています。

4.2. カーネルソースチェッカー

Linux カーネルモジュールのソースチェッカー (ksc) は、所定のカーネルモジュール内におけるホワイトリスト以外の記号を確認するツールです。Red Hat パートナーもこのツールを使用して、Red Hat Bugzilla データベースにバグを報告することで、ホワイトリストに含まれる記号のレビューを要求できます。

4.2.1. 用途

本ツールは "-k" オプションでモジュールへのパスを受け取ります。

```
# ksc -k e1000e.ko
Checking against architecture x86_64
Total symbol usage: 165 Total Non white list symbol usage: 74

# ksc -k /path/to/module
```

出力は、**$HOME/ksc-result.txt** に保存されます。ホワイトリストへ追加された記号のレビューが要求されると、ホワイトリストに記載されていない各記号の使用方法の説明が **ksc-result.txt** ファイルに追加される必要があります。"-p" オプションで **ksc** を実行することで、リクエストバグを報告することができます。

注記

KSC は現在、**xz** 圧縮をサポートしていません。 **ksc** ツールは、**xz** 圧縮方法を処理できず、以下のエラーメッセージを表示します。

```
Invalid architecture, (Only kernel object files are supported)
```

この制限がなくなるまで、システム管理者は **ksc** ツールを実行する前に、**xz** 圧縮を使用して任意のサードパーティーモジュールを手動で展開する必要があります。

4.3. ファイルの DIRECT ACCESS (DAX)

「file system dax」または「fs dax」として知られるファイルの Direct Access により、アプリケーションは、デバイスへのアクセスをバッファーするためにページキャッシュを使用することなく、dax 対応のストレージデバイスへのデータの読み取りおよび書き込みが可能となります。

この機能は、ext4 ファイルシステムまたは xfs ファイルシステムを使用する場合に限り利用でき、**-o dax** でファイルシステムをマウントするか、**/etc/fstab** のマウントエントリーの options セクションに **dax** を追加すると有効になります。

コードの実例を含む詳しい情報は、**kernel-doc** パッケージを参照してください。これは、**/usr/share/doc/kernel-doc-<version>/Documentation/filesystems/dax.txt** に保存されていて、'<version>' が対応するカーネルバージョン番号になります。

4.4. ユーザー空間用のメモリー保護キー (別名 PKU または PKEYS)

メモリー保護キーは、ページベースの保護を強化するメカニズムを提供しますが、アプリケーションが保護ドメインを変更した時にページテーブルを修正する必要はありません。これは、各ページテーブルエントリーで以前無視されていた 4 ビットを「保護キー」に割り当て、可能なキーを 16 個提供することで有効となります。

メモリー保護キーは、いくつかの Intel CPU チップセットのハードウェア機能です。プロセッサーがこの機能をサポートしているかどうかは、**pku** in **/proc/cpuinfo** の存在を確認します。

```
$ grep pku /proc/cpuinfo
```

この機能をサポートするために、CPU は各キーに対して 2 つの別々のビット (Access Disable と Write Disable) を持つ、ユーザーがアクセス可能な新しいレジスター (PKRU) を提供します。この新しいレジスターの読み取りおよび書き込み用の新しい命令が 2 つ (RDPKRU と WRPKRU) 存在します。

プログラミングの実例を含む詳しい情報は、**kernel-doc** パッケージが提供する **/usr/share/doc/kernel-doc-*/Documentation/x86/protection-keys.txt** を参照してください。

4.5. KERNEL ADRESS SPACE LAYOUT RANDOMIZATION

Kernel Adress Space Layout Randomization (KASLR) は 2 つの部分で構成され、これらが共に機能して Linux カーネルのセキュリティーを強化します。

- カーネルテキスト KASLR
- メモリー管理の KASLR

カーネルテキストの物理アドレスと仮想アドレスの場所が、個別にランダム化されます。カーネルの物理アドレスは 64 TB の任意の場所に配置できますが、カーネルの仮想アドレスは、[0xffffffff80000000, 0xffffffffc0000000] の間の 1 GB 領域に制限されます。

メモリー管理の KASLR には 3 つのセクションがあり、これらのセクションの開始アドレスは特定のエリアでランダム化されます。したがって、悪意のコードが、カーネルアドレス領域にその記号が置かれていることを知る必要がある場合に、KASLR は悪意のコードにカーネルの実行を挿入またはリダイレクトしないようにすることができます。

メモリー管理の KASLR には、以下の 3 つのセクションがあります。

- 直接マッピングセクション
- vmalloc セクション

- *vmemmap セクション*

*KASLR コードが Linux カーネルにコンパイルされ、デフォルトで有効になりました。明示的に無効にするには、**nokaslr** カーネルオプションをカーネルコマンドラインに追加します。*

4.6. ADVANCED ERROR REPORTING (AER)

4.6.1. AER とは

***Advanced Error Reporting** (**AER**) は、**Peripheral Component Interconnect Express** (**PCIe**) デバイスの拡張エラーレポートを提供するカーネル機能です。The **AER** カーネルドライバーは、以下を行うために **PCIe AER 機能をサポートする** root ポートをアタッチします。*

- *エラーが発生した場合の包括的なエラー情報を収集*
- *ユーザーにエラーを報告します。*
- *エラーのリカバリーアクションの実行*

例4.1 AER 出力例

```
Feb  5 15:41:33 hostname kernel: pcieport 10003:00:00.0: AER: Corrected error received:
id=ae00
Feb  5 15:41:33 hostname kernel: pcieport 10003:00:00.0: AER: Multiple Corrected error received:
id=ae00
Feb  5 15:41:33 hostname kernel: pcieport 10003:00:00.0: PCIe Bus Error: severity=Corrected,
type=Data Link Layer, id=0000(Receiver ID)
Feb  5 15:41:33 hostname kernel: pcieport 10003:00:00.0:   device [8086:2030] error
status/mask=000000c0/00002000
Feb  5 15:41:33 hostname kernel: pcieport 10003:00:00.0:    [ 6] Bad TLP
Feb  5 15:41:33 hostname kernel: pcieport 10003:00:00.0:    [ 7] Bad DLLP
Feb  5 15:41:33 hostname kernel: pcieport 10003:00:00.0: AER: Multiple Corrected error received:
id=ae00
Feb  5 15:41:33 hostname kernel: pcieport 10003:00:00.0: PCIe Bus Error: severity=Corrected,
type=Data Link Layer, id=0000(Receiver ID)
Feb  5 15:41:33 hostname kernel: pcieport 10003:00:00.0:   device [8086:2030] error
status/mask=00000040/00002000
```

*When **AER** はエラーをキャプチャーし、エラーメッセージをコンソールに送信します。エラーが修復可能である場合、コンソールの出力は警告になります。*

4.6.2. AER メッセージの収集および表示

*AER メッセージを収集および表示するには、rasdaemon プログラム **を使用します**。*

手順

1. ***rasdaemon** パッケージをインストールします。*

```
~]# yum install rasdaemon
```

2. ***rasdaemon** サービスを有効にして起動します。*

```
~]# systemctl enable --now rasdaemon
```

3. ***ras-mc-ctl*** *コマンドを実行して、ログに記録されたエラーの概要を表示する（-* ***summary*** *オプション）、またはエラーデータベースに保存されたエラーを表示します（-* ***errors*** *オプション）。*

```
~]# ras-mc-ctl --summary
~]# ras-mc-ctl --errors
```

関連資料

- ***rasdaemon*** *サービスの詳細は、*rasdaemon ***(8)*** *man ページを参照してください。*
- ***ras-mc-ctl サービスの詳細は、***ras ***-mc-ctl(8)*** *man ページを参照してください。*

第5章 手動のカーネルアップグレード

Red Hat Enterprise Linux カーネルは、サポートしているハードウェアとの整合性と互換性を保てるように、Red Hat Enterprise Linux カーネルチームがカスタムを構築します。リリースする前に、カーネルは Red Hat が定める厳格な品質保証テストセットをパスしなければなりません。

Red Hat Enterprise Linux カーネルは、**Yum** または **PackageKit** パッケージマネージャーを使用し、簡単にアップグレードして検証できるように、RPM 形式でパッケージされます。**PackageKit** は自動的に Red Hat コンテンツ配信ネットワークサーバーをクエリーし、カーネルパッケージなど利用可能な更新が含まれれるパッケージを通知します。

本章は、**yum** ではなく、**rpm** コマンドを使用して手動でカーネルパッケージを更新する必要のあるユーザーに**だけ**有用です。

警告

できる限り **Yum** または **PackageKit** パッケージマネージャーを使用して新しいカーネルをインストールしてください。理由は、これらのツールは常に新しいカーネルを **インストール** するからです。他の方法で現行バージョンを入れ替えると (失敗した場合) システムが起動できなくなる可能性が出てきます。

警告

Red Hat では、カスタムのカーネルは **サポートされていません**。ただし、ソリューションの記事からガイダンスを取得できます。

yum を使用したカーネルパッケージのインストールの詳細は、『システム管理者のガイド』の関連のセクションを参照してください。

Red Hat コンテンツ配信ネットワークに関する情報は、『システム管理者のガイド』の関連セクションを参照してください。

5.1. カーネルパッケージの概要

Red Hat Enterprise Linux には以下のカーネルパッケージが含まれています。

- **kernel**: シングルコア、マルチコア、マルチプロセッサーシステム用のカーネルが含まれます。
- **kernel-debug**: カーネル診断ができるように複数のデバッグオプションが有効になっているカーネルが含まれます。デバッグオプションが有効になっているとパフォーマンスが低下します。
- **kernel-devel**: **kernel** パッケージに対して、モジュールを構築するのに十分なカーネルヘッダーと makefiles を含んでいます。

- ***kernel-debug-devel***: カーネル診断ができるように複数のデバッグオプションが有効になっている開発バージョンのカーネルが含まれます。 デバッグオプションが有効になっているとパフォーマンスが低下します。

- ***kernel-doc***: カーネルソースからのドキュメントファイルです。これらのファイルには、同梱で配布される Linux カーネルとデバイスドライバーのさまざまな部分が文書化されています。このパッケージをインストールすると、オプションへの参照が提供され、読み込み時に Linux カーネルモジュールに渡すことができます。
  デフォルトでは、これらのファイルは **/usr/share/doc/kernel-doc-**kernel_version/ ディレクトリーに配置されています。

- ***kernel-headers***: Linux カーネルと、ユーザー空間ライブラリーおよびプログラムとの間のインターフェースを指定する C ヘッダーファイルが含まれます。ヘッダーファイルは、ほとんどの標準プログラムを構築するのに必要な構造と定数を定義します。

- ***linux-firmware***: さまざまなデバイス操作に必要なファームウェアすべてを含んでいます。

- ***perf***: Linux カーネルのパフォーマンスモニタリングを有効にする **perf** ツールを含んでいます。

- ***kernel-abi-whitelists***: Red Hat Enterprise Linux カーネル ABI に関連する情報を含んでいます。これには、外部の Linux カーネルモジュールが必要とするカーネル記号の一覧と実施を支援する **yum** プラグインが含まれます。

- ***kernel-tools***: Linux カーネル操作のツールとサポートドキュメントが含まれています。

5.2. アップグレードへの準備

カーネルをアップグレードする前に、予防的な前準備手順の実行をお薦めします。

まず、問題が発生した場合に備えてシステム用に機能するブートメディアがあることを確認します。ブートローダーで新しいカーネルをブートするように正しく設定されていない場合には、このメディアを使って Red Hat Enterprise Linux をブートすることができます。

USB メディアは多くの場合、**ペンドライブ**、**サムディスク** または **キー** などと呼ばれるフラッシュデバイスの形式、または、外部接続のハードディスクとして提供されています。このタイプのほとんどすべてが ***VFAT*** ファイルシステムとしてフォーマットされています。ただし、***ext2***、***ext3***、***ext4*** または ***VFAT*** としてフォーマットされているメディア上でブート可能な USB メディアを作成することができます。

ディストリビューションのイメージファイル、または最低限ブートメディア (minimal boot media) イメージを USB メディアに転送することができます。デバイスには十分な空き領域があることを確認してください。約 4 GB がディストリビューション DVD イメージ用に必要で、約 700 MB がディストリビューション CD イメージ用に、そして約 10 MB が最低限ブートメディアイメージ用に必要です。

Red Hat Enterprise Linux のインストール DVD、またはインストール CD-ROM#1 からの ***boot.iso*** ファイルのコピーと、約 16 MB の空き領域を持つ ***VFAT*** ファイルシステムでフォーマットした USB ストレージデバイスが必要になります。

USB ストレージデバイスの使用方法は『How to format a USB key』 および『How to manually mount a USB flash drive in a non-graphical environment solution』を参照してください。

以下の手順では、コピー先のファイルと同じパス名を指定しなければ、USB ストレージデバイス上にある既存のファイルは影響を受けません。USB ブートメディアを作成するには、**root** ユーザーとして以下のコマンドを実行します。

1. ***syslinux*** パッケージがインストールされていない場合は、これをインストールします。root として ***yum install syslinux*** コマンドを実行します。

2. USB ストレージデバイス上に **SYSLINUX** ブートローダーをインストールします。

```
# syslinux /dev/sdX1
```

ここでの **sdX** はデバイス名です。

3. ***boot.iso*** と USB ストレージデバイス用にマウントポイントを作成します。

```
# mkdir /mnt/isoboot /mnt/diskboot
```

4. ***boot.iso*** をマウントします。

```
# mount -o loop boot.iso /mnt/isoboot
```

5. USB ストレージデバイスをマウントします。

```
# mount /dev/sdX1 /mnt/diskboot
```

6. ***ISOLINUX*** ファイルを ***boot.iso*** から USB ストレージデバイスにコピーします。

```
# cp /mnt/isoboot/isolinux/* /mnt/diskboot
```

7. ***boot.iso*** からの ***isolinux.cfg*** ファイルを USB デバイス用の ***syslinux.cfg*** ファイルとして使用します。

```
# grep -v local /mnt/isoboot/isolinux/isolinux.cfg > /mnt/diskboot/syslinux.cfg
```

8. ***boot.iso*** と USB ストレージデバイスをアンマウントします。

```
# umount /mnt/isoboot /mnt/diskboot
```

9. このブートメディアでマシンを再起動して、ブートできることを確認してから他の操作に移ります。

別の方法として、フロッピードライブがあるシステム上でブートディスクを作成するには、***mkbootdisk*** パッケージをインストールして、***root*** として ***mkbootdisk*** コマンドを実行します。このパッケージをインストールした後に使用法について確認するには、***man mkbootdisk*** の man ページを参照してください。

どのカーネルがインストールされているかを判定するには、シェルプロンプトでコマンド ***yum list installed "kernel-*"*** を実行します。出力では、システムのアーキテクチャーに応じて、以下のパッケージのすべてか、または一部が示されます。バージョン番号は異なる場合があります。

```
# yum list installed "kernel-*"
kernel.x86_64                3.10.0-54.0.1.el7          @rhel7/7.0
kernel-devel.x86_64          3.10.0-54.0.1.el7          @rhel7
kernel-headers.x86_64        3.10.0-54.0.1.el7          @rhel7/7.0
```

この出力から、カーネルのアップグレード用にダウンロードすべきパッケージを判断します。シングルプロセッサーのシステムでは、必要なパッケージは **kernel** パッケージのみです。異なるパッケージの説明は、「カーネルパッケージの概要」 を参照してください。

5.3. アップグレードされたカーネルのダウンロード

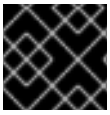
システム用に更新されたカーネルが利用可能かを判定する手段は数種類あります。

- セキュリティーエラータ – セキュリティー問題を修復するカーネルを含む、セキュリティーエラータについての詳細は、カスタマーポータルを参照してください。
- Red Hat コンテンツ配信ネットワーク: Red Hat コンテンツ配信ネットワークにサブスクライブするシステムについては、**yum** パッケージマネージャーが最新カーネルをダウンロードし、システム上のカーネルをアップグレードできます。**Dracut** ユーティリティーは、必要な場合は初期 RAM ファイルシステムイメージを作成し、新規カーネルを起動するためにブートローダーを設定します。Red Hat コンテンツ配信ネットワークからパッケージをインストールする方法は、『システム管理者のガイド』の関連のセクションを参照してください。Red Hat コンテンツ配信ネットワークにシステムを登録する方法は、『システム管理者のガイド』の関連のセクションを参照してください。

yum を使用して Red Hat Network から更新されたカーネルをダウンロードしてインストールする場合は、「初期 RAM ファイルシステムイメージの確認」 および 「ブートローダーの確認」 の手順のみに従ってください。カーネルはデフォルトで起動するように変更 **しないでください**。カーネルはデフォルトで起動するように変更しないでください。カーネルを手動でインストールするには、「アップグレードの実行」 に進みます。

