Red Hat Enterprise Linux 7

커널 관리 가이드

모듈, kpatch 또는 kdump를 사용하여 RHEL에서 Linux 커널 관리

Last Updated: 2024-10-04

Red Hat Enterprise Linux 7 커널 관리 가이드

모듈, kpatch 또는 kdump를 사용하여 RHEL에서 Linux 커널 관리

Jaroslav Klech
Red Hat Customer Content Services
jklech@redhat.com

Sujata Kurup
Red Hat Customer Content Services
skurup@redhat.com

Khushbu Borole
Red Hat Customer Content Services
kborole@redhat.com

Marie Dolezelova
Red Hat Customer Content Services

Mark Flitter
Red Hat Customer Content Services

Douglas Silas
Red Hat Customer Content Services

Eliska Slobodova
Red Hat Customer Content Services

Jaromir Hradilek
Red Hat Customer Content Services

Maxim Svistunov
Red Hat Customer Content Services

Robert Krátký
Red Hat Customer Content Services

Stephen Wadeley
Red Hat Customer Content Services

Florian Nadge
Red Hat Customer Content Services

법적 공지

초록

Kernel Administration Guide는 Red Hat Enterprise Linux 7 커널 유지 관리를 위한 작업을 문서화합니다. 이번 릴리스에서는 kpatch 사용, 커널 모듈 관리 및 커널 수동 업데이트에 대한 정보가 포함되어 있습니다.

차례

머리말

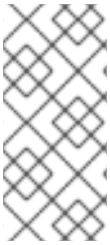
The *Kernel Administration Guide* 커널 작업에 대해 설명하고 몇 가지 실제 작업을 보여줍니다. 커널 모듈 사용에 대한 정보부터 가이드에 따라 **sysfs** 기능과의 상호 작용, 커널 수동 업그레이드 및 kpatch 사용에 대해 설명합니다. 또한 이 가이드에서는 커널 실패 시 **vmcore** 컬렉션을 설정 및 테스트하는 프로세스를 통해 크래시 덤프 메커니즘을 도입합니다.

The *Kernel Administration Guide* 또한 커널을 관리하는 선택된 사용 사례를 설명하고 명령줄 옵션, 커널 튜닝 가능 항목(Switch라고도 함)에 대한 참조 자료를 포함하고 커널 기능에 대한 간략한 설명을 포함합니다.

1장. 커널 모듈 작업

이 장의 설명:

- 커널 모듈이란 무엇입니까.
- 관련 리뷰: How to use the **kmod** 모듈 및 해당 종속성을 관리하는 유틸리티입니다.
- 커널 모듈의 동작을 제어하도록 모듈 매개 변수를 구성하는 방법.
- 부팅 시 모듈을 로드하는 방법.



참고

이 장에 설명된 커널 모듈 유틸리티를 사용하려면 먼저 다음을 확인하십시오. **kmod** root로를 실행하여 시스템에 패키지가 설치됩니다.

```
# yum install kmod
```

1.1. 커널 모듈이란 무엇입니까?

Linux 커널은 설계에 의해 모놀리식입니다. 그러나 각 사용 사례에 필요한 경우 선택적 또는 추가 모듈로 컴파일됩니다. 즉, 동적으로 로드된 커널 *모듈을 사용하여 커널 기능을 확장할 수 있습니다*. 커널 모듈은 다음을 제공할 수 있습니다.

- 새 하드웨어에 대한 지원을 추가하는 장치 드라이버입니다.
- **GFS2** 또는 **NFS** 와 같은 파일 시스템 지원.

커널 자체와 마찬가지로 모듈은 동작을 사용자 지정하는 매개 변수를 사용할 수 있습니다. 기본 매개 변수는 대부분의 경우 잘 작동합니다. 커널 모듈과 관련하여 사용자 공간 툴은 다음 작업을 수행할 수 있습니다.

- 현재 실행 중인 커널에 로드된 모듈 나열.
- 사용 가능한 매개 변수 및 모듈 관련 정보에 사용 가능한 모든 모듈 쿼리.
- 실행 중인 커널로 동적으로 모듈 로드 또는 언로드(제거)

kmod 패키지에서 제공하는 이러한 유틸리티 중 많은 유틸리티는 작업을 수행할 때 모듈 종속성을 고려합니다. 따라서 수동 dependency-tracking이 거의 필요하지 않습니다.

최신 시스템에서 커널 모듈은 필요할 때 다양한 메커니즘에 의해 자동으로 로드됩니다. 그러나 모듈을 수동으로 로드하거나 언로드해야 하는 경우가 있습니다. 예를 들어 기본 기능을 제공할 수 있거나 모듈이 예기치 않게 수행하는 경우 한 모듈이 다른 모듈을보다 우선합니다.

1.2. 커널 모듈 종속 항목

특정 커널 모듈은 때때로 하나 이상의 다른 커널 모듈에 의존합니다.
/lib/modules/<KERNEL_VERSION>/modules.dep 파일에는 해당 커널 버전에 대한 전체 커널 모듈 종속 항목이 포함되어 있습니다.

종속성 파일은 **kmod** 패키지의 일부인 **depmod** 프로그램으로 생성됩니다. **kmod** 에서 제공하는 많은 유틸리티에서는 작업을 수행할 때 모듈 종속성을 고려하여 **수동** dependency-tracking이 필요하지 않습니다.

주의

커널 모듈의 코드는 무제한 모드의 커널 공간에서 실행됩니다. 이 때문에 어떤 모듈을 로드해야하는지 염두에 두어야합니다.

추가 리소스

- **/lib/modules/<KERNEL_VERSION>/modules.dep** 에 대한 자세한 내용은 **modules.dep(5)** 매뉴얼 페이지를 참조하십시오.
- depmod의 synopsis 및 옵션을 포함한 자세한 내용은 **depmod (8)** 매뉴얼 페이지를 참조하십시오.

1.3. 현재 로드된 모듈 나열

다음과 같이 **lsmod** 명령을 실행하여 현재 커널에 로드된 모든 커널 모듈을 나열할 수 있습니다.

```
# 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 출력은 세 개의 열을 지정합니다.

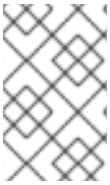
- **module**
  - 현재 메모리에 로드된 커널 모듈의 이름입니다.
- **크기**
  - 커널 모듈이 킬로바이트로 사용하는 메모리 양입니다.
- **사용됨**

- **모듈** 필드에 있는 종속 항목 수를 나타내는 10진수입니다.
- 종속 **모듈** 이름의 쉼표로 구분된 문자열입니다. 이 목록을 사용하여 먼저 언로드하려는 모듈에 따라 모든 모듈을 언로드할 수 있습니다.

마지막으로 **lsmod** 출력은 덜 자세한 정보이며 **/proc/modules** pseudo-file의 콘텐츠보다 상당히 쉽게 읽을 수 있습니다.

1.4. 모듈에 대한 정보 표시

modinfo < ***MODULE_NAME*** **>** 명령을 사용하여 커널 모듈에 대한 자세한 정보를 표시할 수 있습니다.



참고

커널 모듈의 이름을 인수로 입력하는 경우 중 하나에 대한 인수 **kmod** 유틸리티 이름 끝에 **.ko** 확장자를 추가하지 마십시오. 커널 모듈 이름에는 확장자가 없습니다. 해당 파일은 다음과 같습니다.

예 1.1. lsmod를 사용하여 커널 모듈에 대한 정보 나열

Intel PRO/1000 네트워크 드라이버인 **e1000e** 모듈에 대한 정보를 표시하려면 **root** 로 다음 명령을 입력합니다.

```
# 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. 시스템 런타임에 커널 모듈 로드

Linux 커널 기능을 확장하는 최적의 방법은 커널 모듈을 로드하는 것입니다. 다음 절차에서는 **modprobe** 명령을 사용하여 현재 실행 중인 커널에 커널 모듈을 찾아서 로드하는 방법을 설명합니다.

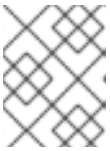
사전 요구 사항

- 루트 권한.
- **kmod** 패키지가 설치되어 있습니다.
- 각 커널 모듈이 로드되지 않습니다. 이를 확인하려면 Listing currently Loaded Modules 을 참조하십시오.

절차

1. 로드할 커널 모듈을 선택합니다.
   모듈은 **/lib/modules/$(uname -r)/kernel/<SUBSYSTEM>/** 디렉터리에 있습니다.
2. 관련 커널 모듈을 로드합니다.

```
# modprobe <MODULE_NAME>
```



참고

커널 모듈의 이름을 입력할 때 이름 끝에 **.ko.xz** 확장을 추가하지 마십시오. 커널 모듈 이름에는 확장자가 없습니다. 해당 파일은 다음과 같습니다.

3. 필요한 경우 관련 모듈이 로드되었는지 확인합니다.

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

모듈이 올바르게 로드되면 이 명령은 관련 커널 모듈을 표시합니다. 예를 들어 다음과 같습니다.

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

중요

이 절차에 설명된 변경 사항은 시스템을 재부팅한 후에는 **유지되지 않습니다**. 시스템 재부팅 **후에도 유지** 되도록 커널 모듈을 로드하는 방법에 대한 자세한 내용은 시스템 부팅 시 커널 모듈 로드 를 참조하십시오.

추가 리소스

- **modprobe** 에 대한 자세한 내용은 **modprobe(8)** 매뉴얼 페이지를 참조하십시오.

1.6. 시스템 런타임 시 커널 모듈 언로드

경우에 따라 실행 중인 커널에서 특정 커널 모듈을 언로드해야 합니다. 다음 절차에서는 **modprobe** 명령을 사용하여 현재 로드된 커널에서 시스템 런타임에서 커널 모듈을 찾아서 언로드하는 방법을 설명합니다.

사전 요구 사항

- 루트 권한.
- **kmod** 패키지가 설치되어 있습니다.

절차

1. **lsmod** 명령을 실행하고 언로드할 커널 모듈을 선택합니다.
   커널 모듈에 종속 항목이 있는 경우 커널 모듈을 언로드하기 전에 해당 항목을 언로드합니다. 종속 항목이 있는 모듈을 식별하는 방법에 대한 자세한 내용은 Listing currently Loaded Modules 및 Kernel module dependencies 를 참조하십시오.

2. 관련 커널 모듈을 언로드합니다.

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

커널 모듈의 이름을 입력할 때 이름 끝에 **.ko.xz** 확장을 추가하지 마십시오. 커널 모듈 이름에는 확장자가 없습니다. 해당 파일은 다음과 같습니다.

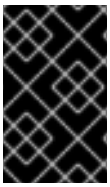
주의

커널 모듈을 실행 중인 시스템에서 사용할 때 언로드하지 마십시오. 이렇게 하면 불안정하거나 작동하지 않는 시스템이 발생할 수 있습니다.

3. 필요한 경우 관련 모듈이 언로드되었는지 확인합니다.

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

모듈이 성공적으로 언로드된 경우 이 명령에서 출력을 표시하지 않습니다.



중요

이 절차를 완료한 후에는 부팅 시 자동으로 로드되도록 정의된 커널 모듈은 시스템을 재부팅한 후에도 **언로드되지 않습니다**. 이 결과를 처리하는 방법에 대한 자세한 내용은 시스템 부팅 시 커널 모듈이 자동으로 로드되지 않도록 방지를 참조하십시오.

추가 리소스

- **modprobe** 에 대한 자세한 내용은 **modprobe(8)** 매뉴얼 페이지를 참조하십시오.