5.4. アップグレードの実行

必要なパッケージをすべて取り込んだ後は、既存カーネルをアップグレードします。



重要

新しいカーネルに問題がある場合を考え、古いカーネルの維持を強く推奨します。

シェルプロンプトで、カーネル RPM パッケージを格納しているディレクトリーに移動します。**rpm** コマンドに **-i** 引数を使用して古いカーネルを残します。**-U** オプションは現在インストールしてあるカーネルを上書きして、ブートローダーの問題を起こすので、これは **使用しないでください**。以下に例を示します。

```
# rpm -ivh kernel-kernel_version.arch.rpm
```

次の手順では、初期 RAM ファイルシステムイメージが作成されているかどうかを検証します。詳細は、「初期 RAM ファイルシステムイメージの確認」 を参照してください。

5.5. 初期 RAM ファイルシステムイメージの確認

初期 RAM ファイルシステムイメージの仕事は、IDE、SCSI、RAID などのブロックデバイスモジュールをプレロードすることです。そうすることで、それらのモジュールが通常配備されている root ファイルシステムがアクセス可能になりマウントできるようになります。 Red Hat Enterprise Linux 7 システムでは、パッケージマネージャーの **Yum**、**PackageKit**、または **RPM** のいずれかを使用して新しいカーネルをインストールする場合は、常に **Dracut** ユーティリティーがインストールスクリプトにより呼び出され、**initramfs** (初期 RAM ファイルシステムイメージ) を作成します。

/etc/sysctl.conf ファイルまたは別の **sysctl** 設定ファイルを変更してカーネル属性を変更し、変更した設定がブートプロセスの初期段階で使用される場合には、**dracut -f** コマンドを使用して、初期 RAM ファイルシステムイメージの再構築が必要な場合があります。たとえば、ネットワーク関連の変更を受けて、ネットワークにアタッチされたストレージから起動する場合などです。

IBM eServer System i（「IBM eServer System i 上の初期 RAM ファイルシステムイメージとカーネルの検証」を参照）以外のすべてのアーキテクチャー上では、**dracut** コマンドを実行すると **initramfs** を作成できます。ただし、**initramfs** は手動で作成する必要はありません。このステップは、カーネルとその関連パッケージが Red Hat 配布の RPM パッケージからインストールされているか、またはアップグレードされている場合には自動的に実行されます。

現在のカーネルバージョンに該当する **initramfs** が存在していること、それが **grub.cfg** 設定ファイル内で正しく指定されているかを検証するには、以下の手順にしたがいます。

初期 RAM ファイルシステムイメージの確認

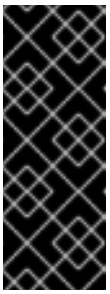
1. **root** として、**/boot** ディレクトリーのコンテンツを一覧表示して、カーネル (**vmlinuz-**kernel_version) と最新のバージョン番号を持つ **initramfs-**kernel_version を見つけます。

   例5.1 カーネルと initramfs バージョンの一致を確認

   ```
   # ls /boot
   config-3.10.0-67.el7.x86_64
   config-3.10.0-78.el7.x86_64
   efi
   grub
   grub2
   initramfs-0-rescue-07f43f20a54c4ce8ada8b70d33fd001c.img
   initramfs-3.10.0-67.el7.x86_64.img
   initramfs-3.10.0-67.el7.x86_64kdump.img
   initramfs-3.10.0-78.el7.x86_64.img
   initramfs-3.10.0-78.el7.x86_64kdump.img
   initrd-plymouth.img
   symvers-3.10.0-67.el7.x86_64.gz
   symvers-3.10.0-78.el7.x86_64.gz
   System.map-3.10.0-67.el7.x86_64
   System.map-3.10.0-78.el7.x86_64
   vmlinuz-0-rescue-07f43f20a54c4ce8ada8b70d33fd001c
   vmlinuz-3.10.0-67.el7.x86_64
   vmlinuz-3.10.0-78.el7.x86_64
   ```

   例5.1「カーネルと initramfs バージョンの一致を確認」 は以下を示しています。

   - 3 つのカーネルがインストールされています (より正確には、3 つのカーネルファイルが **/boot** ディレクトリーにあります)。
   - 最新のカーネルは **vmlinuz-3.10.0-78.el7.x86_64** です。
   - そのカーネルバージョンに一致する **initramfs** ファイルである **initramfs-3.10.0-78.el7.x86_64kdump.img** も存在します。

   重要

   /boot ディレクトリーで、複数の **initramfs-**kernel_version**kdump.img** ファイルが見つかる場合があります。それらは、カーネルのデバッグ目的で **Kdump** メカニズムで作成される特殊ファイルであり、システムの起動には使用されず、無視しても問題ありません。**kdump** の詳細は、「Red Hat Enterprise Linux 7 カーネルクラッシュダンプガイド」を参照してください。

2. 使用している **initramfs-**kernel_version ファイルが、**/boot** ディレクトリーにある最新カーネルのバージョンと一致しない場合、または他の特定の状況では、Dracut ユーティリティーを使用して **initramfs** ファイルを生成する必要がある場合があります。**root** としてオプションなしで **dracut** を呼び出すと、それが **/boot** 内にある最新のカーネル用に **initramfs** ファイルを生成するようになります。

```
# dracut
```

dracut が既存の **initramfs** を上書きするには (たとえば、**initramfs** が破損している場合など)、**-f**、**--force** オプションを使用する必要があります。これを使用しないと、**dracut** は既存の **initramfs** ファイルの上書きを拒否します。

```
# dracut
        Does not override existing initramfs (/boot/initramfs-3.10.0-78.el7.x86_64.img) without --force
```

現在のディレクトリーに initramfs を作成するには、**dracut** initramfs_name kernel_version を呼び出します。

```
# dracut "initramfs-$(uname -r).img" $(uname -r)
```

プレロードするカーネルモジュールを指定する必要がある場合には、**/etc/dracut.conf** 設定ファイルの **add_dracutmodules+="**module more_modules **"** ディレクティブの括弧の中に (**.ko** などの任意のファイル名のサフィックスを取り除いて) 対象のモジュール名を追加します。dracut で作成した **initramfs** イメージファイルのファイルコンテンツを一覧表示するには、**lsinitrd** initramfs_file コマンドを使用します。

```
# lsinitrd /boot/initramfs-3.10.0-78.el7.x86_64.img
Image: /boot/initramfs-3.10.0-78.el7.x86_64.img: 11M
========================================================================

dracut-033-68.el7
========================================================================


drwxr-xr-x  12 root     root            0 Feb  5 06:35 .
drwxr-xr-x   2 root     root            0 Feb  5 06:35 proc
lrwxrwxrwx   1 root     root           24 Feb  5 06:35 init -> /usr/lib/systemd/systemd
drwxr-xr-x  10 root     root            0 Feb  5 06:35 etc
drwxr-xr-x   2 root     root            0 Feb  5 06:35 usr/lib/modprobe.d
[output truncated]
```

オプションと用途に関する詳しい情報は **man dracut** と **man dracut.conf** を参照してください。

3. **/boot/grub2/grub.cfg** 設定ファイルを検査して、起動中のカーネルバージョンについて **initramfs-**kernel_version**.img** ファイルが存在することを確認します。以下は例になります。

```
# grep initramfs /boot/grub2/grub.cfg
initrd16 /initramfs-3.10.0-123.el7.x86_64.img
initrd16 /initramfs-0-rescue-6d547dbfd01c46f6a4c1baa8c4743f57.img
```

詳細は、「ブートローダーの確認」 を参照してください。

IBM eServer System i 上の初期 RAM ファイルシステムイメージとカーネルの検証

IBM eServer System i のマシンでは、初期 RAM ファイルシステムとカーネルファイルは 1 つのファイルに統合してあり、これは **addRamDisk** コマンドで作成されます。カーネルとその関連パッケージがインストールされているか、または Red Hat 配布の RPM パッケージでアップグレードされている場合は、このステップは自動的に実行されるので、手動で実行する必要はありません。このファイルが作成されていることを確認するには、**root** で以下のコマンドを実行して **/boot/vmlinitrd-kernel_version** ファイルがすでに存在することを確認します。

```
# ls -l /boot/
```

kernel_version は、先程インストールしたカーネルバージョンと一致する必要があります。

初期 RAM ファイルシステムイメージへの変更を戻す方法

たとえば、システムの設定を間違えたことで起動しなくなったような場合は、以下の手順に従って初期 RAM ファイルシステムイメージに加えた変更を戻す必要があります。

初期 RAM ファイルシステムイメージへの変更を戻す方法

1. GRUB メニューでレスキューカーネルを選択してシステムを再起動します。
2. **initramfs** の誤動作を引き起こしている間違った設定を変更します。
3. root で以下のコマンドを実行して、正しい設定で **initramfs** を作成し直します。

```
# dracut --kver kernel_version --force
```

上記の手順は、**sysctl.conf** ファイルで **vm.nr_hugepages** を間違って設定してしまった場合などに便利です。**sysctl.conf** ファイルは **initramfs** に含まれているため、新たな **vm.nr_hugepages** 設定は **initramfs** で適用されてしまい、**initramfs** が再構築されてしまいます。ただし、設定が間違っているので、新規の **initramfs** は破損しており、新規に構築されるカーネルは起動しないため、上記の手順を使用した設定の修正が必要になります。

初期 RAM ファイルシステムイメージのコンテンツの一覧表示

initramfs に含まれるファイルを一覧表示するには、root で以下のコマンドを実行します。

```
# lsinitrd
```

/etc ディレクトリーにあるファイルだけを表示するには、以下のコマンドを使用します。

```
# lsinitrd | grep etc/
```

現行カーネルの **initramfs** に保存されている特定ファイルのコンテンツを出力するには、**-f** オプションを使用します。

```
# lsinitrd -f filename
```

たとえば、**sysctl.conf** のコンテンツを出力するには、以下のコマンドを実行します。

```
# lsinitrd -f /etc/sysctl.conf
```

カーネルのバージョンを指定するには、**--kver** オプションを使用します。

```
# lsinitrd --kver kernel_version -f /etc/sysctl.conf
```

たとえば、カーネルバージョン 3.10.0-327.10.1.el7.x86_64 に関する情報を一覧表示するには、以下のコマンドを使用します。

```
# lsinitrd --kver 3.10.0-327.10.1.el7.x86_64 -f /etc/sysctl.conf
```

5.6. ブートローダーの確認

yum *コマンドまたは* ***rpm*** *コマンドで、カーネルをインストールしてください。*

rpm *を使用してカーネルをインストールすると、カーネルパッケージはブートローダー設定ファイル内にその新しいカーネル用のエントリーを作成します。*

*いずれのコマンドも、****/etc/sysconfig/kernel*** *設定ファイルに以下の設定を含める場合にのみ、新しいカーネルがデフォルトのカーネルとして起動するよう設定することに留意してください。*

```
DEFAULTKERNEL=kernel
UPDATEDEFAULT=yes
```

DEFAULTKERNEL *オプションは、デフォルトのカーネルパッケージタイプを指定します。****UPDATEDEFAULT*** *オプションは、新規カーネルパッケージがデフォルトで新しいカーネルにするかを指定します。*

第6章 カーネルライブパッチでパッチの適用

Red Hat Enterprise Linux カーネルのライブパッチソリューションを使用して、システムの再起動またはプロセスの再起動を行わずに、実行中のカーネルにパッチを当てることができます。

このソリューションでは、システム管理者は以下を行うことができます。

- 重大なセキュリティーパッチをカーネルに即座に適用することが可能。
- 長時間実行しているタスクの完了、ユーザーのログオフ、スケジュールダウンタイムを待つ必要がない。
- システムのアップタイムをより制御し、セキュリティーや安定性を犠牲にしない。

重要な、重要なすべての CVE は、カーネルライブパッチソリューションで解決されるわけではありません。この目的は、セキュリティー関連パッチに必要な再起動を減らすことであり、完全になくすことではありません。ライブパッチの範囲の詳細は、**「RHEL 7 はライブカーネルパッチ (kpatch) をサポートしていますか?」** を参照してください。

警告

カーネルのライブマイグレーションパッチと、その他のカーネルサブコンポーネントとの間に、いくらか非互換性が存在します。カーネルのライブパッチを使用する前に、*「kpatch の制限」* セクションを慎重に確認してください。

6.1. KPATCH の制限

- **kpatch** 機能は、汎用のカーネルアップグレードメカニズムではありません。システムをすぐに再起動できない場合など、単純なセキュリティーおよびバグ修正の更新を適用する場合に使用します。
- パッチの読み込み中または読み込み後は、**SystemTap** ツールまたは **kprobe** ツールを使用しないでください。このようなプローブが削除されるまでは、パッチが適用できなくなる可能性があります。

6.2. サードパーティーのライブパッチサポート

kpatch ユーティリティーは、Red Hat リポジトリー提供の RPM モジュールを含む、Red Hat がサポートする唯一のカーネルライブパッチユーティリティーです。Red Hat は、Red Hat 提供でないライブカーネルパッチはサポートしません。

サードパーティーのライブパッチのサポートは、パッチを提供しているベンダーにお問い合わせください。

サードパーティーのライブパッチを実行しているシステムの場合、Red Hat は、Red Hat が同梱し、サポートしているソフトウェアの複製を求める権利を有します。これが可能でない場合、Red Hat は、同じ動作が発生するかどうかを確認するために、ライブパッチを適用せずに、お使いのテスト環境で同じようなシステムとワークロードの展開を求めます。

サードパーティソフトウェアサポートポリシーの詳細は、*「Red Hat グローバルサポートサービスは、*

サードパーティーのソフトウェア、ドライバー、そして認定されていないハードウェアおよびハイパーバイザー、もしくはゲストのオペレーティングシステムについてどのようなサポートを提供していますか?」を参照してください。

6.3. カーネルライブパッチへのアクセス

*ライブのカーネルパッチ機能は、RPM パッケージとして提供されるカーネルモジュール (**.ko** ファイル) として実装されます。*

すべてのお客様は、通常のチャンネルから提供されるカーネルライブパッチにアクセスできます。ただし、延長サポートサービスにサブスクライブしていないお客様は、次のマイナーリリースが利用可能になると、現行のマイナーリリースに対する新しいパッチへのアクセスを失うことになります。たとえば、標準のサブスクリプションを購入しているお客様は、RHEL 8.3 がリリースされるまで RHEL 8.2 のライブパッチのみを行うことができます。

6.4. カーネルライブパッチのコンポーネント

カーネルのライブパッチのコンポーネントは、以下のようになります。

カーネルパッチモジュール

- *カーネルライブパッチの配信メカニズム*
- *パッチが適用されるカーネル用に構築したカーネルモジュール。*
- *パッチモジュールには、カーネルに必要な修正のコードが含まれます。*
- *パッチモジュールは、**kpatch** カーネルサブシステムで登録し、置き換えられるオリジナル機能の情報を提供します。また、置換される機能に一致するポインターも含まれます。カーネルパッチモジュールは RPM として提供されます。*
- *命名規則は、**kpatch_<kernel version>_<kpatch version>_<kpatch release>** です。名前の「kernel version」の部分の **ピリオド** および **ハイフン** が **アンダースコア** に置き換えられています。*

kpatch ユーティリティー

パッチモジュールを管理するためのコマンドラインユーティリティー。

kpatch サービス

***multiuser.target** で必要な **systemd** サービス。このターゲットは、システムの起動時にカーネルパッチをロードします。*

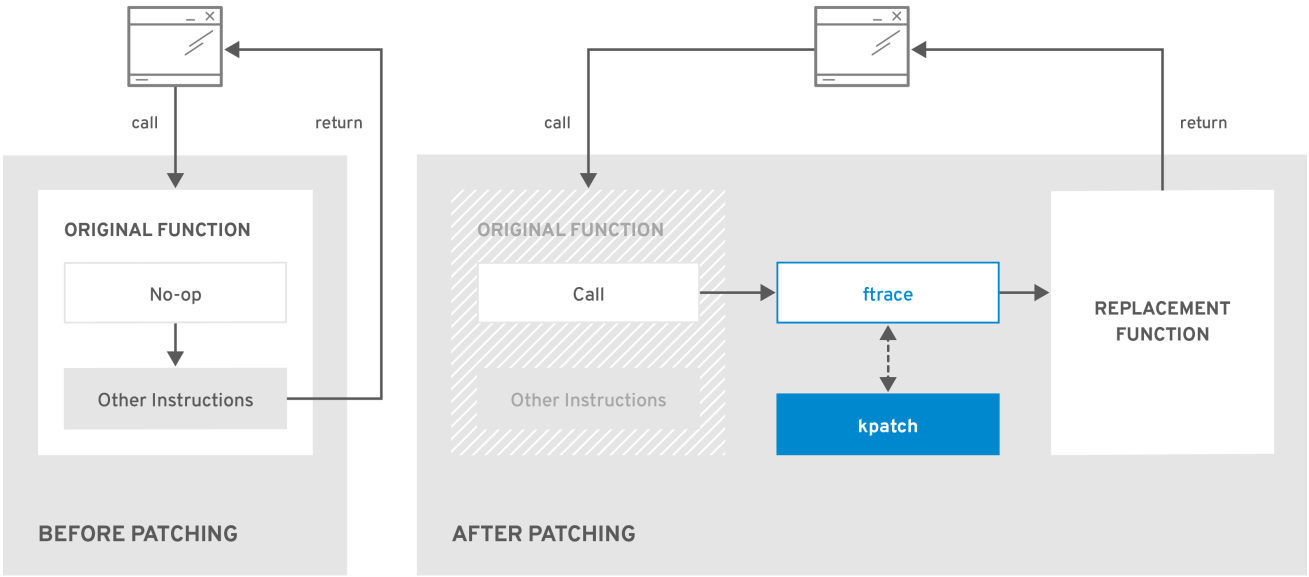
6.5. カーネルライブパッチの仕組み

***kpatch** カーネルパッチソリューションは、**livepatch** カーネルサブシステムを使用して、古い機能の出力先を新しい機能に変更します。ライブカーネルパッチがシステムに適用されると、以下が発生します。*

1. *カーネルパッチモジュールは、**/var/lib/kpatch/** ディレクトリーにコピーされ、次回の起動時に **systemd** を介して、カーネルへの再適用として登録されます。*
2. *実行中のカーネルに kpatch モジュールがロードされ、新しいコードのメモリー内の場所を指定するポインターを使用して、このパッチが適用された機能が **ftrace** メカニズムに登録されます。*