1.7. 시스템 부팅 시 커널 모듈 자동 로드

다음 절차에서는 부팅 프로세스 중에 자동으로 로드되도록 커널 모듈을 구성하는 방법을 설명합니다.

사전 요구 사항

- 루트 권한.
- **kmod** 패키지가 설치되어 있습니다.

절차

1. 부팅 프로세스 중에 로드할 커널 모듈을 선택합니다.
모듈은 **/lib/modules/$(uname -r)/kernel/<SUBSYSTEM>/** 디렉터리에 있습니다.

2. 모듈에 대한 구성 파일을 생성합니다.

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

참고

커널 모듈의 이름을 입력할 때 이름 끝에 **.ko.xz** 확장을 추가하지 마십시오. 커널 모듈 이름에는 확장자가 없습니다. 해당 파일은 다음과 같습니다.

3. 필요한 경우 재부팅 후 관련 모듈이 로드되었는지 확인합니다.

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

위의 예제 명령은 성공하고 관련 커널 모듈을 표시합니다.

중요

이 절차에 설명된 변경 사항은 시스템을 재부팅한 후에도 **지속됩니다**.

추가 리소스

- 부팅 프로세스 중에 커널 모듈을 로드하는 방법에 대한 자세한 내용은 **modules-load.d(5)** 매뉴얼 페이지를 참조하십시오.

1.8. 시스템 부팅 시 커널 모듈이 자동으로 로드되지 않도록 방지

다음 절차에서는 부팅 프로세스 중에 자동으로 로드되지 않도록 커널 모듈을 거부 목록에 추가하는 방법을 설명합니다.

사전 요구 사항

- 루트 권한.
- **kmod** 패키지가 설치되어 있습니다.
- 거부 목록에 있는 커널 모듈이 현재 시스템 구성에 필요하지 않은지 확인합니다.

절차

1. 거부 목록에 배치할 커널 모듈을 선택합니다.

```
$ lsmod

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

lsmod 명령은 현재 실행 중인 커널에 로드된 모듈 목록을 표시합니다.

- 또는 잠재적으로 로드되지 않도록 하려는 언로드되지 않은 커널 모듈을 식별합니다. 모든 커널 모듈은 **/lib/modules/<KERNEL_VERSION>/kernel/<SUBSYSTEM>/** 디렉터리에 있습니다.

2. 거부 목록에 대한 구성 파일을 생성합니다.

```
# 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
…
```

이 예제에서는 **vim** 편집기에서 편집한 **blacklist.conf** 파일의 내용을 보여줍니다. **blacklist** 행을 사용하면 부팅 프로세스 중에 관련 커널 모듈이 자동으로 로드되지 않습니다. 그러나 **blacklist** 명령은 거부 목록에 없는 다른 커널 모듈에 대한 종속성으로 모듈이 로드되지 않도록 합니다. 따라서 **설치** 행은 모듈을 설치하는 대신 **/bin/false** 가 실행됩니다.

해시 기호로 시작하는 줄은 파일을 더 쉽게 읽을 수 있도록 주석입니다.

참고

커널 모듈의 이름을 입력할 때 이름 끝에 **.ko.xz** 확장을 추가하지 마십시오. 커널 모듈 이름에는 확장자가 없습니다. 해당 파일은 다음과 같습니다.

3. 재구축하기 전에 현재 초기 램디스크 이미지의 백업 사본을 생성합니다.

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

위의 명령은 새 버전에 예기치 않은 문제가 있는 경우 backup **initramfs** 이미지를 생성합니다.

- 또는 커널 모듈을 거부 목록에 배치하려는 커널 버전에 해당하는 다른 초기 램디스크 이미지의 백업 사본을 생성합니다.

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

4. 변경 사항을 반영할 새 초기 ramdisk 이미지를 생성합니다.

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

- 현재 부팅된 것과 다른 커널 버전의 초기 램디스크 이미지를 빌드하는 경우 target **initramfs** 및 커널 버전을 모두 지정합니다.

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

5. 시스템을 재부팅합니다.

```
$ reboot
```

중요

이 절차에 설명된 변경 사항은 시스템을 재부팅 **한 후 적용됩니다**. 키 커널 모듈을 거부 목록에 부적절하게 배치하면 불안정하거나 작동하지 않는 시스템에 직면할 수 있습니다.

추가 리소스

- **dracut** 유틸리티에 대한 자세한 내용은 **dracut(8)** 매뉴얼 페이지를 참조하십시오.
- Red Hat Enterprise Linux 8 및 이전 버전에서 시스템 부팅 시 커널 모듈이 자동으로 로드되지 않도록 하는 방법에 대한 자세한 내용은 How do I avoid a kernel module from loading automatically?

1.9. 보안 부팅을 위해 커널 모듈에 서명

Red Hat Enterprise Linux 7에는 UEFI Secure Boot 기능이 포함되어 있으므로 UEFI Secure Boot가 활성화된 시스템에서 Red Hat Enterprise Linux 7을 설치하고 실행할 수 있습니다. Red Hat Enterprise Linux 7에는 UEFI 시스템에서 Secure Boot를 사용할 필요가 없습니다.

Secure Boot가 활성화된 경우 UEFI 운영 체제 부트 로더, Red Hat Enterprise Linux 커널 및 모든 커널 모듈은 개인 키로 서명되고 해당 공개 키로 인증되어야 합니다. 서명 및 인증되지 않은 경우 시스템은 부팅 프로세스를 완료할 수 없습니다.

Red Hat Enterprise Linux 7 배포에는 다음이 포함됩니다.

- 서명된 부트 로더
- 서명된 커널
- 서명된 커널 모듈

또한 서명된 첫 번째 단계 부트 로더와 서명된 커널에는 Red Hat 공개 키가 포함됩니다. 서명된 실행 파일 바이너리 및 임베디드 키를 사용하면 Red Hat Enterprise Linux 7이 UEFI 펌웨어에서 UEFI Secure Boot를 지원하는 시스템의 UEFI 펌웨어에서 제공하는 Microsoft UEFI Secure Boot 인증 기관 키를 설치, 부팅 및 실행할 수 있습니다.

참고

일부 UEFI 기반 시스템에는 Secure Boot 지원이 포함되어 있지 않습니다.

다음 섹션에서 제공되는 정보는 Secure Boot가 활성화된 UEFI 기반 빌드 시스템에서 Red Hat Enterprise Linux 7과 함께 사용하기 위해 개인 빌드 커널 모듈을 자체 서명하는 단계를 설명합니다. 이러한 섹션에서는 공개 키를 커널 모듈을 배포하려는 대상 시스템으로 가져오기 위한 사용 가능한 옵션에 대한 개요도 제공합니다.

커널 모듈에 서명하고 로드하려면 다음을 수행해야 합니다.

1. 시스템에 관련 유틸리티가 설치되어 있어야 합니다.
2. 커널 모듈 인증.
3. 공개 및 개인 키 쌍을 생성합니다.
4. 대상 시스템의 공개 키를 가져옵니다.
5. 개인 키로 커널 모듈에 서명합니다.
6. 서명된 커널 모듈을 로드합니다.

1.9.1. 사전 요구 사항

외부에서 빌드된 커널 모듈에 로그인할 수 있으려면 빌드 시스템에 다음 표에 나열된 유틸리티를 설치합니다.

표 1.1. 필수 유틸리티

유틸리티	패키지에 의해 제공	에서 사용됨	목적
openssl	**openssl**	빌드 시스템	공개 및 개인 X.509 키 쌍 생성
sign-file	**kernel-devel**	빌드 시스템	커널 모듈에 서명하는 데 사용되는 Perl 스크립트
perl	**perl**	빌드 시스템	서명 스크립트를 실행하는 데 사용되는 Perl 인터프리터
mokutil	**mokutil**	대상 시스템	공개 키 수동 등록에 사용되는 선택적 유틸리티
keyctl	**keyutils**	대상 시스템	시스템 키 링에 공개 키를 표시하는 데 사용되는 선택적 유틸리티

참고

커널 모듈을 빌드하고 서명하는 빌드 시스템은 UEFI Secure Boot를 활성화할 필요가 없으며 UEFI 기반 시스템도 필요하지 않습니다.

1.9.2. 커널 모듈 인증

Red Hat Enterprise Linux 7에서 커널 모듈이 로드되면 커널의 시스템 키 링에서 공개 X.509 키를 사용하여 모듈의 서명을 확인하고 커널의 시스템 차단 키 링에 있는 키를 제외하고 모듈 서명을 확인합니다. 다음 섹션에서는 키/키링의 소스, 시스템의 다른 소스에서 로드된 키의 예를 보여줍니다. 또한 사용자는 커널 모듈을 인증하는 데 필요한 사항을 확인할 수 있습니다.

1.9.2.1. 커널 모듈을 인증하는 데 사용되는 공개 키의 소스

부팅 중에 커널은 아래 표에 표시된 대로 일련의 영구 키 저장소 집합에서 X.509 키를 시스템 키 링 또는 시스템 black-list 키 링으로 로드합니다.

표 1.2. 시스템 키 링의 소스

X.509 키의 소스	키를 추가하는 사용자 기능	UEFI Secure Boot 상태	부팅 중 로드된 키
커널 내에 포함	없음	-	**.system_keyring**
UEFI Secure Boot "db"	제한됨	활성화되지 않음	없음
		활성화됨	**.system_keyring**

X.509 키의 소스	키를 추가하는 사용자 기능	UEFI Secure Boot 상태	부팅 중 로드된 키
UEFI Secure Boot "dbx"	제한됨	활성화되지 않음	없음
		활성화됨	**.system_keyring**
shim.efi 부트 로더에 포함	없음	활성화되지 않음	없음
		활성화됨	**.system_keyring**
MOK(Machine Owner Key) 목록	있음	활성화되지 않음	없음
		활성화됨	**.system_keyring**

시스템이 UEFI 기반이 아니거나 UEFI Secure Boot가 활성화되지 않은 경우 커널에 포함된 키만 시스템 키 링에 로드됩니다. 이 경우 커널을 다시 빌드하지 않고 해당 키 세트를 보강할 수 없습니다.

시스템 차단 목록 키 링은 취소된 X.509 키 목록입니다. 모듈이 블랙 리스트의 키로 서명되면 공개 키가 시스템 키 링에 있는 경우에도 인증이 실패합니다.

keyctl 유틸리티를 사용하여 시스템 키 링에 키에 대한 정보를 표시할 수 있습니다. 다음은 UEFI Secure Boot가 활성화되지 않은 Red Hat Enterprise Linux 7 시스템의 출력 예를 단축한 것입니다.

```
# 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...
```

다음은 UEFI Secure Boot가 활성화된 Red Hat Enterprise Linux 7 시스템의 출력 예를 단축한 것입니다.

```
# 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...
```

위의 출력에서는 **shim.efi** 부트 로더에 포함된 Red Hat Secure Boot "db" 키와 **Red Hat Secure Boot (CA 키 1)** 에서 두 개의 키를 추가하는 것을 보여줍니다. UEFI Secure Boot 관련 소스에서 키를 식별하는 커널 콘솔 메시지도 찾을 수 있습니다. 여기에는 UEFI Secure Boot db, shim 및 MOK 목록이 포함됩니다.

```
# 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. 커널 모듈 인증 요구사항

이 섹션에서는 UEFI Secure Boot 기능이 활성화된 시스템에 커널 모듈을 로드하기 위해 충족해야 하는 조건을 설명합니다.

UEFI Secure Boot가 활성화되어 있거나 **module.sig_enforce** 커널 매개 변수가 지정된 경우 시스템 키링의 키를 사용하여 인증된 서명된 커널 모듈만 로드할 수 있습니다. 또한 공개 키는 시스템 차단 목록 키링에 없어야 합니다.

UEFI Secure Boot가 비활성화되고 **module.sig_enforce** 커널 매개 변수가 지정되지 않은 경우 공개 키 없이 서명되지 않은 커널 모듈과 서명된 커널 모듈을 로드할 수 있습니다. 이는 아래 표에 요약되어 있습니다.

표 1.3. 로드를 위한 커널 모듈 인증 요구사항

모듈 서명	공개 키 및 서명이 유효한 경우	UEFI Secure Boot 상태	sig_enforce	모듈 로드	커널 테인트
서명되지 않음	-	활성화되지 않음	활성화되지 않음	성공	있음
		활성화되지 않음	활성화됨	실패	-
		활성화됨	-	실패	-
signed	없음	활성화되지 않음	활성화되지 않음	성공	있음
		활성화되지 않음	활성화됨	실패	-
		활성화됨	-	실패	-
signed	있음	활성화되지 않음	활성화되지 않음	성공	없음
		활성화되지 않음	활성화됨	성공	없음
		활성화됨	-	성공	없음

1.9.3. 공개 및 개인 X.509 키 쌍 생성

Secure Boot 지원 시스템에서 커널 모듈을 사용하는 데 성공하려면 공개 및 개인 X.509 키 쌍을 생성해야 합니다. 나중에 개인 키를 사용하여 커널 모듈에 서명합니다. 또한 보안 부팅에 대해 해당 공개 키를 MOK(Machine Owner Key)에 추가하여 서명된 모듈의 유효성을 검사해야 합니다. 이에 대한 지침은 1.9.4.2절. “시스템 관리자가 MOK 목록에 공개 키를 수동으로 추가” 을 참조하십시오.

이 키 쌍 생성의 일부 매개변수는 구성 파일을 사용하여 가장 잘 지정됩니다.

1. 키 쌍 생성에 대한 매개 변수를 사용하여 구성 파일을 생성합니다.

```
# 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. 다음 예와 같이 X.509 공개 및 개인 키 쌍을 만듭니다.

```
# 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
```

공개 키는 ***my_signing_key_pub*.der** 파일에 기록되고 개인 키는 ***my_signing_key*.priv** 파일에 기록됩니다.

3. 커널 모듈을 인증하고 로드하려는 모든 시스템에 공개 키를 등록합니다.
자세한 내용은 1.9.4절. "대상 시스템에 공개 키 등록" 의 내용을 참조하십시오.

주의

강력한 보안 조치 및 액세스 정책을 적용하여 개인 키의 콘텐츠를 보호합니다. 잘못된 손에서 키는 해당 공개 키로 인증된 시스템을 손상시키는 데 사용할 수 있습니다.

1.9.4. 대상 시스템에 공개 키 등록

Red Hat Enterprise Linux 7이 Secure Boot가 활성화된 UEFI 기반 시스템에서 부팅되면 커널은 Secure Boot db 키 데이터베이스에 있는 모든 공개 키를 링하지만 취소된 키의 dbx 데이터베이스에는 작동하지 않습니다. 아래 섹션에서는 시스템 키 링에서 공개 키를 사용하여 커널 모듈을 인증할 수 있도록 대상 시스템에서 공개 키를 가져오는 다양한 방법에 대해 설명합니다.

1.9.4.1. 공개 키를 포함한 팩토리 펌웨어 이미지

시스템에서 커널 모듈을 쉽게 인증하려면 시스템 벤더에 공개 키를 해당 팩토리 펌웨어 이미지의 UEFI Secure Boot 키 데이터베이스에 통합하도록 요청하는 것이 좋습니다.

1.9.4.2. 시스템 관리자가 MOK 목록에 공개 키를 수동으로 추가

MOK(Machine Owner Key) 기능 기능을 사용하여 UEFI Secure Boot 키 데이터베이스를 확장할 수 있습니다. Red Hat Enterprise Linux 7이 Secure Boot가 활성화된 UEFI 지원 시스템에서 부팅되면 MOK 목록의 키 외에도 키 데이터베이스의 키 외에 MOK 목록의 키도 시스템 키 링에 추가됩니다. MOK 목록 키는 Secure Boot 데이터베이스 키와 동일한 방식으로 영구적으로 안전하게 저장되지만, 두 가지 별도의 시설입니다. MOK 기능은 **shim.efi**,**MokManager.efi**,**grubx64.efi** 및 Red Hat Enterprise Linux 7 **mokutil** 유틸리티에서 지원됩니다.

MOK 키를 등록하려면 각 대상 시스템의 UEFI 시스템 콘솔에서 사용자의 수동 상호 작용이 필요합니다. 그러나 MOK 기능은 새로 생성된 키 쌍을 테스트하고 서명한 커널 모듈을 테스트하는 편리한 방법을 제공합니다.

MOK 목록에 공개 키를 추가하려면 다음을 수행합니다.

1. MOK 목록에 공개 키를 추가합니다.

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

   MOK 등록 요청에 대한 암호를 입력하고 암호를 확인하라는 메시지가 표시됩니다.

2. 시스템을 재부팅합니다.
   shim.efi 에서 보류 중인 MOK 키 등록 요청을 알 수 있으며, UEFI 콘솔에서 등록을 완료할 수 있도록 **MokManager.efi** 를 시작합니다.

3. 이전에 이 요청과 연결된 암호를 입력하고 등록을 확인합니다.
   공개 키가 MOK 목록에 추가되며 이는 영구적입니다.

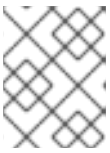
키가 MOK 목록에 있으면 자동으로 시스템 키 링으로 전파되고 UEFI Secure Boot가 활성화되면 부팅됩니다.

1.9.5. 개인 키를 사용하여 커널 모듈 서명

커널 모듈이 준비되었다고 가정합니다.

- Perl 스크립트를 사용하여 커널 모듈에 개인 키로 서명합니다.

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

  참고

  Perl 스크립트를 사용하려면 개인 키와 공개 키가 포함된 파일과 서명할 커널 모듈 파일을 모두 제공해야 합니다.

  커널 모듈은 ELF 이미지 형식이며 Perl 스크립트가 컴퓨팅되며 커널 모듈 파일의 ELF 이미지에 직접 서명을 추가합니다. **modinfo** 유틸리티를 사용하여 커널 모듈 서명에 대한 정보를 표시할 수 있습니다(있는 경우). **modinfo** 사용에 대한 자세한 내용은 1.4절. “모듈에 대한 정보 표시” 을 참조하십시오.

  추가된 서명은 ELF 이미지 섹션에 포함되지 않으며 ELF 이미지의 공식적인 부분이 아닙니다. 따라서 **readelf** 와 같은 유틸리티에서는 커널 모듈에 서명을 표시할 수 없습니다.

이제 커널 모듈을 로드할 준비가 되었습니다. 서명된 커널 모듈은 UEFI Secure Boot가 비활성화된 시스템 또는 UEFI가 아닌 시스템에서도 로드할 수 있습니다. 즉, 서명된 버전과 서명되지 않은 버전의 커널 모듈을 모두 제공할 필요가 없습니다.

1.9.6. 서명된 커널 모듈 로드

공개 키가 등록되어 시스템 키 링에 있으면 **mokutil** 을 사용하여 MOK 목록에 공개 키를 추가합니다. 그런 다음 **modprobe** 명령을 사용하여 커널 모듈을 수동으로 로드합니다.

1. 선택적으로 공개 키를 등록하기 전에 커널 모듈이 로드되지 않는지 확인합니다.
   현재 로드된 커널 모듈을 나열하는 방법에 대한 자세한 내용은 1.3절. "현재 로드된 모듈 나열" 을 참조하십시오.

2. 현재 부팅의 시스템 키 링에 추가된 키를 확인합니다.

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

   공개 키가 아직 등록되지 않았으므로 명령의 출력에 표시되지 않아야 합니다.

3. 공개 키 등록을 요청합니다.

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

4. UEFI 콘솔을 재부팅하고 등록을 완료합니다.

   ```
   # reboot
   ```

5. 시스템 키 링의 키를 다시 확인합니다.

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

6. 모듈을 원하는 커널의 **/extra/** 디렉터리에 복사합니다.

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

7. 모듈식 종속성 목록을 업데이트합니다.

   ```
   # depmod -a
   ```

8. 커널 모듈을 로드하고 성공적으로 로드되었는지 확인합니다.

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

   a. 부팅 시 모듈을 로드하려면 선택적으로 **/etc/modules-loaded.d/my_module.conf** 파일에 추가합니다.

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

2장. SYSCTL 및 커널 튜닝 가능 항목으로 작업

2.1. 커널 튜닝 가능 항목이란 무엇입니까?

커널 튜닝 가능 항목은 부팅 시 또는 시스템이 실행되는 동안 필요에 따라 Red Hat Enterprise Linux의 동작을 사용자 지정하는 데 사용됩니다. 일부 하드웨어 매개변수는 부팅 시에만 지정되며 시스템이 실행되면 변경할 수 없지만, 대부분의 경우 필요에 따라 변경하고 다음 부팅 시 영구적으로 설정할 수 있습니다.

2.2. 커널 튜닝 가능 항목 사용 방법

커널 튜닝 가능 항목을 수정하는 방법은 세 가지가 있습니다.

1. **sysctl** 명령 사용
2. **/etc/sysctl.d/** 디렉터리에서 구성 파일 수동 수정
3. 쉘을 통해 **/proc/sys**에 마운트된 가상 파일 시스템과 상호 작용

참고

모든 부팅 시간 매개 변수가 sysfs 하위 시스템을 제어하는 것은 아니며, 일부 하드웨어 특정 옵션을 커널 명령 줄에 설정해야 합니다. 이 가이드의 Kernel Parameters 섹션은 해당 옵션을 해결합니다.

2.2.1. sysctl 명령 사용

sysctl 명령은 커널 튜닝 가능 항목을 나열, 읽기, 설정하는 데 사용됩니다. 튜닝 가능 항목을 일시적으로 또는 영구적으로 나열하거나 읽고 설정할 때 튜닝 가능 항목을 필터링할 수 있습니다.

1. 변수 나열