3. パッチが当てられた機能にカーネルがアクセスすると、**ftrace** メカニズムにリダイレクトされます。これにより、元々の機能は回避され、パッチを当てたバージョンの機能にカーネルをリダイレクトします。

図6.1 カーネルライブパッチの仕組み



6.6. カーネルライブパッチの有効化

カーネルパッチモジュールは RPM パッケージに含まれ、パッチが適用されたカーネルバージョンに固有のものとなります。各 RPM パッケージは、徐々に蓄積されていきます。

以下のサブセクションでは、指定のカーネルに対して、将来のすべての累積パッチの更新を受け取る方法を説明します。

警告

Red Hat は、Red Hat がサポートするシステムに適用されたサードパーティーのライブパッチをサポートしません。

6.6.1. ライブパッチストリームへのサブスクライブ

この手順では、特定のライブパッチパッケージをインストールする方法を説明します。そうすることで、指定のカーネルのライブパッチストリームをサブスクライブし、今後のカーネルに対する累計なライブパッチ更新をすべて受けることができます。

警告

ライブパッチは累積的であるため、特定のカーネルにデプロイされている個々のパッチを選択できません。

前提条件

- root 権限

手順

1. 必要に応じて、カーネルバージョンを確認します。

```
# uname -r
3.10.0-1062.el7.x86_64
```

2. カーネルのバージョンに一致するライブパッチパッケージを検索します。

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

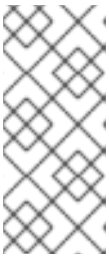
3. ライブパッチパッケージをインストールします。

```
# yum install "kpatch-patch = $(uname -r)"
```

上記のコマンドでは、特定カーネルにのみに最新の累積パッチをインストールし、適用します。

パッケージのバージョンが 1-1 以上であれば、ライブパッチパッケージには、パッチモジュールが含まれます。この場合、ライブパッチパッケージのインストール時に、カーネルにパッチが自動的に適用されます。

カーネルパッチモジュールは、今後の再起動時に **systemd** システムおよびサービスマネージャーにより読み込まれる **/var/lib/kpatch/** ディレクトリーにもインストールされます。



注記

指定のカーネルに利用可能なライブパッチがない場合は、空のライブパッチパッケージがインストールされます。空のライブパッチパッケージには、kpatch_version-kpatch_release 0-0 (例: **kpatch-patch-3_10_0-1062-0-0.el7.x86_64.rpm**) が含まれます。空の RPM のインストールを行うと、指定のカーネルの将来のすべてのライブパッチにシステムがサブスクライブされます。

4. 必要に応じて、カーネルがパッチを当てていることを確認します。

```
# kpatch list
Loaded patch modules:
kpatch_3_10_0_1062_1_1 [enabled]
```

```
Installed patch modules:
kpatch_3_10_0_1062_1_1 (3.10.0-1062.el7.x86_64)
…
```

この出力は、カーネルパッチモジュールがカーネルに読み込まれていることを示しています。つまり、カーネルは現在、**kpatch-patch-3_10_0-1062-1-1.el7.x86_64.rpm** パッケージの最新の修正でパッチが当てられています。

関連情報

- **kpatch** コマンドラインユーティリティーの詳細は、man ページの **kpatch(1)** を参照してください。
- RHEL 7 のソフトウェアパッケージの詳細は、**『システム管理者のガイド** の関連セクションを参照してください。

6.7. カーネルパッチモジュールの更新

カーネルパッチモジュールが配信され、RPM パッケージを通じて適用されているため、累計のカーネルパッチモジュール更新は、他の RPM パッケージの更新と似ています。

前提条件

- root 権限
- 「ライブパッチストリームへのサブスクライブ」 の説明に従って、システムがライブパッチストリームにサブスクライブされています。

手順

- 現在のカーネルの新しい累積バージョンを更新します。

```
# yum update "kpatch-patch = $(uname -r)"
```

上記のコマンドは、現在実行中のカーネルに利用可能な更新を自動的にインストールし、適用します。これには、新たにリリースされた累計なライブパッチが含まれます。

- もしくは、インストールしたすべてのカーネルパッチモジュールを更新します。

```
# yum update "kpatch-patch*"
```

注記

システムが同じカーネルで再起動すると、**kpatch.service** サービスにより、カーネルが自動的に再適用されます。

関連情報

- ソフトウェアパッケージの更新の詳細情報は、**『システム管理者のガイド』** の関連のセクションを参照してください。

6.8. カーネルライブパッチの無効化

システム管理者が、Red Hat Enterprise Linux カーネルライブパッチソリューション関連の不足の悪影響に遭遇した場合は、このメカニズムを無効化する選択肢があります。以下のセクションでは、ライブパッチソリューションを無効にする方法を説明します。

重要

現在、Red Hat はシステムの再起動なしで、ライブパッチを元に戻すことはサポートしていません。ご不明な点がございましたら、サポートチームまでお問い合わせください。

6.8.1. ライブパッチパッケージの削除

以下の手順は、ライブパッチパッケージを削除して、Red Hat Enterprise Linux カーネルのライブパッチソリューションを無効にする方法を説明します。

前提条件

- root 権限
- ライブパッチパッケージがインストールされている。

手順

1. ライブパッチパッケージを選択します。

```
# yum list installed | grep kpatch-patch
kpatch-patch-3_10_0-1062.x86_64        1-1.el7        @@commandline
…
```

上記の出力例は、インストールしたライブパッチパッケージを一覧表示します。

2. ライブパッチパッケージを削除します。

```
# yum remove kpatch-patch-3_10_0-1062.x86_64
```

ライブパッチパッケージが削除されると、カーネルは次回の再起動までパッチが当てられたままになりますが、カーネルパッチモジュールはディスクから削除されます。次回の再起動後に、この一致するカーネルにはパッチが適用されません。

3. システムを再起動します。

4. ライブパッチパッケージが削除されたことを確認します。

```
# yum list installed | grep kpatch-patch
```

パッケージが正常に削除された場合、このコマンドでは何も出力されません。

5. 必要に応じて、カーネルのライブパッチソリューションが無効になっていることを確認します。

```
# kpatch list
Loaded patch modules:
```

この出力例では、現在読み込まれているパッチモジュールがないため、カーネルにパッチが適用されておらず、ライブパッチソリューションがアクティブでないことが示されています。

関連情報

- **kpatch** コマンドラインユーティリティーの詳細は、man ページの **kpatch(1)** を参照してください。
- ソフトウェアパッケージの使用方法は、『システム管理者のガイド』 の関連のセクションを参照してください。

6.8.2. カーネルパッチモジュールのアンインストール

以下の手順では、Red Hat Enterprise Linux カーネルライブパッチソリューションが、後続のブートでカーネルパッチモジュールを適用しないようにする方法を説明します。

前提条件

- root 権限
- ライブパッチパッケージがインストールされている。
- カーネルパッチモジュールがインストールされ、ロードされている。

手順

1. カーネルパッチモジュールを選択します。

```
# kpatch list
Loaded patch modules:
kpatch_3_10_0_1062_1_1 [enabled]

Installed patch modules:
kpatch_3_10_0_1062_1_1 (3.10.0-1062.el7.x86_64)
…
```

2. 選択したカーネルパッチモジュールをアンインストールします。

```
# kpatch uninstall kpatch_3_10_0_1062_1_1
uninstalling kpatch_3_10_0_1062_1_1 (3.10.0-1062.el7.x86_64)
```

- アンインストールしたカーネルモジュールが読み込まれていることに注意してください。

```
# kpatch list
Loaded patch modules:
kpatch_3_10_0_1062_1_1 [enabled]

Installed patch modules:
<NO_RESULT>
```

選択したモジュールをアンインストールすると、カーネルは次回の再起動までパッチが当てられますが、カーネルパッチモジュールはディスクから削除されます。

3. システムを再起動します。
4. 必要に応じて、カーネルパッチモジュールがアンインストールされていることを確認します。

```
# kpatch list
Loaded patch modules:
```

-

上記の出力例では、ロードまたはインストールされたカーネルパッチモジュールが表示されていません。したがって、カーネルにパッチが適用されておらず、カーネルのライブパッチソリューションはアクティブではありません。

関連情報

- ***kpatch** コマンドラインユーティリティーの詳細は、man ページの **kpatch(1)** を参照してください。*

6.8.3. kpatch.service の無効化

以下の手順では、Red Hat Enterprise Linux カーネルライブパッチソリューションが、後続のブートでカーネルパッチモジュールをグローバルに適用しないようにする方法を説明します。

前提条件

- *root 権限*
- *ライブパッチパッケージがインストールされている。*
- *カーネルパッチモジュールがインストールされ、ロードされている。*

手順

1. ***kpatch.service** が有効化されていることを確認します。*

```
# systemctl is-enabled kpatch.service
enabled
```

2. ***kpatch.service** を無効にします。*

```
# systemctl disable kpatch.service
Removed /etc/systemd/system/multi-user.target.wants/kpatch.service.
```

   - *適用されたカーネルモジュールが依然としてロードされていることに注意してください。*

```
# kpatch list
Loaded patch modules:
kpatch_3_10_0_1062_1_1 [enabled]

Installed patch modules:
kpatch_3_10_0_1062_1_1 (3.10.0-1062.el7.x86_64)
```

3. *システムを再起動します。*
4. *必要に応じて、**kpatch.service** のステータスを確認します。*

```
# 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)
```

*この出力テストサンプルでは、**kpatch.service** が無効になっており、実行されていないことを証明しています。したがって、カーネルのライブパッチソリューションはアクティブではありません。*

5. *カーネルパッチモジュールがアンロードされたことを確認します。*

```
# kpatch list
Loaded patch modules:

Installed patch modules:
kpatch_3_10_0_1062_1_1 (3.10.0-1062.el7.x86_64)
```

上記の出力例では、カーネルパッチモジュールがインストールされていても、カーネルにパッチが適用されていないことを示しています。

関連情報

- ***kpatch** コマンドラインユーティリティーの詳細は、man ページの **kpatch(1)** を参照してください。*
- ***systemd** システムおよびサービスマネージャー、ユニット設定ファイル、その場所、および **systemd** ユニットタイプの完全なリストの詳細は、**『システム管理者のガイド』** の関連のセクションを参照してください。*

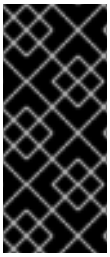
第7章 カーネルクラッシュダンプガイド

7.1. KDUMP について

7.1.1. kdump と kexec について

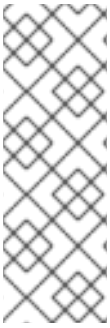
kdump とは、システムのメモリー内容を保存して後で分析できるようにカーネルのクラッシュをダンプする仕組みを指します。kdump は **kexec** に依存し、この kexec を使用して、別のカーネルのコンテキストから Linux カーネルを起動し、BIOS を迂回して通常は失われてしまう 1 番目のカーネルメモリーの内容を保持することができます。

システムクラッシュが発生すると kdump は kexec を使用して 2 番目のカーネルで起動します (**キャプチャーカーネル**)。この 2 番目のカーネルはシステムメモリーの予約部分に収納されていて 1 番目のカーネルからはアクセスできません。2 番目のカーネルは起動するとクラッシュしたカーネルメモリーの内容 (**クラッシュダンプ**) をキャプチャーして保存します。



重要

カーネルクラッシュダンプは、障害時に唯一利用可能な情報である可能性があるので、ビジネスに不可欠な環境ではこのデータの重要性を過小評価してはいけません。Red Hat は、システム管理者が通常のカーネル更新サイクルで **kexec-tools** を定期的に更新してテストすることをお勧めします。これは、新しいカーネル機能が実装されている場合に特に重要です。



注記

HP Watchdog タイマー (**hpwdt**) ドライバーは、RHEV ハイパーバイザーとして実行中されている HP システムで事前に読み込まれるため、これらのシステムは NMI ウォッチドッグを使用できます。**kexec-tools-2.0.15-33.el7.x86_64** から開始する更新された kexec-tools パッケージは **hpwdt** ドライバーを事前に読み込みます。

ドライバー **bnx2x** および **bmx2fc** が kdump カーネルのブラックリストに入れられ、2 つ目のカーネルがパニックを生じさせ、ダンプはキャプチャーされません。

7.1.2. メモリー要件

kdump でカーネルクラッシュダンプをキャプチャーしてさらに分析ができるように保存するにはシステムメモリーの一部をキャプチャーカーネル用に永続的に予約する必要があります。予約するとその部分はメインカーネルでは使用できなくなります。

メモリー要件は、特定のシステムパラメーターによって異なります。主な要因は、システムのハードウェアアーキテクチャーです。次のコマンドをシェルプロンプトで入力してマシンの正確なアーキテクチャー名を特定し (***x86_64*** など)、標準出力に表示させます。

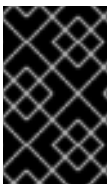
```
uname -m
```

予約すべきメモリーサイズを左右する別の要因として搭載しているシステムメモリーの総量が影響します。たとえば、x86_64 アーキテクチャーでは予約メモリーは 4 KB のメモリーごとに、160 MB + 2 ビットになります。搭載されている物理メモリーの合計が 1 TB のシステムの場合には 224 MB (160 MB + 64 MB) ということになります。システムアーキテクチャーごとの kdump のメモリー要件と物理メモリーの容量の完全リストは、「kdump メモリー要件」 を参照してください。

多くのシステムでは必要なメモリー量は kdump によって自動的に算出、予約が行われます。この動作

はデフォルトで有効になっていますが、利用可能な合計メモリーサイズが一定以上搭載されているシステムに限られます。この自動割り当て動作に必要なメモリーサイズはシステムのアーキテクチャーによって異なります。システムアーキテクチャーに基づいた自動メモリー予約については、「「メモリー自動予約の最小しきい値」」を参照してください。

システムメモリーが自動割り当ての動作に必要な最小メモリーに満たない場合、または独自の予約メモリーサイズを必要とするような場合には予約メモリーを手動で設定することができます。コマンドラインでこの作業を行う場合は「メモリー使用量の設定」を参照してください。グラフィカルユーザーインターフェースで予約メモリーの量を設定する方法は、「メモリー使用量の設定」を参照してください。



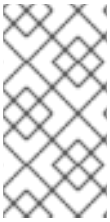
重要

kdump サービスを設定したら、自動メモリー割り当てであっても設定のテストを行うことを強く推奨します。設定のテスト方法については、「kdump 設定のテスト」を参照してください。

7.2. KDUMP のインストールと設定

7.2.1. kdump のインストール

多くの場合、Red Hat Enterprise Linux 7 の新規インストールで **kdump** サービスはデフォルトでインストールされます。グラフィカルインターフェースまたはテキストインターフェースを使って対話形式でインストールする場合には、**Anaconda** インストーラーに kdump の設定画面があります。インストーラーの画面には **kdump** というタイトルが付けられ、メイン画面の **インストールの概要** からアクセスすることができます。kdump のメモリー要件の詳細は、「kdump メモリー要件」を参照してください。インストーラーの kdump 設定画面は **『Red Hat Enterprise Linux 7 インストールガイド』** で説明しています。



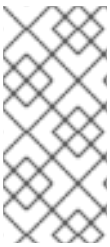
注記

Red Hat Enterprise Linux の以前のリリースでは **Firstboot** ユーティリティーで kdump の設定ができました。このユーティリティーはインストールの**終了後**、システムをはじめて再起動すると自動的に実行されていました。Red Hat Enterprise Linux 7.1 からは kdump の設定がインストーラー内に移動しています。

カスタムのキックスタートを使ったインストールなど一部のインストール方法では、デフォルトで kdump をインストールしない場合または有効にしない場合があります。このような場合に、kdump を追加でインストールするには **root** で次のコマンドをシェルプロンプトから実行します。

```
# yum install kexec-tools
```

お使いのシステムアーキテクチャー向けの **kexec-tools** パッケージが含まれるカスタムのリポジトリーか、アクティブなサブスクリプションがシステムにある場合に、上記のコマンドで kdump およびその他に必要なパッケージすべてが確実にインストールされます。



注記

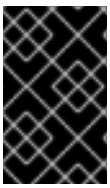
システムに kdump がインストールされているかわからない場合は **rpm** を使うと確認できます。

```
$ rpm -q kexec-tools
```

この他、グラフィカルな設定ツールもあります。ただし上記のコマンドを使った場合に、デフォルトではグラフィカルな設定ツールはインストールされません。「グラフィカルユーザーインターフェースでの kdump の設定」に記載されているこのユーティリティーをインストールするには、**root** で以下のコマンドを実行します。

```
# yum install system-config-kdump
```

Yum を使用した Red Hat Enterprise Linux 7 の新規パッケージのインストール方法は『Red Hat Enterprise Linux 7 システム管理者のガイド』を参照してください。



重要

Red Hat Enterprise Linux 7.4 から、**kdump** は **Intel IOMMU** ドライバーをサポートしています。7.3 以前のバージョンのカーネルを実行する場合は、**Intel IOMMU** のサポートを無効にすることを推奨します。

7.2.2. コマンドラインで kdump の設定

7.2.2.1. メモリー使用量の設定

kdump カーネル用に予約されるメモリーは必ずシステムの起動時にその予約が行われます。 つまり、メモリーのサイズはシステムのブートローダー設定で指定されています。

kdump カーネル用に割り当てるメモリーを指定するには、**crashkernel=** オプションを必要な値に設定します。たとえば、128 MB のメモリーを予約するには、以下を使用します。

```
crashkernel=128M
```

AMD64 システム、Intel 64 システム、および IBM Power Systems サーバーで **crashkernel=** オプションを変更する方法 、 および **zipl** を使用して IBM Z では、「カーネルコマンドラインパラメーターの設定」を参照してください。

crashkernel= オプションは、複数の方法で定義できます。**auto** の値により、「kdump メモリー要件」に記載されているガイドラインに従って、システムの合計メモリーに基づく予約メモリーの自動設定が可能になります。メモリーサイズが大きいシステムでは、オペレーティングシステムに設定された上限まで、**crashkernel=auto** オプションが指定されたアーキテクチャーに従って計算されます。

この動作を変更するには、**auto** の値を特定のメモリー量に置き換えます。

crashkernel= オプションは、特にメモリー量が少ないシステムで有用です。たとえば、128 MB のメモリーを予約するには、以下を使用します。

```
crashkernel=128M
```

搭載しているメモリーの合計サイズに応じて予約メモリーサイズが可変するように設定することもできます。可変のメモリー予約を設定する場合の構文は **crashkernel=<range1>:<size1>、<range2>:<size2>** になります。以下に例を示します。

```
crashkernel=512M-2G:64M,2G-:128M
```

上記の例では、システムメモリーの合計が 512 MB 以上 2 GB 未満の場合、64 MB のメモリーを予約します。メモリー合計サイズが 2 GB を超える場合は、128 MB が kdump 用に予約されます。

システムによっては、特定の固定オフセットを指定して、メモリーの予約を行う必要があります。オフセットが設定されると、予約メモリーはそこから開始されます。予約メモリーをオフセットするには、以下の構文を使用します。

```
crashkernel=128M@16M
```

上記の例の場合、kdump は 128 MB のメモリー予約を 16 MB (物理アドレス 0x01000000) から開始することになります。オフセットパラメーターを 0 に設定する、または完全に省略すると kdump により自動的にオフセットが設定されます。上記のように、変数メモリー予約を設定する場合にもこの構文を使用します。この場合、オフセットは常に最後に指定されます (例: **crashkernel=512M-2G:64M,2G-:128 _M@16 _M**)。

7.2.2.2. kdump タイプの設定

カーネルクラッシュがキャプチャーされると、コアダンプはローカルファイルシステムのファイルとして保存したり、デバイスに直接書き込みしたり、**NFS** (Network File System) または **SSH** (Secure Shell) プロトコルを使用してネットワーク上で送信したりすることができます。現時点で設定できるのは、これらのオプションの 1 つのみです。デフォルトのオプションでは、**vmcore** ファイルをローカルファイルシステムの **/var/crash** ディレクトリーに保存します。

vmcore ファイルをローカルファイルシステムの **/var/crash/** ディレクトリーに保存するには、次を行います。

- **/etc/kdump.conf** ファイルを編集し、パスを指定します。

  ```
  path /var/crash
  ```

  オプションの **path /var/crash** は、**kdump** が **vmcore** ファイルを保存するファイルシステムパスを表します。**/etc/kdump.conf** ファイルでダンプターゲットを指定すると、**path** は指定されたダンプ出力先に対する相対パスになります。

  ダンプターゲットが **/etc/kdump.conf** ファイルで指定されていない場合には、**パス** は root ディレクトリーからの絶対パスになります。現在のシステムにマウントされている内容に応じて、ダンプターゲットと調整されるダンプパスが自動的に適用されます。

警告

kdump は、ダンプターゲットが **/var/crash** にマウントされ、オプションの **path** が **/etc/kdump.conf** ファイルの **/var/crash** として設定されていると、**vmcore** ファイルを **/var/crash/var/crash** ディレクトリーに保存します。たとえば、以下の例では、**ext4** ファイルシステムが **/var/crash** ですでにマウントされており、**path** も **/var/crash** として設定されます。

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

そのため、**/var/crash/var/crash** パスが作成されます。この問題を解決するには、**path /var/crash** の代わりに **path /** オプションを使用します。

*ダンプの場所を変更するには、**root** としてテキストエディターで **/etc/kdump.conf** 設定ファイルを開き、以下のようにオプションを編集します。*

*コアダンプの保存先のローカルディレクトリーを変更する場合は **#path /var/crash** の行頭にあるハッシュ記号 (#) を取り除き、値を変更先のディレクトリーパスに置き換えます。*

```
path /usr/local/cores
```

重要

*Red Hat Enterprise Linux 7 では kdump のダンプ出力先として **path** ディレクティブで指定されているディレクトリーが **kdump** systemd サービスの起動時に存在していなければなりません。この動作は、サービスの起動時にそのディレクトリーが存在しない場合に自動的に作成されていた Red hat Enterprise Linux の以前のリリースのものとは異なります。*

*オプションで、ファイルを別のパーティションに書き込む場合は、**#ext4** で始まる行のハッシュ記号を取り除き、値を変更先のディレクトリーパスに置き換えます。値にはデバイス名 (**#ext4 /dev/vg/lv_kdump** 行)、ファイルシステムのラベル (**#ext4 LABEL=/boot** 行)、UUID (**#ext4 UUID=03138356-5e61-4ab3-b58e-27507ac41937** 行) のいずれかを使用できます。ファイルシステムタイプと、デバイス名、ラベル、UUID を希望の値に変更します。以下に例を示します。*

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

重要

*ストレージデバイスの指定は **LABEL=** または **UUID=** を使用することを推奨します。**/dev/sda3** などのディスクデバイス名は、再起動しても一貫性は保証されません。永続的なディスクデバイスの命名については『Red Hat Enterprise Linux 7 ストレージ管理ガイド』を参照してください。*

重要

*s390x ハードウェア上の DASD にダンプする場合には、続行する前に **/etc/dasd.conf** でダンプサービスが正しく指定されている必要があります。*

*ダンプをデバイスに直接書き込む場合は **#raw /dev/vg/lv_kdump** の行頭にあるハッシュ記号 (#) を取り除き、値をダンプ出力先のデバイス名に置き換えます。以下に例を示します。*

```
raw /dev/sdb1
```

***NFS** プロトコルを使ってリモートのマシンにダンプを保存する場合は **#nfs my.server.com:/export/tmp** の行頭にあるハッシュ記号 (#) を取り除き、値を有効なホスト名とディレクトリーパスに置き換えます。以下に例を示します。*

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

***SSH** プロトコルを使ってリモートのマシンにダンプを保存する場合は **#ssh user@my.server.com** の行頭にあるハッシュ記号 (#) を取り除き、値を有効なユーザー名とホスト名に置き換えます。設定に SSH キーも含める場合は **#sshkey /root/.ssh/kdump_id_rsa** 行の先頭にあるハッシュ記号 (#) を取り除き、値をダンプ出力先となるサーバー上で有効なキーの場所に変更します。以下に例を示します。*

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

SSH サーバーの設定方法およびキーベースの認証設定については『Red Hat Enterprise Linux 7 システム管理者のガイド』を参照してください。

対応しているダンプ出力先と非対応のダンプ出力先のタイプ別一覧は、表7.3「対応している kdump のダンプ出力先」 を参照してください。

7.2.2.3. コアコレクターの設定

vmcore ダンプファイルのサイズを小さくするために、**kdump** では外部アプリケーション (core collector) を指定して、データの圧縮や必要に応じた関連性のないすべての情報の除外ができます。現在、完全サポートしているコアコレクターは **makedumpfile** のみになります。

コアコレクターを有効にするには、**root** として、テキストエディターで **/etc/kdump.conf** 設定ファイルを開いて、**#core_collector makedumpfile -l --message-level 1 -d 31** の行頭にあるハッシュ記号 (#) を取り除き、以下の説明通りにコマンドラインのオプションを編集します。

ダンプファイルの圧縮を有効にするには、**-l** パラメーターを追加します。以下は例になります。

```
core_collector makedumpfile -l
```

ダンプから特定のページを削除するには、- **d** value を追加します。value は、表7.4「サポートしているフィルターレベル」 で説明されているように、省略するページの値の合計になります。ゼロと未使用ページを除外する場合は次のようになります。

```
core_collector makedumpfile -d 17 -c
```

利用可能なオプションの完全な一覧は、**makedumpfile(8)** man ページを参照してください。

7.2.2.4. デフォルト動作の設定

デフォルトでは、**kdump** が 「kdump タイプの設定」 で指定したダンプ出力先でコアダンプの作成に失敗すると、**kdump** は vmcore を保存せずにシステムを再起動します。この動作を変更するには、**root** でテキストエディターで **/etc/kdump.conf** 設定ファイルを開いて **#default shell** の行頭にあるハッシュ記号("#") を取り除き、表7.5「サポートしているデフォルトの動作」 の説明に従って値を希望のアクションに置き換えます。

以下は例になります。

```
default reboot
```

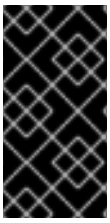
7.2.2.5. サービスの有効化

起動時に **kdump** デーモンを開始するには、シェルプロンプトで **root** として以下を入力します。

```
systemctl enable kdump.service
```

これにより、**multi-user.target** のサービスが有効になります。同様に **systemctl disable kdump** を入力すると **kdump** が無効になります。現在のセッションでサービスを開始する場合は **root** として、次のコマンドを使用します。

```
systemctl start kdump.service
```



重要

Red Hat Enterprise Linux 7 では kdump の出力先として指定されているディレクトリーが **kdump** systemd サービスの起動時に存在していなければなりません。この動作は、サービスの起動時にそのディレクトリーが存在しない場合に自動的に作成されていた Red hat Enterprise Linux の以前のリリースのものとは異なります。

systemd およびサービスの設定全般については『Red Hat Enterprise Linux 7 システム管理者のガイド』を参照してください。

7.2.3. グラフィカルユーザーインターフェースでの kdump の設定

Kernel Dump Configuration ユーティリティーを起動するにはパネルから **Activities** → **Other** → **Kernel crash dumps** の順で選択するか、シェルプロンプトで **system-config-kdump** を入力します。その結果、図7.1「基本設定」 にウィンドウが表示されます。

このユーティリティーを使用すると **kdump** の設定のほか、起動時にサービスを有効または無効にすることもできます。設定が完了したら **適用** をクリックして変更を保存します。認証が済んでいる場合を除きスーパーユーザーのパスワード入力が求められます。また、設定の変更を適用するにはシステムの再起動が必要な旨を示すメッセージが表示されます。

重要

SELinux が Enforcing モードで実行中の IBM Z または PowerPC システムでは、カーネルダンプ設定ユーティリティーを起動する前に kdumpgui_run_bootloader のブール値を有効にする必要があります。このブール値により、system-config-kdump が bootloader_t SELinux ドメインのブートローダーで実行できるようになります。このブール値を永続的に有効化するには、root として以下のコマンドを実行します。

```
# setsebool -P kdumpgui_run_bootloader 1
```

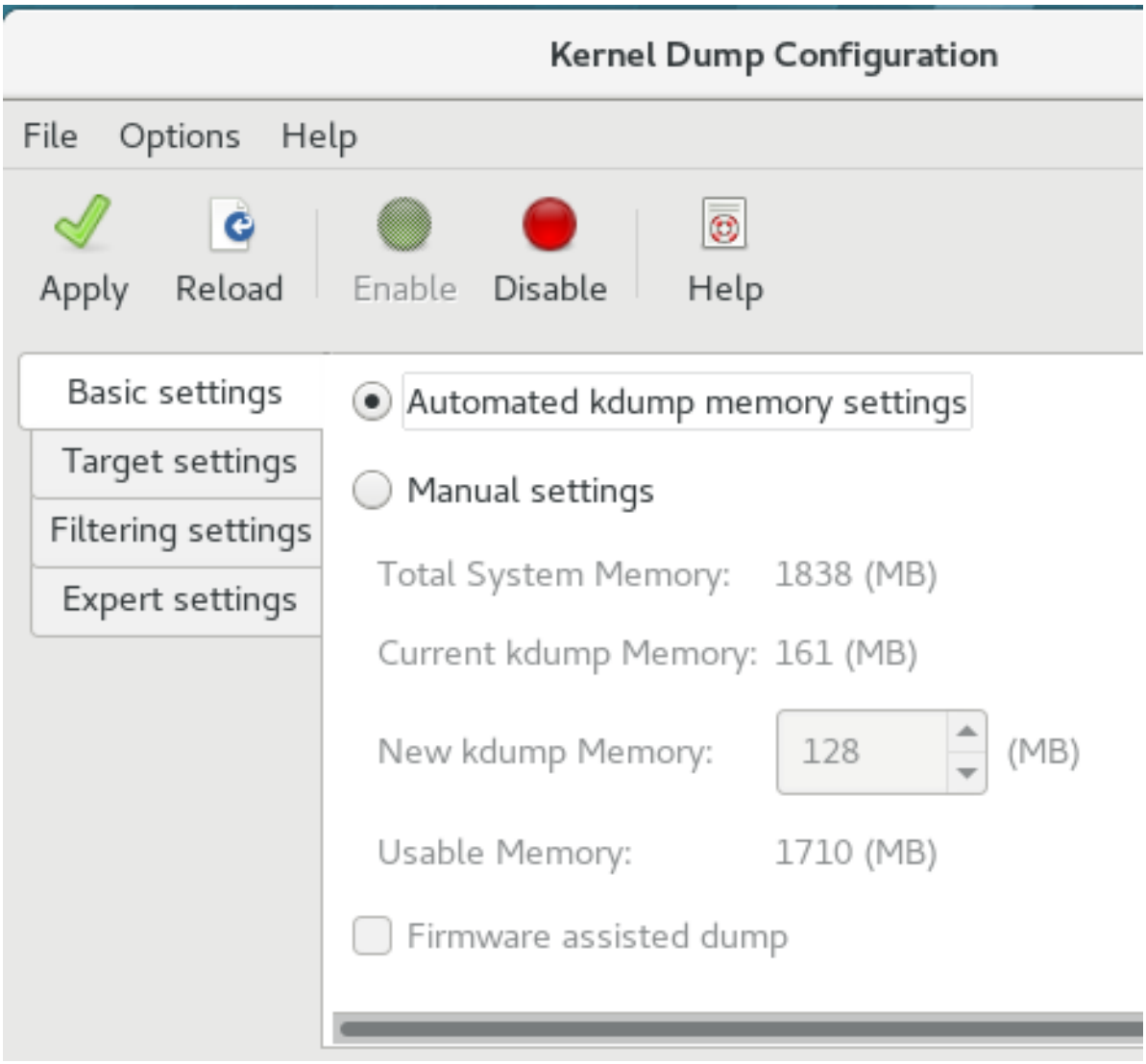
重要

s390x ハードウェア上の DASD にダンプする場合には、続行する前に **/etc/dasd.conf** でダンプサービスが正しく指定されている必要があります。