```
# sysctl -a
```

2. 변수 읽기

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

3. 임시로 변수 작성

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

4. 영구적으로 변수 작성

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

2.2.2. /etc/sysctl.d에서 파일 수정

부팅 시 기본값을 덮어쓰려면 **/etc/sysctl.d** 에 파일을 수동으로 채울 수도 있습니다.

1. **/etc/sysctl.d**에 새 파일을 만듭니다.

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

2. 다음 형식으로 행당 하나씩 설정할 변수를 포함합니다.

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

3. 파일을 저장

4. 머신을 재부팅하기 위해 머신을 재부팅하려면 머신을 재부팅하지 않고 변경 사항을 적용합니다.
   또는
   Execute **sysctl -p /etc/sysctl.d/99-custom.conf**

2.3. 어떤 튜닝 가능 항목을 제어할 수 있습니까?

튜닝 가능 항목은 커널 sybsystem에 의해 그룹으로 나뉩니다. Red Hat Enterprise Linux 시스템에는 다음과 같은 튜닝 가능 항목이 있습니다.

표 2.1. sysctl 인터페이스 테이블

클래스	하위 시스템
abi	실행 도메인 및 개인 정보
crypto	암호화 인터페이스
debug	커널 디버깅 인터페이스
dev	장치 관련 정보
fs	글로벌 및 특정 파일 시스템 튜닝 가능 항목
kernel	글로벌 커널 튜닝 가능
net	네트워크 튜닝 가능 항목
sunrpc	sun Remote Procedure Call (NFS)
user	사용자 네임스페이스 제한
vm	메모리, 버퍼 및 캐시 튜닝 및 관리

2.3.1. 네트워크 인터페이스 튜닝 가능 항목

시스템 관리자는 네트워킹 튜닝 가능 항목을 통해 실행 중인 시스템의 네트워크 구성을 조정할 수 있습니다.

네트워킹 튜닝 가능 항목은 **/proc/sys/net** 디렉토리에 포함되어 있으며 여기에는 다양한 네트워킹 주제를 위한 여러 하위 디렉터리가 포함됩니다. 네트워크 구성을 조정하려면 시스템 관리자가 이러한 하위 디렉터리 내의 파일을 수정해야 합니다.

가장 자주 사용되는 디렉토리는 다음과 같습니다.

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

/proc/sys/net/core/ 디렉터리에는 커널 계층과 네트워킹 계층 간의 상호 작용을 제어하는 다양한 설정이 포함되어 있습니다. 이러한 튜닝 가능 항목 중 일부를 조정하여 시스템 성능을 개선할 수 있습니다. 예를 들어 수신 대기열의 크기를 늘려 최대 연결 또는 네트워크 인터페이스 전용 메모리를 늘릴 수 있습니다. 시스템의 성능은 개별 문제에 따라 다른 측면에 따라 다릅니다.

/proc/sys/net/ipv4/ 디렉터리에는 시스템의 공격을 방지하거나 라우터 역할을 할 때 유용한 추가 네트워킹 설정이 포함되어 있습니다. 디렉터리에는 IP 및 TCP 변수가 모두 포함되어 있습니다. 이러한 변수에 대한 자세한 설명은 **/usr/share/doc/kernel-doc-<version>/Documentation/networking/ip-sysctl.txt** 를 참조하십시오.

/proc/sys/net/ipv4/ 디렉터리에 있는 다른 디렉터리는 네트워크 스택의 다른 측면을 다룹니다.

1. **/proc/sys/net/ipv4/conf/** - 모든 특수 구성을 재정의하는 구성되지 않은 장치 및 설정에 대한 기본 설정을 포함하여 각 시스템 인터페이스를 다양한 방식으로 구성할 수 있습니다.
2. **/proc/sys/net/ipv4/neigh/** - 시스템에 직접 연결된 호스트와 통신할 수 있는 설정이 포함되어 있으며, 두 단계 이상의 시스템에 대한 다른 설정을 포함합니다.
3. **/proc/sys/net/ipv4/route/** - 시스템의 인터페이스와 함께 라우팅에 적용되는 사양을 포함합니다.

이 네트워크 튜닝 가능 항목은 IPv4 인터페이스와 관련이 있으며 **/proc/sys/net/ipv4/{all, <interface_name>}/** 디렉터리에서 액세스할 수 있습니다.

커널 설명서 사이트에서 다음 매개변수에 대한 설명이 채택되었습니다.[1]

log_martians

작동하지 않는 주소가 있는 패킷을 커널 로그에 기록합니다.

유형	기본값
부울	0

하나 이상의 **conf/{all,interface}/log_martians** 이 TRUE로 설정되어 있는 경우 사용 가능

추가 리소스

- net.ipv4.conf.all.log_martians의 커널 매개 변수는 무엇입니까?
- 메시지 파일에서 "마트안 소스" 로그가 표시되는 이유는 무엇입니까?

accept_redirects

ICMP 리디렉션 메시지를 수락합니다.

유형	기본값
부울	1

인터페이스에 대한 **accept_redirects** 는 다음 조건에서 활성화됩니다.

- **conf/{all,interface}/accept_redirects** 모두 TRUE(인터페이스 전달이 활성화된 경우)
- **conf/{all,interface}/accept_redirects** 중 하나 이상이 TRUE(인터페이스에 대한 전달이 비활성화됨)

자세한 내용은 ICMP 리디렉션 활성화 또는 비활성화를 참조하십시오.

forwarding

인터페이스에서 IP 전달을 활성화합니다.

유형	기본값
부울	0

추가 리소스

- 패킷 전달 및 비로컬 바인딩 활성화

mc_forwarding

멀티 캐스트 라우팅을 수행합니다.

유형	기본값
부울	0

- 읽기 전용 값
- 멀티 캐스트 라우팅 데몬이 필요합니다.
- 인터페이스에 대한 멀티 캐스트 라우팅을 활성화하려면 **conf/all/mc_forwarding** 도 TRUE로 설정해야 합니다.

추가 리소스

- 읽기 전용 동작에 대한 설명은 커널 매개 변수 "net.ipv4.conf.all.mc_forwarding"을 설정하는 동안 시스템이 "키에서 거부됨"을 보고하는 이유를 참조하십시오.

medium_id

장치를 연결된 매체로 구별하는 데 사용되는 임의의 값입니다.

유형	기본값
정수	0

참고

- 동일한 매체에 있는 두 개의 장치는 브로드캐스트 패킷이 한 개에서만 수신될 때 다른 ID 값을 가질 수 있습니다.

- 기본값 0은 장치가 해당 매체에 대한 유일한 인터페이스임을 의미합니다.
- 값 -1은 중간을 알 수 없음을 의미합니다.
- 현재 proxy_arp 동작을 변경하는 데 사용됩니다.
- proxy_arp 기능은 다른 미디어에 연결된 두 장치 간에 전달되는 패킷에 대해 활성화됩니다.

추가 리소스 - 예를 들어 Linux 2.2 및 2.4의 "medium_id" 기능 사용을 참조하십시오.

proxy_arp

arp를 프록시합니다.

유형	기본값
부울	0

인터페이스의 **proxy_arp** 가 활성화되어 있습니다. **conf/{all,interface}/proxy_arp** 중 하나가 TRUE로 설정되어 있지 않으면 비활성화됩니다.

proxy_arp_pvlan

프라이빗 VLAN 프록시 arp.

유형	기본값
부울	0

RFC 3069와 같은 기능을 지원하기 위해 프록시 arp 응답을 동일한 인터페이스로 다시 허용

shared_media

send(router) 또는 accept(host) RFC1620 공유 미디어 리디렉션.

유형	기본값
부울	1

참고

- secure_redirects를 덮어씁니다.
- 인터페이스에 **shared_media** 가 활성화되어 있어야 합니다.
  conf/{all,interface}/shared_media 중 하나가 TRUE로 설정된 경우

secure_redirects

인터페이스의 현재 게이트웨이 목록에 나열된 게이트웨이에만 ICMP 리디렉션 메시지를 수락합니다.

유형	기본값
부울	1

참고

- 비활성화된 경우에도 RFC1122 리디렉션 규칙이 적용됩니다.
- shared_media로 재정의됩니다.
- 인터페이스에 대한 secure_redirects가 활성화되어 있습니다.
  conf/{all,interface}/secure_redirects 중 하나가 TRUE로 설정된 경우

send_redirects

라우터인 경우 리디렉션을 보냅니다.

유형	기본값
부울	1

conf/{all,interface}/send_redirects 중 하나가 TRUE로 설정된 경우 인터페이스의 send_redirects가 활성화됩니다.

bootp_relay

이 호스트가 로컬 호스트가 아닌 소스 주소 0.b.c.d로 패킷을 수락합니다.

유형	기본값
부울	0

참고

- **이러한 패킷을 관리하려면 BOOTP 데몬이 활성화되어야 합니다.**
- **인터페이스에 BOOTP 릴레이를 활성화하려면 conf/all/bootp_relay 도 TRUE로 설정해야 합니다.**
- **구현되지 않음 Red Hat Enterprise Linux 네트워킹 가이드의 DHCP Relay Agent 를 참조하십시오.**

accept_source_route

SRR 옵션으로 패킷을 수락합니다.

유형	기본값
부울	1

참고

- 인터페이스에서 SRR 옵션이 있는 패킷을 허용하려면 **conf/all/accept_source_route** 도 TRUE 로 설정해야 합니다.

accept_local

로컬 소스 주소가 있는 패킷을 수락합니다.

유형	기본값
부울	0

참고

- 적절한 라우팅과 함께, 이는 전선을 통해 두 개의 로컬 인터페이스 간에 패킷을 지시하는 데 사용할 수 있고, 그들이 제대로 수락하도록 할 수 있습니다.
- accept_local이 효과를 적용하려면 **rp_filter** 를 0이 아닌 값으로 설정해야 합니다.

route_localnet

라우팅하는 동안 루프백 주소를 모티안 소스 또는 대상으로 간주하지 마십시오.

유형	기본값
부울	0

참고

- 이를 통해 로컬 라우팅 목적으로 127/8 을 사용할 수 있습니다.

rp_filter

소스 유효성 검사 활성화

유형	기본값
정수	0

값	효과
0	소스 검증 없음
1	RFC3704에 정의된 엄격한 모드 (Strict Reverse Path)
2	RFC3704 Loose Reverse Path에 정의된 느슨한 모드

참고

- **RFC3704에서 현재 권장되는 방법은 DDos 공격으로부터 IP 스푸핑을 방지하기 위해 엄격한 모드를 활성화하는 것입니다.**
- **symmetric 라우팅 또는 기타 복잡한 라우팅을 사용하는 경우 느슨한 모드를 사용하는 것이 좋습니다.**
- **{interface}에서 소스 검증을 수행할 때 conf/{all,interface}/rp_filter 에서 가장 높은 값이 사용됩니다.**

arp_filter

유형	기본값
부울	0

값	효과
0	(기본값) 커널은 다른 인터페이스의 주소를 사용하여 arp 요청에 응답할 수 있습니다. 이는 일반적으로 성공적인 의사소통의 기회를 증가하기 때문에 의미가 있습니다.
1	동일한subnet에 여러 네트워크 인터페이스를 사용할 수 있으며 커널이 ARP의 IP에서 패킷을 라우팅할지 여부에 따라 각 인터페이스에 대한 ARP를 사용할 수 있습니다(이 작업을 위해 라우팅을 기반으로 소스 라우팅을 사용해야 함). 즉, arp 요청에 응답하는 카드(일반적으로 1)를 제어할 수 있습니다.

참고

- **IP 주소는 특정 인터페이스가 아닌 Linux의 전체 호스트에 의해 소유됩니다. 로드 밸런싱과 같은 더 복잡한 설정의 경우에만 이 동작으로 인해 문제가 발생합니다.**
- **인터페이스에 대한 arp_filter 는 conf/{all,interface}/arp_filter 중 하나를 TRUE로 설정하면 활성화됩니다.**

arp_announce

인터페이스에서 전송되는 ARP 요청의 IP 패킷에서 로컬 소스 IP 주소를 무효화하기 위한 다양한 제한 수준을 정의합니다.

유형	기본값
정수	0

값	효과
0	(기본값) 모든 인터페이스에 구성된 모든 로컬 주소를 사용합니다.

값	효과
1	이 인터페이스의 대상 서브넷에 없는 로컬 주소를 방지합니다. 이 모드는 수신 인터페이스에 구성된 논리 네트워크의 일부가 되도록 이 인터페이스를 통해 이 인터페이스를 통해 연결할 수 있는 대상 호스트에 유용합니다. 요청을 생성할 때 대상 IP를 포함하는 모든 서브넷을 확인하고 해당 서브넷의 경우 소스 주소를 보존합니다. 이러한 서브넷이 없는 경우 수준 2에 대한 규칙에 따라 소스 주소를 선택합니다.
2	항상 이 타겟에 가장 적합한 로컬 주소를 사용하십시오. 이 모드에서는 IP 패킷의 소스 주소를 무시하고 대상 호스트와 통신하려는 로컬 주소를 선택합니다. 이러한 로컬 주소는 대상 IP 주소를 포함하는 발신 인터페이스의 모든 서브넷에서 기본 IP 주소를 찾아 선택합니다. 적합한 로컬 주소가 없으면 발신 인터페이스 또는 다른 모든 인터페이스에 있는 첫 번째 로컬 주소를 선택할 수 있으므로 요청에 대한 회신과 당사가 공개하는 소스 IP 주소에 상관없이 종종 발생합니다.

참고

- **conf/{all,interface}/arp_announce 의 가장 높은 값이 사용됩니다.**
- **제한 수준을 늘리면 해결된 대상에서 응답을 받을 수 있는 반면 수준을 낮추면 더 유효한 발신자의 정보가 표시됩니다.**

arp_ignore

로컬 대상 IP 주소를 확인하는 수신된 ARP 요청에 응답하여 응답을 전송하는 다양한 모드를 정의합니다.

유형	기본값
정수	0

값	효과
0	(기본값): 모든 인터페이스에 구성된 모든 로컬 대상 IP 주소에 대한 응답
1	대상 IP 주소가 들어오는 인터페이스에 구성된 로컬 주소인 경우에만 응답
2	대상 IP 주소가 들어오는 인터페이스에 구성된 로컬 주소이고 보낸 사람의 IP 주소가 이 인터페이스의 동일한 서브넷의 일부인 경우에만 응답하십시오.
3	범위 호스트로 구성된 로컬 주소에 대해 응답하지 마십시오. 글로벌 및 링크 주소에 대한 해결 방법만 회신됩니다.
4-7	reserved
8	{interface}에서 ARP 요청이 수신될 때 conf/{all,interface}/arp_ignore의 최대 값이 사용되는 모든 로컬 주소에 대해 응답하지 마십시오.

참고

arp_notify

주소 및 장치 변경 알림에 대한 모드를 정의합니다.

유형	기본값
부울	0

값	효과
0	아무것도 하지 않음
1	장치가 내장되거나 하드웨어 주소 변경 시 gratuitous arp 요청을 생성합니다.

참고

arp_accept

ARP 표에 IP가 없는 자유 ARP 프레임에 대한 동작 정의

유형	기본값
부울	0

값	효과
0	ARP 표에 새 항목을 생성하지 마십시오.
1	ARP 테이블에 새 항목을 만듭니다.

응답과 요청 유형인 경우 ARP 테이블을 업데이트할 ARP 테이블을 업데이트합니다. ARP 테이블에 gratuitous arp 프레임의 IP 주소가 이미 포함되어 있는 경우, 이 설정이 설정되거나 해제되었는지에 관계없이 arp 테이블이 업데이트됩니다.

app_solicit

멀티 캐스트 프로브로 다시 삭제하기 전에 netlink를 통해 사용자 공간 ARP 데몬으로 전송할 최대 프로브 수입니다(mcast_solicit 참조).

유형	기본값
정수	0

참고
see mcast_solicit

disable_policy

이 인터페이스에 대해 IPSEC 정책(SPD) 비활성화

유형	기본값
부울	0

needinfo

disable_xfrm

정책에 관계없이 이 인터페이스에서 IPSEC 암호화를 비활성화합니다.

유형	기본값
부울	0

needinfo

igmpv2_unsolicited_report_interval

다음 원치 않는 IGMPv1 또는 IGMPv2 보고서 다시 전송 간격(밀리초)입니다.

유형	기본값
정수	10000

노트
Milliseconds

igmpv3_unsolicited_report_interval

요청되지 않은 다음 IGMPv3 보고서 전송을 수행하는 간격(밀리초)입니다.

유형	기본값
정수	1000

노트
Milliseconds

tag

필요에 따라 사용할 수 있는 번호를 작성할 수 있습니다.

유형	기본값
정수	0

xfrm4_gc_thresh

IPv4 대상 캐시 항목에 대한 가비지 수집을 시작하는 임계값입니다.

유형	기본값
정수	1

참고
이 값을 두 번 표시해도 시스템은 새 할당을 거부합니다.

2.3.2. 글로벌 커널 튜닝 가능

시스템 관리자는 글로벌 커널 튜닝 가능 항목을 통해 실행 중인 시스템에서 일반 설정을 구성하고 모니터링할 수 있습니다.

글로벌 커널 튜닝 가능 항목은 **/proc/sys/kernel/** 디렉토리에 이름이 지정된 제어 파일로 직접 포함되거나 다양한 구성 항목에 대해 추가 하위 디렉터리에 그룹화됩니다. 글로벌 커널 튜닝 가능 항목을 조정하려면 시스템 관리자가 제어 파일을 수정해야 합니다.

커널 설명서 사이트에서 다음 매개변수에 대한 설명이 채택되었습니다.[2]

dmesg_restrict

권한이 없는 사용자가 **dmesg** 명령을 사용하여 커널 로그 버퍼의 메시지를 볼 수 없는지를 나타냅니다.
자세한 내용은 커널 sysctl 설명서를 참조하십시오.

core_pattern

코어 dumpfile 패턴 이름을 지정합니다.

최대 길이	기본값
128자	"core"

자세한 내용은 커널 sysctl 설명서를 참조하십시오.

hardlockup_panic

하드 잠금이 감지되면 커널 패닉을 제어합니다.

유형	값	효과
정수	0	커널이 하드 잠금에서 패닉을 일으키지 않음
정수	1	커널 잠금의 패닉

패닉을 일으키려면 시스템이 먼저 하드 잠금을 감지해야 합니다. 탐지는 nmi_watchdog 매개 변수로 제어합니다.

추가 리소스

- 커널 sysctl 문서
- softlockup 탐지기 및 하드lockup 탐지기

softlockup_panic

소프트 잠금이 감지되면 커널 패닉을 제어합니다.

유형	값	효과
정수	0	커널이 소프트 잠금에서 패닉을 일으키지 않음
정수	1	소프트 잠금 시 커널 패닉

기본적으로 RHEL7에서 이 값은 0입니다.

softlockup_panic 에 대한 자세한 내용은kernel_parameters 를 참조하십시오.

kptr_restrict

/proc 및 기타 인터페이스를 통해 커널 주소 노출에 제한이 있는지 여부를 나타냅니다.

유형	기본값
정수	0

값	효과
0	출력 전에 커널 주소를 해시
1	특정 조건에서는 인쇄된 커널 포인터를 0으로 교체
2	인쇄된 커널 포인터를 0의 무조건 대체

자세한 내용은 커널 sysctl 설명서를 참조하십시오.

nmi_watchdog

x86 시스템에서 하드 잠금 탐지기를 제어합니다.

유형	기본값
정수	0

값	효과
0	잠금 탐지기를 비활성화합니다.

값	효과
1	잠금 탐지기 활성화

하드 잠금 탐지기는 각 CPU가 인터럽트에 응답하는 기능을 모니터링합니다.

자세한 내용은 커널 sysctl 설명서를 참조하십시오.

watchdog_thresh

watchdog **hrtimer** hrtimer , NMI 이벤트 및 소프트/하드 잠금 임계값의 빈도를 제어합니다.

기본 임계값	소프트 잠금 임계값
10초	2 * **watchdog_thresh**

이 튜닝 가능 항목을 0으로 설정하면 잠금 탐지가 모두 비활성화됩니다.

자세한 내용은 커널 sysctl 설명서를 참조하십시오.

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

이러한 튜닝 가능 항목은 커널이 패닉해야 하는 상황에서 지정합니다.
패닉 매개변수 그룹에 대한 자세한 내용은커널 sysctl 설명서를 참조하십시오.

printk, printk_delay, printk_ratelimit, printk_ratelimit_burst, printk_devkmsg

이러한 튜닝 가능 항목은 커널 오류 메시지의 로깅 또는 인쇄를 제어합니다.
printk 매개변수 그룹에 대한 자세한 내용은Kernel sysctl 설명서를 참조하십시오.

Shmall, shmmax, shm_rmid_forced

이러한 튜닝 가능 항목은 공유 메모리에 대한 제한을 제어합니다.
shm 매개변수 그룹에 대한 자세한 내용은Kernel sysctl 설명서를 참조하십시오.

threads-max

fork() 시스템 호출에 의해 생성된 최대 스레드 수를 제어합니다.

min 값	최대 값
20	FUTEX_TID_MASK(0x3ffff)

threads-max 값은 사용 가능한 RAM 페이지에 대해 확인합니다. 스레드 구조가 사용 가능한 RAM 페이지의 너무 많은 부분을 차지하면 **threads-max** 가 그에 따라 줄어듭니다.

자세한 내용은 커널 sysctl 설명서를 참조하십시오.

pid_max

PID 할당 래핑 값입니다.
자세한 내용은 커널 sysctl 설명서를 참조하십시오.

numa_balancing

이 매개변수는 자동 NUMA 메모리 분산을 활성화하거나 비활성화합니다. NUMA 머신의 경우 CPU에서 원격 메모리에 액세스하는 경우 성능 저하가 발생합니다.
자세한 내용은 커널 sysctl 설명서를 참조하십시오.

numa_balancing_scan_period_min_ms, numa_balancing_scan_delay_ms, numa_balancing_scan_period_max_ms, numa_balancing_scan_size_mb

이러한 튜닝 가능 항목은 데이터를 작업이 실행 중인 로컬의 메모리 노드로 마이그레이션해야 하는지의 페이지가 올바르게 배치되는지 여부를 탐지합니다.
numa_balancing_scan 매개변수 그룹에 대한 자세한 내용은커널 sysctl 설명서를 참조하십시오.

---

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

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

3장. 커널 매개변수 및 값 목록

3.1. 커널 명령줄 매개변수

커널 인수라고도 하는 커널 명령줄 매개 변수는 부팅 시만 Red Hat Enterprise Linux의 동작을 사용자 지정하는 데 사용됩니다.

3.1.1. 커널 명령줄 매개변수 설정

이 섹션에서는 다음을 사용하여 AMD64 및 Intel 64 시스템 및 IBM Power Systems 서버에서 커널 명령줄 매개변수를 변경하는 방법을 설명합니다. GRUB2 부트 로더 및 IBM Z에서 사용zipl.

커널 명령줄 매개변수는 에 의해 생성되는 boot/grub/grub.cfg 구성 파일에 저장됩니다.GRUB2 부트 로더. 이 설정 파일을 편집하지 마십시오. 이 파일에 대한 변경은 구성 스크립트에 의해서만 수행됩니다.

AMD64 및 Intel 64 시스템 및 IBM Power Systems 하드웨어의 경우 GRUB2에서 커널 명령줄 매개변수 변경

1. **다음과 같은 일반 텍스트 편집기를 사용하여 root 로 /etc/default/grub 구성 파일을 엽니다.vim 또는 Gedit.**
2. **이 파일에서 다음과 유사한 GRUB_CMDLINE_LINUX 로 시작하는 행을 찾습니다.**

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

3. **필요한 커널 명령줄 매개변수 값을 변경합니다. 그런 다음 파일을 저장하고 편집기를 종료합니다.**
4. **다시 생성 GRUB2 편집한 기본 파일을 사용한 구성입니다. 시스템이 BIOS 펌웨어를 사용하는 경우 다음 명령을 실행합니다.**

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

UEFI 펌웨어가 있는 시스템에서 다음을 실행합니다.

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

위의 절차를 완료하면 부트 로더가 재구성되고, 구성 파일에 지정된 커널 명령줄 매개 변수가 다음 재부팅 후 적용됩니다.

IBM Z Hardware용 zipl에서 커널 명령줄 매개변수 변경

1. **과 같은 일반 텍스트 편집기를 사용하여 root 로 /etc/zipl.conf 구성 파일을 엽니다.vim 또는 Gedit.**
2. **이 파일에서 parameters= 섹션을 찾고 requiremed 매개 변수를 편집하거나 없는 경우 추가합니다. 그런 다음 파일을 저장하고 편집기를 종료합니다.**
3. **다시 생성 zipl configuration:**