7.2.3.1. メモリー使用量の設定

基本設定 タブでは **kdump** カーネル用に予約するメモリーサイズを設定できます。**手動セッティング** のラジオボタンを選択し、**新規の kdump メモリー** フィールドの横にある上矢印ボタンまたは下矢印ボタンをクリックして予約するメモリーサイズを増減させます。システムで使用できるメモリーの残量に応じて **使用可能なメモリー** フィールドが変化します。kdump のメモリー要件の詳細は、「メモリー要件」 を参照してください。

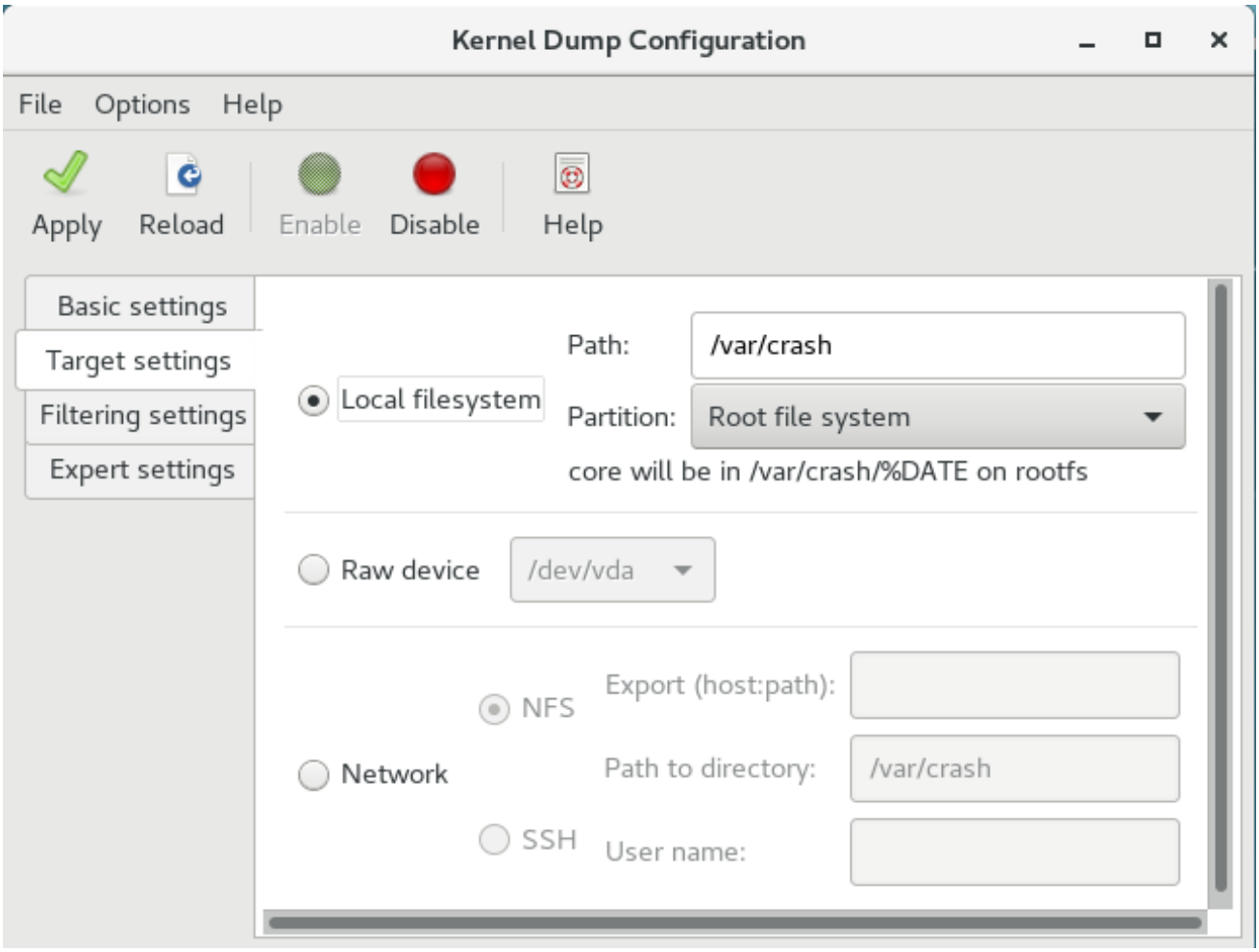
図7.1 基本設定



7.2.3.2. kdump タイプの設定

*出力先 タブでは **vmcore** ダンプの出力先を指定することができます。ダンプはローカルのファイルシステムにファイルとして保存するか、デバイスに直接書き込むか、または **NFS** (Network File System) や **SSH** (Secure Shell) などのプロトコルを使ってネットワーク経由で送信することができます。*

図7.2 出力先



*ローカルのファイルシステムにダンプを保存する場合は **ローカルファイルシステム** のラジオボタンを選択します。必要に応じて **パーティション** のドロップダウンリストから別のパーティションを選択し、**パス** フィールドで出力先ディレクトリーを選択して設定をカスタマイズすることもできます。*

重要

*Red Hat Enterprise Linux 7 では kdump の出力先として指定されているディレクトリーが **kdump** systemd サービスの起動時に存在していなければなりません。この動作は、サービスの起動時にそのディレクトリーが存在しない場合に自動的に作成されていた Red hat Enterprise Linux の以前のリリースのものとは異なります。*

*デバイスに直接ダンプを書き込む場合は **Raw デバイス** ラジオボタンを選択し、目的の出力先デバイスをその横にあるドロップダウンリストから選択します。*

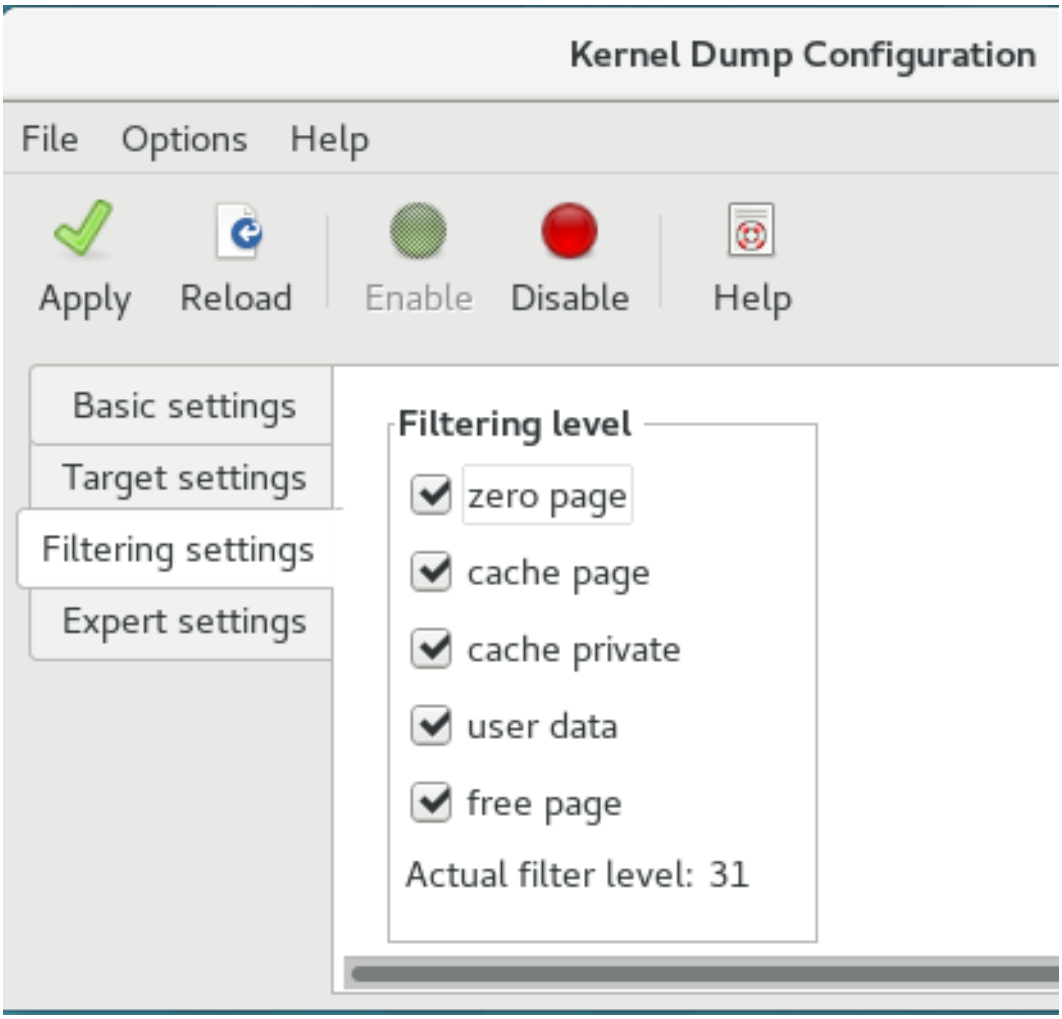
*ネットワーク接続経由で、リモートのマシンにダンプを送信する場合は **ネットワーク** ラジオボタンを選択します。**NFS** プロトコルを使用する場合は **NFS** ラジオボタンを選択して **サーバー名** と **ディレクトリーへのパス** フィールドを入力します。**SSH** プロトコルを使用する場合は **SSH** ラジオボタンを選択して **サーバー名**、**ディレクトリーへのパス**、**ユーザー名** のフィールドにリモートサーバーのアドレス、出力先ディレクトリー、有効なユーザー名をそれぞれ入力します。*

SSH サーバーの設定方法およびキーベースの認証設定については『Red Hat Enterprise Linux 7 システム管理者のガイド』を参照してください。現在対応しているターゲットの一覧は、表7.3「対応している kdump のダンプ出力先」 を参照してください。

7.2.3.3. コアコレクターの設定

Filtering Settings (フィルタリングセッティング) タブでは、**vmcore** ダンプのフィルターレベルを選択することができます。

図7.3 フィルタリング



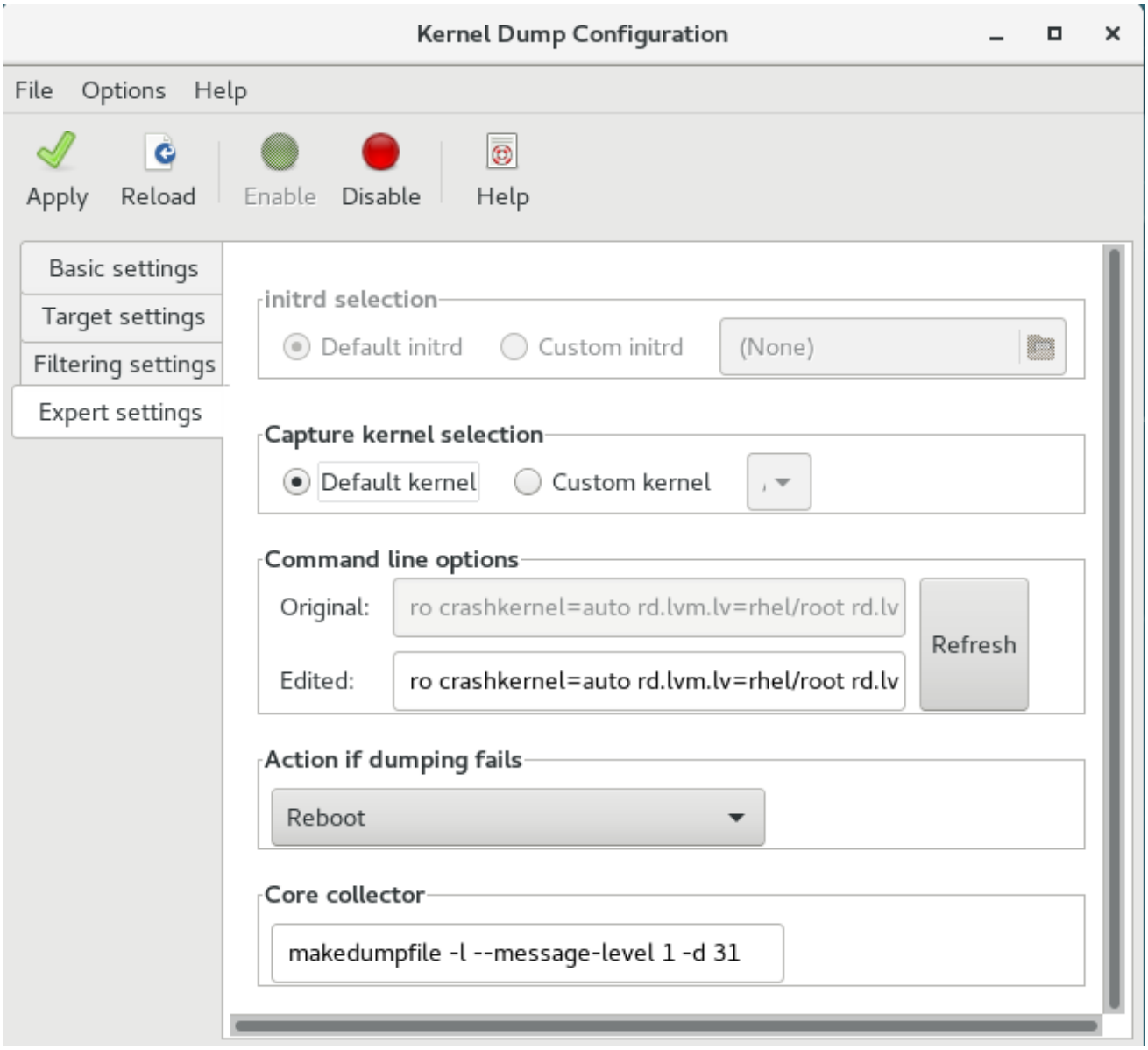
ダンプから **ゼロページ**、**キャッシュページ**、**キャッシュプライベート**、**ユーザーデータ**、または **フリーページ** を除外するには該当ラベルの横にあるチェックボックスを使って選択します。

7.2.3.4. デフォルト動作の設定

kdump がコアダンプを作成できない場合に実行する動作を選択するには、**ダンプが失敗した場合の動作** ドロップダウンリストから適切なオプションを選択します。利用可能なオプションは以下の通りです。

- **rootfs および reboot へのダンプ** を使用すると、コアをローカルに保存して、システムを再起動します。
- **Reboot** システムを再起動するデフォルトの動作
- **シェルの開始** アクティブなシェルプロンプトでユーザーを表示します。
- **halt** システムを停止します。
- **Poweroff** システムの電源を切ります。

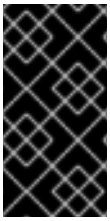
図7.4 フィルタリング



***makedumpfile コアコレクター** に渡されるオプションをカスタマイズするには **Core collector** のテキストフィールド を編集します。詳細は、「コアコレクターの設定」 を参照してください。*

7.2.3.5. サービスの有効化

*起動時に **kdump** サービスを開始する場合はツールバーの **有効化** ボタンをクリックしてから **適用** ボタンをクリックします。これにより、**multi-user.target** のサービスが有効になり、起動します。**無効化** ボタンをクリックして**適用** ボタンをクリックするとサービスが直ちに無効になります。*



重要

*Red Hat Enterprise Linux 7 では kdump の出力先として指定されているディレクトリーが **kdump** systemd サービスの起動時に存在していなければなりません。この動作は、サービスの起動時にそのディレクトリーが存在しない場合に自動的に作成されていた Red hat Enterprise Linux の以前のリリースのものとは異なります。*

systemd の出力先およびサービスの設定全般については『Red Hat Enterprise Linux 7 システム管理者のガイド』を参照してください。

7.3. KDUMP のカーネルドライバーのブラックリスト化

*カーネルドライバーのブラックリスト化により、それらの読み込みや使用が禁止されます。**/etc/sysconfig/kdump** ファイルでドライバーを追加すると、**kdump initramfs** がブラックリストに指定したモジュールを読み込まなくなります。*

*カーネルドライバーをブラックリストに登録すると、**oom** killer またはその他のクラッシュカーネルの障害を防ぐことができます。カーネルドライバーをブラックリストに指定するには、**/etc/sysconfig/kdump** ファイルの **KDUMP_COMMANDLINE_APPEND=** 変数を更新し、以下のブラックリストオプションのいずれかを指定できます。*

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

手順

1. *ブラックリストに指定するカーネルモジュールを選択します。*

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

   ***lsmod** コマンドは、現在実行中のカーネルに読み込まれているモジュールの一覧を表示します。*

2. *以下のように **/etc/sysconfig/kdump** ファイルの **KDUMP_COMMANDLINE_APPEND=** 行を更新します。*

```
KDUMP_COMMANDLINE_APPEND="rd.driver.blacklist=hv_vmbus,hv_storvsc,hv_utils,hv_netv
sc,hid-hyperv"
```