```
# zipl
```

참고

추가 옵션을 사용하지 않고 **zipl** 명령만 실행하면 기본값이 사용됩니다. 사용 가능한 옵션에 대한 자세한 내용은 **zipl(8)** 매뉴얼 페이지를 참조하십시오.

위의 절차를 완료하면 부트 로더가 재구성되고, 구성 파일에 지정된 커널 명령줄 매개 변수가 다음 재부팅 후 적용됩니다.

3.1.2. 제어할 수 있는 커널 명령줄 매개변수

커널 명령줄 매개변수의 전체 목록은 https://www.kernel.org/doc/Documentation/admin-guide/kernel-parameters.txt 를 참조하십시오.

3.1.2.1. 하드웨어별 커널 명령줄 매개변수

pci=option[,option...]

PCI 하드웨어 하위 시스템의 동작 지정

설정	효과
earlydump	[X86] 커널이 모든 것을 변경하기 전에 PCI 구성 공간을 덤프합니다.
Off	[X86] PCI 버스를 조사하지 마십시오.
noaer	[PCIE] PCIEAER 커널 매개변수가 활성화된 경우 이 커널 부팅 옵션을 사용하여 PCIE 고급 오류 보고 사용을 비활성화할 수 있습니다.
noacpi	[X86]IRQ(Interrupt Request) 라우팅 또는 PCI 스캔에는 고급 구성 및 전원 인터페이스(ACPI)를 사용하지 마십시오.
bfsort	PCI 장치를 범위 우선 순서로 정렬합니다. 이 정렬은 이전 커널과 호환되는 장치 순서를 얻기 위해 수행됩니다.
nobfsort	PCI 장치를 범위 우선 순서로 정렬하지 마십시오.

추가 PCI 옵션은 **kernel-doc-<version>.noarch** 패키지에 있는 디스크 문서의 에 설명되어 있습니다. 여기서 '<version>'을 해당 커널 버전으로 교체해야 합니다.

acpi=option

고급 구성 및 전원 인터페이스의 동작 지정

설정	효과
acpi=off	ACPI 비활성화

설정	효과
acpi=ht	ACPI 부팅 테이블 구문 분석을 사용하지만 ACPI 인터프리터를 사용할 수 없습니다. 이 경우 Hyper Threading에 필요하지 **않은** ACPI 기능이 비활성화됩니다.
acpi=force	ACPI 하위 시스템을 활성화해야 합니다.
acpi=strict	ACPI 계층을 ACPI 사양을 완전히 준수하지 않는 플랫폼의 내결함성을 줄일 수 있도록 합니다.
acpi_sci=<value>	ACPI SCI 인터럽트를 설정합니다. 여기서 <value>는 edge,level,high,low 중 하나입니다.
acpi=noirq	IRQ 라우팅에 ACPI를 사용하지 마십시오.
acpi=nocmcff	수정된 오류를 위해 먼저 펌웨어(FF) 모드를 비활성화합니다. 이렇게 하면 HEST CMC 오류 소스를 구문 분석하여 펌웨어가 FF 플래그를 설정했는지 확인합니다. 이로 인해 중복 수정된 오류 보고서가 발생할 수 있습니다.

4장. 커널 기능

이 장에서는 많은 사용자 공간 툴을 활성화하는 커널 기능의 목적 및 사용에 대해 설명하고 해당 툴을 추가로 조사하기 위한 리소스를 포함합니다.

4.1. 제어 그룹

4.1.1. 컨트롤 그룹이란 무엇입니까?

참고

Control Group Namespaces는 Red Hat Enterprise Linux 7.5에서 기술 프리뷰입니다.

Linux 제어 그룹(cgroups)은 시스템 하드웨어 사용에 대한 제한을 활성화하여 cgroup 내에서 실행되는 개별 프로세스가 cgroup 구성에 허용된 만큼만 사용하도록 합니다.

제어 그룹은 네임스페이스 에서 활성화한 리소스에서 사용 볼륨을 제한합니다. 예를 들어 네트워크 네임스페이스를 사용하면 프로세스에서 특정 네트워크 카드에 액세스할 수 있으며 cgroup은 프로세스가 해당 카드의 사용을 50% 이상 초과하지 않도록 하여 다른 프로세스에서 대역폭을 사용할 수 있도록 합니다.

Control Group 네임스페이스는 /proc/self/ns/cgroup 인터페이스를 통해 개별 cgroup의 가상화된 보기를 제공합니다.

목적은 글로벌 네임스페이스에서 cgroup으로 권한 있는 데이터가 유출되지 않도록 하고 컨테이너 마이그레이션과 같은 다른 기능을 활성화하는 것입니다.

이제 컨테이너를 단일 cgroup과 연결하는 것이 훨씬 더 쉬워졌기 때문에 컨테이너에 훨씬 더 일관된 cgroup 보기가 있으므로 컨테이너 내에 작업이 속해 있는 cgroup의 가상화된 보기를 사용할 수 있습니다.

4.1.2. 네임스페이스란 무엇인가?

네임스페이스는 분리된 시스템 리소스를 가상으로 볼 수 있는 커널 기능입니다. 시스템 리소스에서 프로세스를 격리하면 프로세스가 상호 작용할 수 있는 항목을 지정하고 제어할 수 있습니다. 네임스페이스는 컨트롤 그룹의 필수 요소입니다.

4.1.3. 지원되는 네임스페이스

Red Hat Enterprise Linux 7.5 이상에서 지원되는 네임스페이스는 다음과 같습니다.

- mount
  - 마운트 네임스페이스는 파일 시스템 마운트 지점을 격리하여 각 프로세스에서 Wich 내에 고유한 파일 시스템 공간을 가질 수 있도록 합니다.
- UTS
  - 호스트 이름 및 NIS 도메인 이름
- IPC
  - System V IPC, POSIX 메시지 대기열
- PID

- 프로세스 ID

- 네트워크
  - 네트워크 장치, 스택, 포트 등.

- 사용자
  - 사용자 및 그룹 ID

- 제어 그룹
  - cgroup 분리

참고

제어 그룹 사용에 대한 자세한 내용은 리소스 관리 가이드에설명되어 있습니다.

4.2. 커널 소스 검사기

Linux Kernel Module Source Checker(ksc)는 지정된 커널 모듈에서 허용 목록에 없는 기호를 확인하는 툴입니다. Red Hat 파트너는 Red Hat bugzilla 데이터베이스의 버그를 제출하여 이 툴을 사용하여 화이트리스트 포함에 대한 기호 검토를 요청할 수도 있습니다.

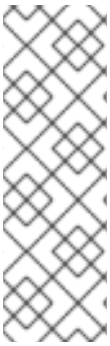
4.2.1. 사용법

툴에서 "-k" 옵션을 사용하여 모듈 경로 허용