3. *また、以下のように **/etc/sysconfig/kdump** ファイルの **KDUMP_COMMANDLINE_APPEND=** 行を更新することもできます。*

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

4. *kdump サービスを再起動します。*

```
$ systemctl restart kdump
```

7.4. KDUMP 設定のテスト

警告

以下のコマンドでは、カーネルがクラッシュします。次の手順を行う場合は十分に注意してください。 実稼働のシステムでは絶対に実行しないでください。

設定をテストするため **kdump** を有効にしてシステムを再起動し、サービスが実行されているか確認します。

```
~]# systemctl is-active kdump
active
```

次に、シェルプロンプトで以下のコマンドを入力します。

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

このコマンドにより、Linux カーネルは強制的にクラッシュして **address-YYYY-MM-DD-HH:MM:SS/vmcore** ファイルが設定で選択した場所にコピーされます (デフォルトでは **/var/crash/**)。

注記

このアクションは、設定の妥当性を確認するのに加え、典型的なテストロードで実行された場合にクラッシュダンプが完了するまでの所要時間を記録するために使用できます。

7.4.1. 関連情報

7.4.1.1. インストールされているドキュメント

- ***kdump.conf***(5) - 利用できるオプションの完全なドキュメンテーションを含む **/etc/kdump.conf** 設定ファイルの man ページ。
- ***zipl.conf***(5): **/etc/zipl.conf** 設定ファイルの man ページです。
- ***zipl***(8): IBM Z 向けの **zipl** ブートローダーユーティリティーの man ページです。
- ***makedumpfile***(8) - **makedumpfile** コアコレクターの man ページ。
- ***kexec***(8) – **kexec** の man ページ。
- ***crash***(8) – **crash** ユーティリティの man ページ。
- **/usr/share/doc/kexec-tools-version/kexec-kdump-howto.txt**: **kdump** および kexec のインストールと使用方法に関する概要です。

7.4.1.2. オンラインドキュメント

https://access.redhat.com/site/solutions/6038

kexec および **kdump** 設定に関する Red Hat ナレッジベースアーティクルです。

https://access.redhat.com/site/solutions/223773

サポートしている **kdump** 出力先に関する Red Hat ナレッジベースのアーティクルです。

https://github.com/crash-utility/crash

crash ユーティリティー Git リポジトリー。

https://www.gnu.org/software/grub/

GRUB2 ブートローダーのホームページとドキュメントです。

7.5. ファームウェア支援ダンプの仕組み

7.5.1. ファームウェア支援ダンプについて

kexec および **kdump** のメカニズムは、AMD64 および Intel 64 システムでコアダンプをキャプチャーする信頼できる方法です。ただし、特に小規模システムおよびメインフレームシステムのような長い歴史を持つハードウェアでは、オンボードファームウェアを活用してメモリの領域を分離し、クラッシュ分析に重要なデータの偶発的な上書きを防ぐことができます。

本章では、利用可能なファームウェア支援ダンプの手法について紹介し、ファームウェア支援ダンプを Red Hat Enterprise Linux でどのように活用するかについて説明します。

7.5.2. IBM PowerPC ハードウェアにおける fadump の使用

ファームウェア支援ダンプ (**fadump**) は、IBM PowerPC LPARS で利用可能な **kexec-kdump** に代わる信頼性の高い仕組みです。ファームウェア支援ダンプでは、PCI および I/O デバイスが再初期化され、完全にリセットされたシステムから、vmcore がキャプチャーされます。この仕組みでは、クラッシュ発生時にファームウェアを使ってメモリーが保持されますが、**kdump** ユーザー空間スクリプトが再利用して vmcore を保存します。

そのために、**fadump** では、クラッシュ発生時にシステムファームウェアを使って保持する必要のあるメモリー領域が登録されます。これらの領域には、ブートメモリー、システムレジスター、およびハードウェアのページテーブルエントリー (PTE) を除く、すべてのシステムメモリーコンテンツが含まれます。

PowerPC 特有のハードウェアのリセット方法を含め、**fadump** の仕組みに関する詳細は、**/usr/share/doc/kexec-tools-X.y.z/fadump-howto.txt** を確認してください。ここで「X.y.z」は、お使いのシステムにインストールされている **kexec-tools** のバージョン番号を表します。

注記

ブートメモリー と呼ばれる保持されないメモリー領域は、クラッシュ後にカーネルを正常に起動するのに必要な RAM の容量です。デフォルトのブートメモリーサイズは、256 MB または全システム RAM の 5% のいずれか大きい方です。

kexec で開始されたイベントとは異なり、**fadump** プロセスでは実稼働用のカーネルを使用してクラッシュダンプを復元します。クラッシュ後の起動時に、PowerPC ハードウェアはデバイスノード **/proc/device-tree/rtas/ibm,kernel-dump** が **procfs** で利用できるようにし、fadump 対応の **kdump** スクリプトは vmcore を保存するかどうかを確認します。この処理が完了すると、システムは正しく再起動されます。

fadump の有効化

1. 「kdump のインストールと設定」 の説明に従って **kdump** をインストールし、設定します。

2. */etc/default/grub* の ***GRUB_CMDLINE_LINUX*** の行に、***fadump=on*** を追加します。

```
GRUB_CMDLINE_LINUX="rd.lvm.lv=rhel/swap crashkernel=auto rd.lvm.lv=rhel/root rhgb quiet fadump=on"
```

3. *(この操作は必須ではありません) 予約ブートメモリーサイズに、デフォルト値を使わずに具体的な値を指定する場合は、****/etc/default/grub*** の ***GRUB_CMDLINE_LINUX*** に ***crashkernel=xxM*** を追加します。 **xx** *は必要なメモリーサイズ (メガバイト単位) に指定してください。*

```
GRUB_CMDLINE_LINUX="rd.lvm.lv=rhel/swap crashkernel=xxM rd.lvm.lv=rhel/root rhgb quiet fadump=on"
```

重要

すべてのブート設定オプションと同様に、必要になる前に設定をテストすることを強く推奨します。クラッシュカーネルから起動時に Out of Memory (OOM) エラーが発生する場合は、クラッシュカーネルが正常に起動できるまで ***crashkernel=*** *で指定する値を増やします。この場合は、トライアンドエラーが必要になることがあります。*

7.5.3. IBM Z におけるファームウェア支援ダンプの手法

*IBM Z には、****Stand-alone Dump*** *および* ***VMDUMP*** *の 2 つのファームウェア支援ダンプの仕組みがあります。*

このようなシステムでは ***kdump*** *インフラストラクチャーがサポートされ、利用される Red Hat Enterprise Linux の設定については、「kdump のインストールと設定」 を参照してください。ただし、IBM Z ハードウェアが提供するこれらのファームウェア支援の手法を使用すると、いくつかのメリットが得られます。*

スタンドアロンダンプ (SADMP) メカニズムはシステムコンソールから開始および制御され、IPL 起動可能デバイス上に保管される必要があります。

VMDUMP は SADMP と類似しています。このツールもシステムコンソールから開始されますが、得られるダンプをハードウェアからコピーし、解析のためにそれをシステムに格納する仕組みがあります。

(他のハードウェアベースのダンプメカニズムと同様に) これらの手法のメリットの 1 つは、(kdump サービスが開始される前の) 起動初期段階におけるマシンの状態をキャプチャーできるという点です。

VMDUMP には、ハードウェアからコピーしたダンプファイルを Red Hat Enterprise Linux システムに格納する仕組みがありますが、IBM Z ハードウェアコンソールから、SADMP および VMDUMP の両方の設定および制御が管理されます。

IBM は、stand-alone dump program の記事で SADMP について、VMDUMP の記事で VMDUMP について詳細に説明しています。

また、IBM は、Using the Dump Tools on Red Hat Enterprise Linux 記事で Red Hat Linux Enterprise Linux 7 でダンプツールを使用するための一連のドキュメントを用意しています。

7.5.4. Fujitsu PRIMEQUEST システムにおける sadump の使用

Fujitsu の ***sadump*** *メカニズムは、****kdump*** *が正常に完了できない場合にフォールバックダンプキャプチャーが実行されるように設計されています。*

sadump *プロセスは、システムの ManageMent Board (MMB) インターフェースから手動で呼び出します。*

このシステムでは、通常通り kdump を X86_64 サーバーに対して設定し、さらに以下の追加ステップを実施して **sadump** を有効にする必要があります。

sadump に対して **kdump** が予想どおりに起動するように**/etc/sysctl.conf** で以下の行を追加または編集します。

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

上記の手順に加えて、**/etc/kdump.conf** にいくつかのオプションを追加して、sadump に対して **kdump** が正常に動作するようにする必要もあります。

特に、**kdump** の後にシステムが再起動しないようにする必要があります。**kdump** がコアの保存に失敗した後にシステムが再起動すると、**sadump** を呼び出す機会が失われます。

そのためには、**/etc/kdump.conf** の **default** アクションを halt または shell のどちらかに設定する必要があります。

```
default shell
```

重要

sadump 用にハードウェアを設定する方法は、『FUJITSU Server PRIMEQUEST 2000 Series Installation Manual』を参照してください。

7.6. コアダンプの分析

システムクラッシュの原因を確認するには、**crash** ユーティリティーを使用します。これにより、GDB (GNU Debugger) と非常によく似たインタラクティブなプロンプトを利用できます。稼働中の Linux システムだけでなく **netdump**、**diskdump**、**xendump**、**kdump** などで作成したコアダンプなども対話形式で分析することができます。

7.6.1. crash ユーティリティーのインストール

crash 分析ツールをインストールするには、**root** でシェルプロンプトから次のコマンドを実行します。

```
yum install crash
```

crash に加え、実行中のカーネルに対応する **kernel-debuginfo** パッケージもインストールしておく必要があります。このパッケージでダンプ分析に必要なデータが提供されます。**kernel-debuginfo** をインストールするには、**root** として **debuginfo-install** を使用します。

```
debuginfo-install kernel
```

Yum を使用した Red Hat Enterprise Linux の新規パッケージのインストール方法については『Red Hat Enterprise Linux 7 システム管理者のガイド』を参照してください。

7.6.2. crash ユーティリティーの実行

シェルプロンプトで次の形式のコマンドを入力してユーティリティーを起動します。

```
crash /usr/lib/debug/lib/modules/<kernel>/vmlinux \ /var/crash/<timestamp>/vmcore
```

kdump で取得したのと同じ <kernel> のバージョンを使用します。現在実行中のカーネルを確認するには、**uname -r** のコマンドを使用します。

例7.1 crash ユーティリティーの実行

```
~]# crash /usr/lib/debug/lib/modules/2.6.32-69.el6.i686/vmlinux \
/var/crash/127.0.0.1-2010-08-25-08:45:02/vmcore

crash 5.0.0-23.el6
Copyright (C) 2002-2010  Red Hat, Inc.
Copyright (C) 2004, 2005, 2006  IBM Corporation
Copyright (C) 1999-2006  Hewlett-Packard Co
Copyright (C) 2005, 2006  Fujitsu Limited
Copyright (C) 2006, 2007  VA Linux Systems Japan K.K.
Copyright (C) 2005  NEC Corporation
Copyright (C) 1999, 2002, 2007  Silicon Graphics, Inc.
Copyright (C) 1999, 2000, 2001, 2002  Mission Critical Linux, Inc.
This program is free software, covered by the GNU General Public License,
and you are welcome to change it and/or distribute copies of it under
certain conditions.  Enter "help copying" to see the conditions.
This program has absolutely no warranty.  Enter "help warranty" for details.

GNU gdb (GDB) 7.0
Copyright (C) 2009 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-pc-linux-gnu"...

      KERNEL: /usr/lib/debug/lib/modules/2.6.32-69.el6.i686/vmlinux
    DUMPFILE: /var/crash/127.0.0.1-2010-08-25-08:45:02/vmcore  [PARTIAL DUMP]
        CPUS: 4
        DATE: Wed Aug 25 08:44:47 2010
      UPTIME: 00:09:02
LOAD AVERAGE: 0.00, 0.01, 0.00
       TASKS: 140
    NODENAME: hp-dl320g5-02.lab.bos.redhat.com
     RELEASE: 2.6.32-69.el6.i686
     VERSION: #1 SMP Tue Aug 24 10:31:45 EDT 2010
     MACHINE: i686  (2394 Mhz)
      MEMORY: 8 GB
       PANIC: "Oops: 0002 [#1] SMP " (check log for details)
         PID: 5591
     COMMAND: "bash"
        TASK: f196d560  [THREAD_INFO: ef4da000]
         CPU: 2
       STATE: TASK_RUNNING (PANIC)

crash>
```

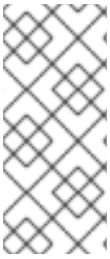
7.6.3. メッセージバッファーの表示

対話式プロンプトで **log** コマンドを入力して、カーネルメッセージバッファーを表示します。

例7.2 カーネルメッセージバッファーの表示

```
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
```

コマンドの使用の詳細は **help log** を実行します。



注記

カーネルメッセージバッファーには、システムクラッシュに関する最も重要な情報が含まれています。したがって、これは常に最初に **vmcore-dmesg.txt** ファイルにダンプされます。これは、たとえば、ターゲットの場所にスペースがないために、 **vmcore** ファイル全体の取得試行に失敗するときに便利です。デフォルトでは、**vmcore-dmesg.txt** は **/var/pkcs/directory/** あります。

7.6.4. バックトレースの表示

対話式プロンプトで **bt** コマンドを入力してカーネルのスタックトレースを表示します。1 プロセスのバックトレースを表示するには **bt <pid>** と入力します。

例7.3 カーネルスタックトレースの表示

```
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
```

*このコマンドの使用方法についての詳しい情報を参照するには、**help bt** と入力してください。*

7.6.5. プロセスの状態表示

*対話式プロンプトで **ps** コマンドを入力してシステム内のプロセスの状態を表示します。1 プロセスの状態を表示するには **ps <pid>** と入力します。*

例7.4 システム内のプロセスの状態表示

```
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
```

*このコマンドの使用方法についての詳しい情報を参照するには、**help ps** と入力してください。*

7.6.6. 仮想メモリー情報の表示

*基本的な仮想メモリー情報を表示するには、対話式プロンプトで **vm** コマンドを入力します。1 プロセスの情報を表示するには **vm <pid>** と入力します。*

例7.5 現在のコンテキストの仮想メモリー情報の表示

```
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 ...
```

*このコマンドの使用方法についての詳しい情報を参照するには、**help vm** と入力してください。*

7.6.7. オープンファイルの表示

*対話式プロンプトで **files** コマンドを入力してオープンファイルに関する情報を表示します。選択した1プロセスだけで開かれているファイルを表示するには **files <pid>** と入力します。*

例7.6 現在のコンテキストのオープンファイルについての情報の表示

```
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
```

*このコマンドの使用方法についての詳しい情報を参照するには、**help files** と入力してください。*

7.6.8. ユーティリティーの終了

*対話式プロンプトを終了して、**crash** を終了するには、**exit** または **q** と入力します。*

例7.7 crash ユーティリティーの終了

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

7.7. よくある質問

クラスター化された環境で kdump を使用する場合には、何に注意したら良いですか?

「RHEL 6 および7 の High Availability アドオンで使用できるように kdump を設定する 」の記事で、High Availability アドオンを使用しているシステム管理者が利用可能なオプションを説明しています。

起動初期に kdump が失敗します。 どのようにしてブートログをキャプチャーするのですか?

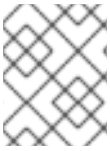
2 番目のカーネルの起動に問題がある場合は、起動初期のブートログを確認する必要があります。 問題のあるマシンに対するシリアルコンソールを有効にすることで、このログを取得することができます。

「RHEL7 でシリアルターミナルを設定する」の記事で、起動初期のブートメッセージへアクセスするために必要な設定が説明されています。

デバッグ用に、どのようにして makedumpfile からのメッセージを増やすのですか?

makedumpfile が失敗する時には、何が悪いのかを理解するためにログのレベルを増やさなければならない場合があります。これはダンプレベルを設定するのとは異なる操作です。 ログのレベルについては、**/etc/kdump.conf** の **core_collector** 行の内容を編集して、**makedumpfile** に対する **message_level** オプションを増やします。

デフォルトで **makedumpfile** はレベル 1 に設定されており、これは、出力を進捗インジケーターに制限します。このメッセージレベルを 31 に設定すると、すべてのデバッグ情報を有効にできます。メッセージレベル 31 では、進捗インジケーター、共通メッセージ、エラーメッセージ、デバッグメッセージ、およびレポートメッセージの詳細を出力します。



注記

メッセージレベルのオプションの詳細は、**makedumpfile (8)** man ページを参照してください。

core_collector 設定の行は、設定時には以下のようになっているはずです。

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

どのようにして Dracut をデバッグするのですか?

dracut が initramfs の構築に失敗することがあります。その場合には、問題を切り分けるために **dracut** のログレベルを増やす必要があります。

/etc/kdump.conf を編集して **dracut_args** 行を変更し、必要なすべての dracut 引数と共にオプション **-L 5** を含めます。

dracut_args で他のオプションを設定していない場合は、次のような結果になるはずです。

```
dracut_args -L 5
```

仮想マシンで利用可能なダンプの手法は何ですか?