```
# 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
```

출력은 **$HOME/ksc-result.txt** 에 저장됩니다. 화이트리스트 추가에 대한 기호 검토가 요청되면 공백이 없는 각 기호에 대한 사용 설명을 **ksc-result.txt** 파일에 추가해야 합니다. 그러면 "-p" 옵션으로**ksc** 를 실행하여 요청 버그가 제출될 수 있습니다.



참고

KSC는 현재 xz 압축을 지원하지 않습니다.ksc 툴은 xz 압축 방법을 처리하고 다음 오류를 보고할 수 없습니다.

> **Invalid architecture, (Only kernel object files are supported)**

이러한 제한이 해결될 때까지 시스템 관리자는 ksc 도구를 실행하기 전에 xz 압축을 사용하여 타사 모듈의 압축을 수동으로 해제해야 합니다.

4.3. 직접 파일 액세스 (DAX)

'파일 시스템 dax' 또는 'fs dax'로 알려진 파일에 대한 직접 액세스를 사용하면 애플리케이션이 장치에 대한 버퍼 액세스에 페이지 캐시를 사용하지 않고 dax-capable 스토리지 장치에서 데이터를 읽고 쓸 수 있습니다.

이 기능은 'ext4' 또는 'xfs' 파일 시스템을 사용할 때 사용할 수 있으며, **-o dax** 로 파일 시스템을 마운트하거나 **/etc/fstab** 의 마운트 항목에 대한 옵션 섹션에**dax** 를 추가하여 활성화할 수 있습니다.

코드 예제를 포함한 자세한 내용은 **kernel-doc** 패키지에서 찾을 수 있으며**/usr/share/doc/kernel-doc-<version>/Documentation/filesystems/dax.txt**에 저장됩니다. 여기서 '<version>'은 해당 커널 버전 번호입니다.

4.4. 사용자 공간에 대한 메모리 보호 키(PKU 또는 PKEYS라고도 함)

메모리 보호 키는 페이지 기반 보호를 시행하기 위한 메커니즘을 제공하지만, 애플리케이션이 보호 도메인을 변경할 때 페이지 테이블을 수정할 필요 없이 메커니즘을 제공합니다. 이는 각 페이지 테이블 항목에서 "보호 키"에 이전에 무시된 4비트를 사용하여 16개의 가능한 키를 지정하여 작동합니다.

메모리 보호 키는 일부 Intel CPU 칩셋의 하드웨어 기능입니다. 프로세서가 이 기능을 지원하는지 확인하려면 **/proc/cpuinfo**에 **pku** 가 있는지 확인하십시오.

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

이 기능을 지원하기 위해 CPU는 각 키에 대해 두 개의 개별 비트(Access Disable 및 Write Disable)를 사용하여 새 사용자 액세스 레지스터(PKRU)를 제공합니다. 새로운 레지스터를 읽고 쓰는 두 가지 새로운 명령(RDPKRU 및 WRPKRU)이 있습니다.

프로그래밍 예제를 포함한 추가 문서는 **kernel-doc** 패키지에서 제공하는**/usr/share/doc/kernel-doc-*/Documentation/x86/protection-keys.txt** 에서 확인할 수 있습니다.

4.5. KERNEL ADRESS SPACE LAYOUT RANDOMIZATION

Kernel Adress Space Layout Randomization (KASLR)은 Linux 커널의 보안을 개선하기 위해 함께 작동하는 두 부분으로 구성되어 있습니다.

- 커널 텍스트 KASLR
- 메모리 관리 KASLR

커널 텍스트 자체의 물리적 주소 및 가상 주소는 다른 위치로 임의로 지정됩니다. 커널의 물리적 주소는 64TB 미만의 아무 곳에나 있을 수 있지만 커널의 가상 주소는 [0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffc0000000] 간에 제한되며 1GB 공간입니다.

메모리 관리 KASLR에는 시작 주소가 특정 영역에서 임의의 3개의 섹션이 있습니다. 따라서 KASLR은 커널 주소 공간에 관심 기호가 어디에 있는지 확인하는 데 의존하는 경우 커널 실행을 악의적인 코드로 리디렉션하는 것을 방지할 수 있습니다.

메모리 관리 KASLR 섹션은 다음과 같습니다.

- 직접 매핑 섹션
- vmalloc 섹션
- vmemmap 섹션

KASLR 코드는 이제 Linux 커널로 컴파일되며 기본적으로 활성화되어 있습니다. 명시적으로 비활성화하려면 **nokaslr** 커널 옵션을 커널 명령줄에 추가합니다.

4.6. 고급 오류 보고(AER)

4.6.1. AER란?

AER(Advanced Error Reporting)는PCIe(Peripheral Component Interconnect Express) 장치에 대한 향상된 오류 보고 기능을 제공하는 커널 기능입니다. AER 커널 드라이버는 다음 작업을 위해PCIe AER 기능을 지원하는 루트 포트를 연결합니다.

- **오류가 발생한 경우 포괄적인 오류 정보 수집**
- **사용자에게 오류 보고**
- **오류 복구 작업 수행**

예 4.1. AER 출력 예

```
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
```

AER 가 오류를 캡처하면 오류 메시지를 콘솔에 보냅니다. 오류를 복구할 수 있는 경우 콘솔 출력은 경고입니다.

4.6.2. AER 메시지 수집 및 표시

AER 메시지를 수집하고 표시하려면 rasdaemon 프로그램을 사용합니다.

절차

1. **rasdaemon 패키지를 설치합니다.**

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

2. **rasdaemon 서비스를 활성화하고 시작합니다.**

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

3. **로그 오류(-summary 옵션)에 대한 요약을 표시하거나 오류 데이터베이스(- errors 옵션)에 저장된 오류를 표시하는 ras - mc-ctl 명령을 실행합니다.**

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

추가 리소스

- rasdaemon 서비스에 대한 자세한 내용은 **rasdaemon (8)** 매뉴얼 페이지를 참조하십시오.
- **ras-mc-ctl** 서비스에 대한 자세한 내용은**ras-mc-ctl(8)** 매뉴얼 페이지를 참조하십시오.

5장. 커널 수동 업그레이드

Red Hat Enterprise Linux 커널은 지원되는 하드웨어와의 무결성 및 호환성을 보장하기 위해 Red Hat Enterprise Linux 커널 팀에서 직접 구축했습니다. Red Hat에서 커널을 출시하기 전에 먼저 엄격한 품질 보증 테스트 세트를 통과해야 합니다.

Red Hat Enterprise Linux 커널은 RPM 형식으로 패키징되어 쉽게 업그레이드하고 사용하여 확인할 수 있습니다. Yum 또는 PackageKit 패키지 관리자. PackageKit Red Hat Content Delivery Network 서버를 자동으로 조회하고 커널 패키지를 포함하여 사용 가능한 업데이트로 패키지를 알려줍니다.

따라서 이 장에서는 **yum** 대신 **rpm** 명령을 사용하여 커널 패키지를 수동으로 업데이트해야 하는사용자에게만 유용합니다.

주의

가능하면 다음 중 하나를 사용하십시오. Yum 또는 PackageKit 새 커널을 설치할 패키지 관리자는 현재 커널을 교체하는 대신 항상 새 커널을 설치하기 때문에 시스템을 부팅할 수 없게 될 수 있습니다.

주의

Red Hat에서는 사용자 지정 커널을 지원하지 않습니다. 그러나 지침은 솔루션 문서 로부터 얻을 수 있습니다.

커널 패키지를 설치하는 방법에 대한 자세한 내용은 yum시스템 관리자 가이드의 관련 섹션을 참조하십시오.

Red Hat Content Delivery Network에 대한 자세한 내용은 시스템 관리자 가이드의 관련 섹션을 참조하십시오.

5.1. 커널 패키지 개요

Red Hat Enterprise Linux에는 다음과 같은 커널 패키지가 포함되어 있습니다.

- kernel - 단일 코어, 멀티 코어 및 멀티 프로세서 시스템에 대한 커널을 포함합니다.
- kernel-debug - 성능 저하를 통해 커널 진단을 위해 수많은 디버깅 옵션이 활성화된 커널을 포함합니다.
- kernel-devel - 커널 헤더와 makefiles를 충분히 포함하여 모듈을 빌드할 수 있습니다kernel 패키지.
- kernel-debug-devel - 커널 진단에 사용할 수 있는 수많은 디버깅 옵션이 있는 커널 개발 버전이 포함되어 있으며 성능이 저하됩니다.

- kernel-doc - 커널 소스의 문서 파일입니다. Linux 커널과 함께 제공되는 장치 드라이버의 다양한 부분은 이러한 파일에 문서화되어 있습니다. 이 패키지를 설치하면 로드 시 Linux 커널 모듈에 전달할 수 있는 옵션에 대한 참조가 제공됩니다.
  기본적으로 이러한 파일은 **/usr/share/doc/kernel-doc-*kernel_version*/** 디렉토리에 배치됩니다.

- kernel-headers - Linux 커널과 사용자 공간 라이브러리 및 프로그램 간에 인터페이스를 지정하는 C 헤더 파일을 포함합니다. 헤더 파일은 대부분의 표준 프로그램을 빌드하는 데 필요한 구조와 상수를 정의합니다.

- linux-firmware - 다양한 장치가 작동하기 위해 필요한 모든 펌웨어 파일을 포함합니다.

- perf -이 패키지는 다음을 포함합니다.perf Linux 커널의 성능 모니터링을 지원하는 툴입니다.

- kernel-abi-whitelists - Red Hat Enterprise Linux 커널 ABI와 외부 Linux 커널 모듈에 필요한 커널 기호 목록을 포함하여 Red Hat Enterprise Linux 커널 ABI와 관련된 정보를 포함합니다. yum 적용 지원을 위한 플러그인입니다.

- kernel-tools - Linux 커널 및 지원 문서를 조작하는 툴을 포함합니다.

5.2. 업그레이드 준비

커널을 업그레이드하기 전에 몇 가지 준비 단계를 수행하는 것이 좋습니다.

먼저 문제가 발생할 경우 시스템에 대해 부팅 미디어가 작동하는지 확인합니다. 부트 로더가 새 커널을 부팅하도록 제대로 구성되지 않은 경우 이 미디어를 사용하여 Red Hat Enterprise Linux로 부팅할 수 있습니다.

USB 미디어에는 종종 *펜 드라이브*, *엄지상 디스크* 또는 *키가* 외부에 연결된 하드 디스크 장치라고도 하는 플래시 장치 형태가 있습니다. 이 유형의 거의 모든 미디어가 **VFAT** 파일 시스템으로 포맷됩니다.
ext2,**ext3**,**ext4** 또는 **VFAT** 으로 포맷된 미디어에서 부팅 가능한 USB 미디어를 만들 수 있습니다.

배포 이미지 파일 또는 최소 부팅 미디어 이미지 파일을 USB 미디어에 전송할 수 있습니다. 장치에서 사용 가능한 공간이 충분한지 확인합니다. 배포 DVD 이미지, 배포 CD 이미지의 경우 약 4GB 또는 최소 부팅 미디어 이미지에는 약 10MB가 필요합니다.

Red Hat Enterprise Linux 설치 DVD에서 **boot.iso** 파일 사본이나 CD-ROM #1을 설치해야 하며**VFAT** 파일 시스템 및 약 16MB의 여유 공간으로 포맷된 USB 스토리지 장치가 필요합니다.

USB 스토리지 장치를 사용하는 방법에 대한 자세한 내용은 USB 키를 포맷 하는 방법과 그래픽이 아닌 환경 솔루션 문서에 USB 플래시 드라이브를 수동으로 마운트하는 방법을 검토하십시오.

다음 절차는 사용자가 복사한 파일과 동일한 경로 이름이 없는 한 USB 스토리지 장치의 기존 파일에는 영향을 미치지 않습니다. USB 부팅 미디어를 생성하려면 **root** 사용자로 다음 명령을 수행합니다.

1. 설치 **syslinux** 패키지가 시스템에 설치되어 있지 않은 경우 해당 패키지를 설치합니다. 이렇게 하려면 root로 **yum install**을 실행합니다. **syslinux** 명령.

2. 설치 SYSLINUX USB 스토리지 장치의 부트로더:

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

   ...여기서 *sdX* 는 장치 이름입니다.

3. **boot.iso** 및 USB 스토리지 장치에 대한 마운트 지점을 생성합니다.

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

4. Mount **boot.iso**:

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

5. USB 스토리지 장치를 마운트합니다.

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

6. 복사 ISOLINUX **boot.iso** 에서 USB 스토리지 장치로의 파일:

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

7. **boot.iso** 의 **isolinux.cfg** 파일을 USB 장치의 경우**syslinux.cfg** 파일로 사용합니다.

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

8. **boot.iso** 및 USB 스토리지 장치를 마운트 해제합니다.

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

9. 부팅 미디어를 사용하여 시스템을 재부팅하고 계속하기 전에 시스템을 부팅할 수 있는지 확인합니다.

또는 플로피 드라이브가 있는 시스템에서 를 설치하여 부트 디스켓을 만들 수 있습니다. mkbootdisk **root** 로 **mkbootdisk** 명령을 패키징하고 실행합니다. 사용 정보는 패키지를 설치한 후**man mkbootdisk** man 페이지를 참조하십시오.

설치된 커널 패키지를 확인하려면 쉘 프롬프트에서 **yum list installed "kernel-*"** 명령을 실행합니다. 출력은 시스템의 아키텍처에 따라 다음 패키지의 일부 또는 모두로 구성되며 버전 번호는 다를 수 있습니다.