多くの場合、クラッシュやパニックの後にマシンからメモリーダンプを取得するには **kdump** の仕組みがあれば十分です。これは、ベアメタルのインストールと同じように設定することができます。

ただし、場合によっては、ハイパーバイザーと直接連携してクラッシュダンプを取得する必要がある場合があります。ハイパーバイザーを使用したクラッシュダンプの取得方法として **libvirt** には **pvpanic** と **virsh dump** という 2 つの仕組みがあります。これらの手法は 『仮想化の導入および管理ガイド』に記載されています。

pvpanic の仕組みについては、『仮想化の導入および管理ガイド』の「パニックデバイスの設定」に記載されています。

virsh dump コマンドについては、『仮想化の導入および管理ガイド』の「ドメインのコアのダンプファイルの作成」で説明されています。

Red Hat サポートサービスに大きなサイズのダンプをアップロードする場合はどうしたらいいですか？

分析のため、Red Hat グローバルサポートサービスにカーネルクラッシュのダンプファイルを送信していただく必要がある場合があります。ただし、ダンプファイルのサイズはフィルターで特定の情報に絞り込んだ場合でも非常に大きくなる可能性があります。新しいサポートケースの起票時に、250 MB を超えるファイルは Red Hat カスタマーポータルからは直接アップロードできないため、Red Hat では大きなサイズのファイルのアップロード用に FTP サーバーを用意しています。

FTP サーバーのアドレスは **dropbox.redhat.com** で、ファイルは **/incoming/** ディレクトリーにアップロードしてください。ご使用の FTP クライアントを passive モードに設定しておく必要があります。ご使用のファイアウォールでこのモードを使用できない場合は **origin-dropbox.redhat.com** サーバーを active モードでご利用ください。

アップロードするファイルは必ず **gzip** などで圧縮し、ファイル名にはわかりやすい名前を付けてください。ファイル名にサポートケース番号を使用されることをお薦めします。必要なファイルをすべてアップロードしたらサポートケース担当のエンジニアへ正確なファイル名とその SHA1 または MD5 チェックサムをお知らせください。

詳細な説明については「Ret Hat サポートチームにファイル (vmcore、rhev logcollector、sosreport、ヒープダンプ、ログファイルなど) を送付する」を参照ください。

クラッシュダンプが完了するまでに、どれくらい時間がかかりますか?

この情報は、障害復旧計画の目的で、ダンプ完了の所要時間を把握するために、多くの場合に必要となります。ただし、この時間の長さは、ディスクにコピーされるメモリー量と、RAM とストレージ間のインターフェースの速度に大きく左右されます。

テストがどのようなタイミングで行われても、システムは典型的な負荷をかけた状態で稼働させておく必要があります。そうでないと、除外されるページによって、完全に負荷がかかった実稼働システムにおける kdump の動作で誤った見解が提示される可能性があります。特にこの違いは、大量のメモリーが使用されている状況でより顕著に見られます。

ダンプの所要時間を評価する場合に、ストレージインターフェースもプランニング時に考慮する必要があります。ネットワーク制約事項が原因で、ローカルに接続された SATA ディスクと比べると、**ssh** などを使用した接続ダンプは完了までに長く時間がかかる可能性があります。

インストール時に Kdump をどのように設定するのですか?

キックスタートまたは対話型の GUI を使用すれば、わずかなオプションを指定するだけでインストール時に **kdump** を設定することができます。

anaconda インストール GUI を使用した **kdump** の設定については、『インストールガイド』の「Kdump」セクションに詳しい説明が記載されています。

kickstart 構文は以下の通りです。

```
%addon com_redhat_kdump [--disable,enable] [--reserve-mb=[auto,value]]
%end
```

キックスタートに対するこのアドオンにより、kdump 機能の無効化/有効化や、オプションとして予約メモリーサイズの定義 (自動のデフォルトオプションを明示的に呼び出す、またはメガバイト単位で数値を指定する) が可能になります。 なお、スイッチ全体が省略された場合も、デフォルトオプションが呼び出されます。

キックスタートを使用してシステムのデプロイメントを自動化する方法の詳細は、『インストールガイド』の「キックスタートを使ったインストール」を参照してください。

キックスタートアドオン構文の詳細については、『インストールガイド』の「キックスタート構文の参考資料」を参照してください。

7.8. サポートしている KDUMP の設定とダンプ出力先

7.8.1. kdump メモリー要件

kdump でカーネルクラッシュダンプをキャプチャーしてさらに分析ができるように保存するにはシステムメモリーの一部をキャプチャーカーネル用に永続的に予約する必要があります。

コマンドラインでメモリー設定を変更する方法は、「メモリー使用量の設定」 を参照してください。グラフィカルユーザーインターフェースで予約メモリーの設定を変更する方法については、「メモリー使用量の設定」 を参照してください。

表 7.1 では、kernel および kdump パッケージで kdump が自動的に予約されているメモリーをリストします。kdump は自動的に、CPU アーキテクチャーに基づくメモリーと、利用可能な物理メモリーの合計を確保します。

表7.1 kdump によって自動的に予約されているクラッシュメモリーの合計

CPU アーキテクチャー	使用可能なメモリー	クラッシュメモリーは自動的に予約されています
AMD64 および Intel 64 (x86_64)	2 GB 以上	161 MB + 64 MB (1 TB あたり)
64 ビット ARM アーキテクチャー (arm64)	2 GB 以上	512 MB
IBM POWER ppc64/ppc64le	2 GB から 4 GB	384 MB
	4 GB から 16 GB	512 MB
	16 GB から 64 GB	1 GB
	64 GB から 128 GB	2 GB
	128 GB 以上	4 GB

CPU アーキテクチャー	使用可能なメモリー	クラッシュメモリーは自動的に予約されています
IBM Z (s390x)	4 GB 以上	161 MB + 64 MB (1 TB あたり) [1] 160 MB (RHEL-ALT-7.6)

さまざまな Red Hat Enterprise Linux テクノロジーの機能や制限に関する詳しい情報は、https://access.redhat.com/articles/rhel-limits を参照してください。

7.8.2. メモリー自動予約の最小しきい値

*一部のシステムでは、ブートローダーの設定ファイルで **crashkernel=auto** パラメーターを使用するか、グラフィカル設定ユーティリティーで自動割り当ての設定を有効にすると、kdump 用のメモリーを自動的に割り当てることができます。ただし、この自動予約が機能するには、合計メモリーの特定量のメモリーを利用できる必要があります。この容量はシステムのアーキテクチャーによって異なります。*

*次の表は、**kernel** および **kernel-alt** パッケージの自動メモリー割り当てのしきい値の一覧です。システムのメモリーが以下に示すしきい値を下回る場合は手動でメモリー予約を行う必要があります。*

コマンドラインで設定を変更する方法は、「メモリー使用量の設定」を参照してください。グラフィカルユーザーインターフェースで予約メモリーのサイズを変更する方法は、「メモリー使用量の設定」を参照してください。

表7.2 自動メモリー予約に必要な最小メモリーサイズ

アーキテクチャー	必要なメモリー
AMD64 と Intel 64 (**x86_64**)	2 GB
IBM POWER (**ppc64**)	2 GB
IBM Z (**s390x**)	4 GB
64 ビット ARM アーキテクチャー (**arm64**)	2 GB

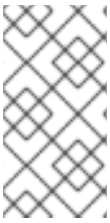
7.8.3. サポートしている kdump のダンプ出力先

カーネルクラッシュをキャプチャーする際、コアダンプを直接デバイスに書き込んでローカルファイルシステムにファイルとして保存するか、またはネットワーク経由で送信することができます。現在サポートしているダンプ出力先および kdump による非サポートが明確なダンプ出力先の全一覧を以下に示します。

コマンドラインでターゲットタイプを設定する方法は、「kdump タイプの設定」を参照してください。グラフィカルユーザーインターフェースでこの作業を行う場合は、「kdump タイプの設定」を参照してください。

表7.3 対応している kdump のダンプ出力先

Type	対応しているダンプ出力先	対応していないダンプ出力先
Raw デバイス	ローカルで添付されたすべての raw ディスクとパーティション	
ローカルファイルシステム	直接接続されているディスクドライブ、ハードウェア RAID 論理ドライブ、LVM デバイス、**mdraid** アレイ上の **ext2**、**ext 3**、**ext4**、および **xfs** ファイルシステム。	**auto** タイプ (自動ファイルシステム検出) など、この表で明示的にサポート対象とされていないローカルファイルシステム。
リモートディレクトリー	**IPv4** で **NFS** または **SSH** プロトコルを使用してアクセスしたリモートディレクトリー。	**NFS** プロトコルを使用してアクセスした **rootfs** ファイルシステム上のリモートディレクトリー。
FCoE (Fibre Channel over Ethernet) プロトコルを使用してアクセスするリモートディレクトリー。	**qla2xxx**、**lpfc**、**bfa** ハードウェア **FCoE** ターゲット。**bnx2fc** および **ixgbe** ソフトウェア **FCoE** ターゲット。	
ハードウェアおよびソフトウェアイニシエーター上で **iSCSI** プロトコルを使用してアクセスするリモートディレクトリー。	**be2iscsi** ハードウェア上で **iSCSI** プロトコルを使用してアクセスするリモートディレクトリー。	マルチパスベースのストレージ
		IPv6 上でアクセスするリモートディレクトリー
		SMB または **CIFS** を使ってアクセスするリモートディレクトリー。
		ワイヤレスネットワークインターフェースを使ってアクセスするリモートディレクトリー



注記

*ソフトウェア **FCoE** のダンプ出力先へダンプすると、OOM (Out of Memory) エラーが発生します。この場合は、デフォルトの **crashkernel=auto** パラメーターの値を大きくします。このカーネルブートパラメーターを設定する方法は、「メモリー使用量の設定」を参照してください。*

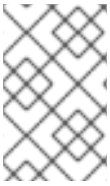
7.8.4. 対応している kdump のフィルターレベル

*ダンプファイルのサイズを縮小させるため kdump では **makedumpfile** コアコレクターを使ってデータを圧縮して必要に応じて関連性のない情報を除外します。以下の表に、**makedumpfile** ユーティリティーで現在対応しているフィルターレベルの完全な一覧を示します。*

コマンドラインでコアコレクターを設定する方法は、「コアコレクターの設定」を参照してください。グラフィカルユーザーインターフェースでこの作業を行う場合は、「コアコレクターの設定」を参照してください。

表7.4 サポートしているフィルターレベル

オプション	説明
1	ゼロページ
2	キャッシュページ
4	キャッシュプライベート
8	ユーザーページ
16	フリーページ



注記

***makedumpfile** コマンドは、Red Hat Enterprise Linux 7.3 以降の透過的なヒュージページおよびhugetlbfs ページの削除をサポートします。これらのタイプのhugepages User Page の両方を考えて、**-8** レベルを使用して削除します。*

7.8.5. サポートしているデフォルトの動作

kdump がコアダンプの作成に失敗すると、デフォルトでは、オペレーティングシステムが再起動します。第1 ダンプ出力先へのコアダンプの保存に失敗した場合、kdump に別の動作を行うよう設定することができます。kdump で現在サポートしているデフォルト動作を以下に示します。

コマンドラインでデフォルト動作を設定する方法については、「デフォルト動作の設定」を参照してください。グラフィカルユーザーインターフェースでこの作業を行う場合は、「デフォルト動作の設定」を参照してください。

表7.5 サポートしているデフォルトの動作

オプション	説明
dump_to_rootfs	root ファイルシステムにコアダンプの保存を試行します。ネットワーク上のダンプ出力先と併用する場合に特に便利なオプションです。ネットワーク上のダンプ出力先にアクセスできない場合、ローカルにコアダンプを保存するよう kdump の設定を行います。システムは、後で再起動します。
reboot	システムを再起動します。コアダンプは失われます。
halt	システムを停止します。コアダンプは失われます。

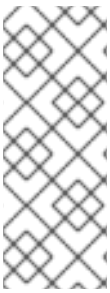
オプション	説明
poweroff	システムの電源を切ります。コアダンプは失われます。
shell	initramfs 内から shell セッションを実行して、ユーザーが手動でコアダンプを記録できるようにします。

7.8.6. kdump サイズの見積もり

***kdump** 環境のプランニングや構築の際に、ダンプファイルに必要な容量がどれくらいか把握してから作成する必要があります。これには、**makedumpfile** コマンドを使用すると役立ちます。*

*以下のように、**--mem-usage** 機能を使用して、ダンプファイルに必要な領域を見積もります。*

 # makedumpfile -f --mem-usage /proc/kcore



注記

***-f オプションを使用した --mem-usage** 機能は、カーネルバージョン v4.11 以降で動作します。*

*v4.11 よりも前のバージョンのカーネルでは、**--mem-usage** にオプション **-f** を指定して実行する前に、カーネルにアップストリームのコミット 464920104bf7 でパッチが当てられていることを確認します。*

***--mem-usage** オプションでは、除外可能なページに関する有用なレポートが提供され、これを使用して、割り当てるダンプレベルを判断することができます。このコマンドは、システムに典型的な負荷をかけた状態で実行する必要があります。 そうでないと、**makedumpfile** は実稼動環境で必要とされる値よりも小さい値を返します。*

```
[root@hostname ~]# makedumpfile -f --mem-usage /proc/kcore

TYPE            PAGES            EXCLUDABLE    DESCRIPTION
----------------------------------------------------------------------
ZERO            501635           yes           Pages filled with zero
CACHE           51657            yes           Cache pages
CACHE_PRIVATE   5442             yes           Cache pages + private
USER            16301            yes           User process pages
FREE            77738211         yes           Free pages
KERN_DATA       1333192          no            Dumpable kernel data
```

重要

***makedumpfile** コマンドは **pages** にレポートを出します。つまり、カーネルページサイズ (Red Hat Enterprise Linux カーネルの場合、AMD64 および Intel 64 のアーキテクチャーでは 4 キロバイト、IBM POWER アーキテクチャーの場合は 64 キロバイト) に対して使用中のメモリーのサイズを計算する必要があります。*

7.8.7. kernel および kernel-alt パッケージでのアーキテクチャーのサポート

*以下の表は、**kernel** および **kernel-alt** パッケージでのアーキテクチャーおよび利用可能なメモリーサポートの概要を示しています。*

表7.6 kernel および kernel-alt パッケージでサポートされるアーキテクチャー

CPU アーキテクチャー	使用可能なメモリー	RHEL 7.5 以前	RHEL 7.6 以降	RHEL-ALT-7.4	RHEL-ALT-7.5 以降
AMD64 および Intel 64 (x86_64)	2GB 以上	161MB + 64MB (1TB あたり)	161MB + 64MB/1TB	2GB から 160MB	2GB から 160MB
64 ビット ARM アーキテクチャー (arm64)	2GB 以上	該当なし	該当なし	2GB から 512MB	2GB から 512MB
IBM POWER ppc64/ppc64le (Upto POWER8)	2GB から 4GB	384MB	384MB	該当なし	該当なし
	4GB から 16GB	512MB	512MB	該当なし	該当なし
	16GB から 64GB	1GB	1GB	該当なし	該当なし
	64GB から 128GB	2GB	2GB	該当なし	該当なし
	128GB 以上	4GB	4GB	該当なし	該当なし
IBM POWER ppc64/ppc64le (Upto POWER9)	2GB から 4GB	384MB	384MB	384MB	384MB
	4GB から 16GB	512MB	512MB	512MB	512MB
	16GB から 64GB	1GB	1GB	1GB	1GB
	64GB から 128GB	2GB	2GB	2GB	2GB
	128GB 以上	4GB	4GB	4GB	4GB
IBM POWER ppc64le (POWER9)	2GB から 4GB	該当なし	該当なし	384MB	384MB
	4GB から 16GB	該当なし	該当なし	512MB	512MB

CPU アーキテクチャー	使用可能なメモリー	RHEL 7.5 以前	RHEL 7.6 以降	RHEL-ALT-7.4	RHEL-ALT-7.5 以降
	16GB から 64GB	該当なし	該当なし	1GB	1GB
	64GB から 128GB	該当なし	該当なし	2GB	2GB
	128GB 以上	該当なし	該当なし	4GB	4GB
IBM Z (s390x)	4GB 以上	161MB + 64MB (1TB あたり)	161MB + 64MB (1TB あたり)	161MB + 64MB (1TB あたり)	160MB

7.9. KEXEC を使用したカーネルの再起動

7.9.1. kexec によるカーネルの再起動

kexec *システムコールを使用すると現在実行中のカーネルから別のカーネルを読み込んだり、起動したりすることが可能で、カーネル内のブートローダーとして機能します。*

kexec *ユーティリティーは、* ***kexec*** *システムコールのカーネルおよび* ***initramfs*** *イメージを読み込み、別のカーネルで起動します。*

次のセクションでは、kexec ユーティリティーを使用して別のカーネルで再起動する際に ***kexec*** *システムコールを手動で呼び出す方法を説明します。*

1. ***kexec*** *ユーティリティーを実行します。*

```
# kexec -l /boot/vmlinuz-3.10.0-1040.el7.x86_64 --initrd=/boot/initramfs-3.10.0-1040.el7.x86_64.img --reuse-cmdline
```

このコマンドは、 ***kexec*** *システムコールのカーネルおよび initramfs イメージを手動で読み込みます。*

2. *システムを再起動します。*

```
# reboot
```

このコマンドはカーネルを検出し、すべてのサービスをシャットダウンしてから、 ***kexec*** *システムコールを呼び出して直前の手順で指定したカーネルに再起動します。*

警告

kexec -3 コマンドを使用してカーネルを再起動すると、システムは、次のカーネルを起動する前に標準のシャットダウンシーケンスを通過しません。そのため、データ損失や応答しないシステムが発生する可能性があります。

7.10. KDUMP に関連する PORTAL LABS

Portal Labs は簡単な Web アプリケーションで、システム管理者がさまざまなシステムタスクを実施するのに役立ちます。現在、Kdump に焦点を当てているラボが 2 つあります。Kdump Helper および Kernel Oops Analyzer

7.10.1. Kdump ヘルパー

Kdump Helper は、一連の質問および操作で構成され、**kdump** 設定ファイルの準備を支援します。

Lab のワークフローには、クラスター化された環境およびスタンドアロン環境の両方に関する手順が含まれています。

7.10.2. Kernel Oops Analyzer

Kernel Oops Analyzer ツールを使うと、Oops メッセージを処理して、クラッシュダンプスタックを解析することなく既知のソリューションがあるかどうかを確認することができます。

Kernel Oops Analyzer では **makedumpfile** からの情報を使用し、クラッシュしたマシンの Oops メッセージとナレッジベースに蓄積された既知の問題を比較します。これにより、予期せぬ障害が発生しても、システム管理者はすばやく既知の問題の可能性を除外して、詳細な解析を依頼するためにサポートチケットを発行することができます。

第8章 カーネル整合性サブシステムによるセキュリティーの強化

8.1. カーネル整合性サブシステム

整合性サブシステムは、システムのデータの整合性を維持するカーネルの一部を構成します。このサブシステムは、ユーザーが特定のシステムファイルに望ましくない変更を行うことを防ぎ、システムを初期状態に維持するのに役立ちます。

カーネル整合性サブシステムは、2 つの主要なコンポーネントで構成されています。

Integrity Measurement Architecture (IMA)

- ファイルのコンテンツの測定を、それが実行され、開かれるたびに実行します。ユーザーは、カスタムポリシーを適用してこの動作を変更できます。
- カーネルのメモリー領域内に測定値を配置すると、システムのユーザーが変更できなくなります。
- ローカルおよびリモートのユーザーが測定値を検証できるようにします。

Extended Verification Module (EVM)

- IMA 測定や SELinux 属性など、システムのセキュリティーに関連するファイルの拡張属性 (**xattr** としても知られている) を保護するには、対応する値を暗号でハッシュ化します。

IMA と EVM のどちらにも、追加機能を備えた多くの機能拡張が含まれます。以下に例を示します。

IMA-Appraisal

- 現在のファイルの内容を、カーネルメモリー内の測定ファイルに先に保存した値に対してローカルで検証します。現在の測定が以前の測定に一致しない場合、この拡張機能は特定のファイルで操作が実行されることを禁止します。

EVM デジタル署名

- カーネルのキーリングに保存されている暗号鍵を使用して、デジタル署名を使用できるようにします。EVM デジタル署名は、コンテンツハッシュ (**security.ima**) が含まれるファイルの xattr 値の送信元および整合性を確認します。

注記

機能拡張は相互に補完しますが、それぞれを独立して設定し、使用することができます。

カーネル整合性サブシステムは、TPM (Trusted Platform Module) を使用して、システムセキュリティーをさらに強化できます。TPM は、重要な暗号化機能についての TNC (Trusted Computing Group) による仕様です。TPM は通常、プラットフォームのマザーボードに接続される専用ハードウェアとして実装され、ハードウェアチップの保護された改ざん防止領域から暗号化機能を提供することで、ソフトウェアベースの攻撃を防ぎます。TPM 機能の一部は次のとおりです。

- 乱数ジェネレーター
- 暗号化キーのジェネレーターと安全なストレージ

- ハッシュジェネレーター
- リモート認証

8.2. INTEGRITY MEASUREMENT ARCHITECTURE

Integrity Measurement Architecture (IMA) は、カーネル整合性サブシステムに属するコンポーネントです。IMA は、アクセス前にファイルのハッシュを測定し、保存し、評価することにより、ローカルファイルのコンテンツを維持することを目的としています。これにより、信頼性のないデータの読み取りおよび実行を防ぐことができ、システムのセキュリティーが強化されます。

8.3. EXTENDED VERIFICATION MODULE

Extended Verification Module (EVM) は、ファイルの拡張属性 (xattr) の変更を監視するカーネル整合性サブシステムに属するコンポーネントです。Integrity Measurement Architecture (IMA) を含む多くのセキュリティー指向のテクノロジーは、コンテンツハッシュなどの機密ファイル情報を拡張属性に保存します。EVM は、この拡張属性と特別な鍵から別のハッシュを作成します。これはシステムの起動時に読み込まれます。結果のハッシュは、拡張属性が使用されるたびに検証されます。たとえば、IMA がファイルを評価する場合です。

8.4. 信頼できる鍵および暗号化された鍵

信頼できる 鍵および **暗号化された鍵** は、カーネルキーリングサービスによって使用されるカーネルが生成する可変長の対称鍵です。このタイプのキーは、暗号化されていない形式でユーザー空間に表示されないため、整合性を検証できます。したがって、それらは Extended Verification Module (EVM) により、実行中のシステムの整合性を検証し、確認するために使用できます。ユーザーレベルのプログラムがアクセス可能なのは、暗号化された **ブロブ** の形式での鍵のみです。

信頼できる鍵は、Trusted Platform Module (TPM) チップというハードウェアが必要になります。これは、鍵を作成し、暗号化 (保護) するために使用されます。TPM は **storage root key** (SRK) という 2048 ビットの RSA 鍵を使用して鍵を保護します。

信頼でき、暗号化された鍵についての詳細は、『RHEL 7 セキュリティーガイド』の「Trusted and Encrypted Keys」セクションを参照してください。

8.5. INTEGRITY MEASUREMENT ARCHITECTURE および EXTENDED VERIFICATION MODULE の有効化

Integrity Measurement Architecture (IMA) および Extended Verification Module (EVM) は、さまざまな方法でシステムセキュリティーを強化するカーネル整合性サブシステムのコンポーネントです。IMA および EVM を設定すると、ファイルに署名できるため、システムのセキュリティーを強化できます。

前提条件

- ***ima-evm-utils*** パッケージおよび ***keyutils*** パッケージがシステムにインストールされている。
- ***securityfs*** ファイルシステムが ***/sys/kernel/security/*** ディレクトリーにマウントされます。
- ***/sys/kernel/security/ima/*** ディレクトリーが存在する。

```
# mount
…
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
```

```
...
```

手順

1. IMA および EVM を有効にするためにシステムを準備します。

   a. 以下のカーネルコマンドラインパラメーターを追加します。

   ```
   # grubby --update-kernel=/boot/vmlinuz-$(uname -r) --args="ima_appraise=fix
   ima_appraise_tcb evm=fix"
   ```

   このコマンドは、現在のブートエントリーの **fix** モードで IMA および EVM を有効にし、ユーザーが IMA 測定を収集し、更新できるようにします。

   ima_appraise_tcb カーネルコマンドラインパラメーターにより、カーネルは、デフォルトの TCB (Trusted Computing Base) 測定ポリシーと評価手順を使用するようになります。評価手順では、以前および現在の測定が一致しないファイルへのアクセスを禁止します。

   b. 変更を有効にするために再起動します。

   c. オプションで、カーネルコマンドラインに新しいパラメーターが含まれることを確認します。

   ```
   # cat /proc/cmdline
   BOOT_IMAGE=/vmlinuz-3.10.0-1136.el7.x86_64 root=/dev/mapper/rhel-root ro
   crashkernel=auto spectre_v2=retpoline rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb quiet
   LANG=en_US.UTF-8 ima_appraise=fix ima_appraise_tcb evm=fix
   ```

2. EVM の公開鍵および非公開鍵のペアを作成し、設定します。

   a. EVM の新規のキーリングを作成します。

   ```
   # evm_kr_id=$(keyctl newring _evm @u)
   ```

   このコマンドは **_evm** キーリングを作成し、これを **@u** システムユーザーキーリングに割り当てます。次に、**_evm** のキーリング ID が **evm_kr_id** 変数に割り当てられ、これにより次回の処理がより容易になります。

   b. オプションで、新たに作成されたキーリングを表示します。

   ```
   # keyctl show
   Session Keyring
   1025767139 --alswrv   0     0  keyring: _ses
   548660789 --alswrv   0 65534   \_ keyring: _uid.0
   456142548 --alswrv   0     0       \_ keyring: _evm
   ```

   c. キーのディレクトリーを作成します。

   ```
   # mkdir -p /etc/keys/
   ```

   d. 1024 ビットの RSA 秘密鍵を **/etc/keys/privkey.pem** ファイルに生成します。

   ```
   # openssl genrsa -out /etc/keys/privkey.pem 1024
   Generating RSA private key, 1024 bit long modulus
   ```

```
.......++++++
...++++++
e is 65537 (0x10001)
```

e. *以前に作成した **/etc/keys/privkey.pem** 秘密鍵を使用して、対応する RSA 公開鍵を **/etc/keys/pubkey.pem** ファイルに派生させます。*

```
# openssl rsa -pubout -in /etc/keys/privkey.pem -out /etc/keys/pubkey.pem
writing RSA key
```

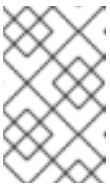
f. *公開鍵を専用の EVM キーリングにインポートします。*

```
# evmctl import --rsa /etc/keys/pubkey.pem $evm_kr_id
1054989579
```

*このコマンドは、**/etc/keys/pubkey.pem** 公開鍵を **_evm** キーリングにインポートします。次に、**_evm** キーリングがカーネルキーリングに割り当てられます。鍵のシリアル番号は、先の例の 2 行目にあります。*

g. *オプションで、新たにインポートされた鍵を表示します。*

```
# keyctl show
Session Keyring
1025767139 --alswrv   0     0  keyring: _ses
548660789 --alswrv   0 65534   \_ keyring: _uid.0
456142548 --alswrv   0     0       \_ keyring: _evm
1054989579 --alswrv   0     0          \_ user: FA0EF80BF06F80AC
```



注記

*この非対称鍵のペアは、**evmctl sign** コマンドを使用して、ファイルの拡張属性の内容にデジタル署名するために使用できます。拡張属性は後でカーネルによって検証されます。*

h. *カーネルマスターキーを作成して、EVM 鍵を保護します。*

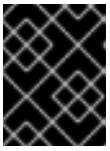
```
# dd if=/dev/urandom bs=1 count=32 2>/dev/null | keyctl padd user kmk-user @u
```

*カーネルマスターキー (**kmk**) はすべて、カーネル領域メモリーに保持されます。カーネルマスターキー **kmk** の 32 バイトの Long 値は、**/dev/urandom** ファイルの乱数バイト数から生成し、ユーザーの (**@u**) キーリングに配置します。*

3. ***kmk** 鍵をもとに暗号化された EVM 鍵を作成します。*

```
# keyctl add encrypted evm-key "new user:kmk 64" @u
351426499
```

***kmk** を使用して 64 バイトのユーザーキー (**evm-key**) を生成してユーザー (**@u**) のキーリングに配置します。鍵のシリアル番号は、先の例の 2 行目にあります。*



重要

ユーザーキーに **evm-key** *という名前を付ける必要があります。これは、EVM サブシステムが予想し、使用する名前であるためです。*

4. *EVM をアクティベートします。*

```
# echo 1 > /sys/kernel/security/evm
```

5. *EVM が初期化されていることを確認します。*

```
dmesg | tail -1
[…] EVM: initialized
```

8.6. INTEGRITY MEASUREMENT ARCHITECTURE によるファイルのハッシュの収集

Integrity Measurement Architecture (IMA) の最初の操作段階は、 **Measurement** *フェーズです。ここでは、ファイルのハッシュを作成し、それらをファイルの拡張属性 (xattrs) として保存することができます。以下のセクションでは、ファイルのハッシュを作成し、検証する方法を説明します。*

前提条件

- ***ima-evm-utils*、*attr*、*keyutils*** *パッケージがシステムにインストールされている。*
- *「Integrity Measurement Architecture および Extended Verification Module の有効化」 で説明されているように、Integrity Measurement Architecture(IMA) および Extended Verification Module(EVM) が有効になっています。*

手順

1. *テストファイルを作成します。*

```
# echo <Test_text> > test_file
```

2. *秘密鍵でファイルに署名します。*

```
# evmctl sign --imahash --key /etc/keys/privkey.pem test_file
```

test_file *ファイルのハッシュを作成すると、IMA はファイルが破損していないことを確認します。EVM は、* ***test_file*** *の拡張属性に保存されるハッシュコンテンツに署名することで、IMA ハッシュに間違いがないことを確認します。*

3. *オプションで、署名されたファイルの拡張属性を確認します。*

```
# getfattr -m . -d test_file
file: test_file
security.evm=0sAwICztVdCQCAZLtD7qAezGl8nGLgqFZzMzQp7Fm1svUet2Hy7TyI2vtT9/9Zf
BTKMK6Mjoyk0VX+DciS85XOCYX6WnV1LF2P/pmPRfputSEq9fVD4SWfKKj2rI7qwpndC1UqR
X1BbN3aRUYeoKQdPdl6Cz+cX4d7vS56FJkFhPGlhq/UQbBnd80=
security.ima=0sAfgqQ5/05X4w/ltZEfbogdI9+KM5
security.selinux="unconfined_u:object_r:admin_home_t:s0"
```

*この出力例は、SELinux および IMA および EVM ハッシュ値に関連する拡張属性を示しています。EVM は、**security.evm** 拡張属性をアクティブに追加し、ファイルコンテンツの整合性に直接関連する **security.ima** などの他のファイルの xattrs に対するオフライン改ざんを検出します。**security.evm** フィールドの値は、秘密鍵で生成された Hash-based Message Authentication Code (HMAC-SHA1) になります。*

関連情報

- *カーネル整合性サブシステムの詳細は、**公式のアップストリーム wiki ページ** を参照してください。*
- *TPM の詳細は、「Trusted Computing Group resources」を参照してください。*
- *暗号化された鍵の作成に関する詳細は、『RHEL 7 セキュリティーガイド』の「Trusted and Encrypted Keys」セクションを参照してください。*

第9章 改訂履歴

0.1-8

2020 年 9 月 29 日 (火) Jaroslav Klech (jklech@redhat.com)

- 7.9 GA 公開用ドキュメントバージョン

0.1-7

2020 年 3 月 31 日 (火) Jaroslav Klech (jklech@redhat.com)

- 7.8 GA 公開用ドキュメントバージョン

0.1-6

2019 年 8 月 6 日 (火) Jaroslav Klech (jklech@redhat.com)

- 7.7 GA 公開用ドキュメントバージョン

0.1-5

2018 年 10 月 19 日 (金) Jaroslav Klech (jklech@redhat.com)

- 7.6 GA 公開用ドキュメントバージョン。

0.1-4

2018 年 3 月 26 日 (月)、Marie Doleželová (mdolezel@redhat.com)

- 7.5 GA 公開用ドキュメントバージョン

0.1-3

2018 年 1 月 5 日 (月) Mark Flitter (mflitter@redhat.com)

- 7.5 ベータ版公開用ドキュメントバージョン

0.1-2

2017 年 7 月 31 日 (月)、Mark Flitter (mflitter@redhat.com)

- 7.4 GA 公開用ドキュメントバージョン

0.1-0

2017 年 4 月 20 日 (木)、Mark Flitter (mflitter@redhat.com)

- レビュー向けの初期ビルド