```
# 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
```

출력에서 커널 업그레이드에 대해 다운로드해야 하는 패키지를 확인합니다. 단일 프로세서 시스템의 경우 필요한 유일한 패키지는 입니다. kernel 패키지. 다양한 패키지에 대한 설명은5.1절. “커널 패키지 개요” 을 참조하십시오.

5.3. 업그레이드된 커널 다운로드

시스템에서 업데이트된 커널을 사용할 수 있는지 확인하는 방법은 여러 가지가 있습니다.

- 보안 에라타 - 보안 문제를 해결하는 커널 업그레이드를 포함하여 보안 에라타에 대한 정보는 Red Hat 고객 포털의 보안 공지 를 참조하십시오.
- Red Hat Content Delivery Network - Red Hat Content Delivery Network에 서브스크립션하는 시스템의 경우 yum 패키지 관리자는 최신 커널을 다운로드하여 시스템에서 커널을 업그레이드할 수 있습니다. The Dracut 필요한 경우 유틸리티는 초기 RAM 파일 시스템 이미지를 생성하고 새 커널을 부팅하도록 부트 로더를 구성합니다. Red Hat Content Delivery Network에서 패키지 설치에 대한 자세한 내용은 시스템 관리자 가이드의 관련 섹션을 참조하십시오. Red Hat Content Delivery Network에 시스템 가입에 대한 자세한 내용은 시스템 관리자 가이드의 관련 섹션을 참조하십시오.

if yum Red Hat Network에서 업데이트된 커널을 다운로드하여 설치하는 데 사용되었으며5.5절. "초기 RAM 파일 시스템 이미지 확인" 및 5.6절. "부트 로더 확인" 의 지침에 따라 기본적으로 부팅할 커널을 변경하지 마십시오. Red Hat Network는 기본 커널을 최신 버전으로 자동 변경합니다. 커널을 수동으로 설치하려면 5.4절. "업그레이드 수행" 계속하십시오.

5.4. 업그레이드 수행

필요한 모든 패키지를 검색한 후 기존 커널을 업그레이드해야 합니다.

중요

새 커널에 문제가 있는 경우 이전 커널을 유지하는 것이 좋습니다.

쉘 프롬프트에서 커널 RPM 패키지가 포함된 디렉터리로 변경합니다. **rpm** 명령과 함께 **-i** 인수를 사용하여 이전 커널을 유지합니다. 부트 로더 문제가 생성되는 현재 설치된 커널을 덮어쓰므로 **-U** 옵션을 사용하지 마십시오. 예를 들어 다음과 같습니다.

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

다음 단계는 초기 RAM 파일 시스템 이미지가 생성되었는지 확인하는 것입니다. 자세한 내용은 5.5절. "초기 RAM 파일 시스템 이미지 확인" 를 참조하십시오.

5.5. 초기 RAM 파일 시스템 이미지 확인

초기 RAM 파일 시스템 이미지의 작업은 IDE, SCSI 또는 RAID와 같은 블록 장치 모듈을 사전 로드하여 해당 모듈이 일반적으로 상주하는 루트 파일 시스템을 그런 다음 액세스하고 마운트할 수 있도록 하는 것입니다. Red Hat Enterprise Linux 7 시스템에서 새 커널이 둘 중 하나를 사용하여 설치될 때마다 Yum, PackageKit또는 RPM 패키지 관리자, Dracut utility는 항상 *initramfs* 를 생성하기 위해 설치 스크립트에 의해 호출됩니다(initial RAM 파일 시스템 이미지).

/etc/sysctl.conf 파일이나 다른 **sysctl** 구성 파일을 수정하여 커널 속성을 변경하고 변경된 설정이 부팅 프로세스 초기에 사용된 경우 **dracut -f** 명령을 실행하여 Initial RAM File System Image를 다시 빌드해야 할 수 있습니다. 예를 들어 네트워킹과 관련된 변경 사항이 있고 네트워크 연결 스토리지에서 부팅되는 경우입니다.

IBM eServer System i("IBM eServer System i에서 초기 RAM 파일 시스템 이미지 및 커널 확인"참조) 이외의 모든 아키텍처에서 **dracut** 명령을 실행하여 **initramfs** 를 생성할 수 있습니다. 그러나 일반적으로 **initramfs** 를 수동으로 생성할 필요는 없습니다. 이 단계는 커널 및 관련 패키지가 Red Hat에서 배포하는 RPM 패키지에서 설치되거나 업그레이드되는 경우 자동으로 수행됩니다.

다음 절차에 따라 현재 커널 버전에 해당하는 **initramfs** 가 존재하고 **grub.cfg** 구성 파일에 올바르게 지정되었는지 확인할 수 있습니다.

초기 RAM 파일 시스템 이미지 확인

1. 루트 로서 **/boot** 디렉토리에 있는 콘텐츠를 나열하고 커널(**vmlinuz-*kernel_version***) 및 **initramfs-*kernel_version*** 을 최신 버전 번호로 찾습니다.

   예 5.1. 커널 및 initramfs 버전이 일치하는지 확인합니다.

   ```
   # 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
```

예 5.1. "커널 및 initramfs 버전이 일치하는지 확인합니다."다음을 보여줍니다.

- 3개의 커널이 설치되어 있습니다(또는 더 정확하게 세 개의 커널 파일이 **/boot** 디렉토리에 있습니다).
- 최신 커널은 **vmlinuz-3.10.0-78.el7.x86_64** 입니다.
- 커널 버전과 일치하는 **initramfs** 파일, **initramfs-3.10.0-78.el7.x86_64kdump.img** 도 존재합니다.

중요

/boot 디렉토리에서 여러 **initramfs-*kernel_version*kdump.img** 파일을 찾을 수 있습니다. 이 파일은 에 의해 생성된 특수 파일입니다. Kdump 커널 디버깅을 위한 메커니즘은 시스템을 부팅하는 데 사용되지 않으며, 무시해도 됩니다. **kdump** 에 대한 자세한 내용은Red Hat Enterprise Linux 7 커널 Crash Dump Guide를 참조하십시오.

2. **initramfs-*kernel_version*** 파일이 **/boot** 디렉토리의 최신 커널 버전과 일치하지 않거나 다른 상황에서는 **initramfs** 파일을 생성해야 할 수 있습니다.Dracut 유틸리티. 옵션 없이 **dracut** 을 루트로 호출하면 **/boot** 에 해당 디렉터리에 있는 최신 커널의**initramfs** 파일이 생성됩니다.

```
# dracut
```

dracut 을 기존**initramfs** 를 덮어쓰려면 **-f,--force** 옵션을 사용해야 합니다(예:**initramfs** 가 손상된 경우). 그렇지 않으면 **dracut** 이 기존**initramfs** 파일을 덮어쓰지 않습니다.

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

dracut *initramfs_name kernel_version*:을 호출하여 현재 디렉터리에서 **initramfs** 를 생성할 수 있습니다.

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

미리 로드되어야 하는 특정 커널 모듈을 지정해야 하는 경우 **add_dracutmodules+="** 모듈의 괄호

안에 해당 모듈의 이름(**.ko**)을 더 추가하여 **/etc/dracut.conf** 설정 파일의 지시문을 추가합니다. ***lsinitrd initramfs _file*** *명령을 사용하여* ***dracut***으로 생성된 ***initramfs*** *이미지 파일의 파일 내용을 나열할 수 있습니다.*

```
# 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]
```

옵션과 사용법에 대한 자세한 내용은 ***man dracut*** and ***man dracut.conf*** *를 참조하십시오.*

3. ***/boot/grub2/grub.cfg*** *구성 파일을 검사하여 부팅 중인커널 버전에* ***initramfs-kernel_version.img*** *파일이 있는지 확인합니다. 예를 들어 다음과 같습니다.*

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

자세한 내용은 5.6절. "부트 로더 확인" *를 참조하십시오.*

IBM eServer System i에서 초기 RAM 파일 시스템 이미지 및 커널 확인

IBM eServer System i 시스템에서 초기 RAM 파일 시스템 및 커널 파일이 ***addRamDisk*** *명령으로 생성되는 단일 파일로 결합됩니다. 이 단계는 커널 및 관련 패키지가 Red Hat에서 배포한 RPM 패키지에서 설치 또는 업그레이드되므로 수동으로 실행할 필요가 없습니다. 이 파일이 생성되었는지 확인하려면* **root** *로 다음 명령을 실행하여* ***/boot/vmlinitrd-kernel_version*** *파일이 이미 있는지 확인합니다.*

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

kernel_version 이 방금 설치된 커널 버전과 일치해야 합니다.

초기 RAM 파일 시스템 이미지의 변경 사항 다시 확인

예를 들어 시스템을 잘못 구성하고 더 이상 부팅하지 않는 경우 다음 절차에 따라 초기 RAM 파일 시스템 이미지에 대한 변경 사항을 취소해야 하는 경우도 있습니다.

초기 RAM 파일 시스템 이미지로 변경 사항 변경 사항

1. *GRUB 메뉴에서 복구 커널을 선택하는 시스템을 재부팅합니다.*
2. ***initramfs*** *에서 오작동으로 인한 잘못된 설정을 변경합니다.*
3. *root로 다음 명령을 실행하여 올바른 설정으로* ***initramfs*** *를 재생성합니다.*

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

*예를 들어 **sysctl.conf** 파일에 **vm.nr_hugepages** 를 잘못 설정한 경우 위의 절차가 유용할 수 있습니다. **sysctl.conf** 파일은 **initramfs** 에 포함되어 있기 때문에 새 **vm.nr_hugepages** 설정이 **initramfs** 에 적용되며 이로 인해 initramfs가 다시 작성됩니다. 그러나 설정이 올바르지 않으므로 새 **initramfs** 가 손상되고 새로 빌드된 커널이 부팅되지 않으므로 위 절차를 사용하여 설정을 수정해야 합니다.*

초기 RAM 파일 시스템 이미지의 콘텐츠 나열
***initramfs** 에 포함된 파일을 나열하려면 root로 다음 명령을 실행합니다.*

```
# lsinitrd
```

***/etc** 디렉토리에 있는 파일만 나열하려면 다음 명령을 사용합니다.*

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

*현재 커널의 **initramfs** 에 저장된 특정 파일의 내용을 출력하려면 **-f** 옵션을 사용합니다.*

```
# lsinitrd -f filename
```

*예를 들어 **sysctl.conf** 의 내용을 출력하려면 다음 명령을 사용합니다.*

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

*커널 버전을 지정하려면 **--kver** 옵션을 사용합니다.*

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

예를 들어 커널 버전 3.10.0-327.10.1.el7.x86_64에 대한 정보를 나열하려면 다음 명령을 사용합니다.

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

5.6. 부트 로더 확인

***yum** 명령을 사용하거나 **rpm** 명령을 사용하여 커널을 설치할 수 있습니다.*

***rpm** 을 사용하여 커널을 설치할 때 커널 패키지는 해당 새 커널의 부트 로더 구성 파일에 항목을 생성합니다.*

*두 명령 모두 **/etc/sysconfig/kernel** 구성 파일에 다음 설정을 포함하는 경우에만 기본 커널로 부팅되도록 새 커널을 구성합니다.*

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

***DEFAULTKERNEL** 옵션은 기본 커널 패키지 유형을 지정합니다. **UPDATEDEFAULT** 옵션은 새 커널 패키지가 새 커널을 기본값으로 만들지 여부를 지정합니다.*

6장. 커널 라이브 패치로 패치 적용

Red Hat Enterprise Linux 커널 라이브 패치 솔루션을 사용하여 프로세스를 재부팅하거나 다시 시작하지 않고도 실행 중인 커널을 패치할 수 있습니다.

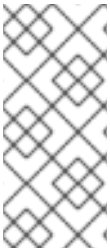
이 솔루션을 통해 시스템 관리자는 다음을 수행합니다.

- 중요한 보안 패치를 커널에 즉시 적용할 수 있습니다.
- 장기 실행 작업이 완료될 때까지, 사용자가 로그아웃하거나 예약된 다운타임을 기다릴 필요가 없습니다.
- 시스템의 가동 시간을 더 많이 제어하고 보안 또는 안정성을 포기하지 마십시오.

커널 라이브 패치 솔루션을 사용하여 중요하거나 중요한 CVE를 모두 해결하는 것은 아닙니다. 당사의 목표는 보안 관련 패치에 필요한 재부팅을 줄이는 것이 아니라 보안 관련 패치를 완전히 제거하는 것입니다. 라이브 패치 범위에 대한 자세한 내용은 다음을 참조하십시오. Customer Portal Solutions article

주의

일부 비호환성은 커널 라이브 패치와 다른 커널 하위 구성 요소 사이에 존재합니다. 커널 라이브 패치를 사용하기 전에 6.1절. "kpatch의 제한 사항" 섹션을 주의 깊게 읽으십시오.



참고

커널 라이브 패치 업데이트의 지원 주기에 대한 자세한 내용은 다음을 참조하십시오.

- 커널 라이브 패치 지원 케싱 업데이트
- 커널 라이브 패치 라이프 사이클

6.1. KPATCH의 제한 사항

- **kpatch** 기능은 범용 커널 업그레이드 메커니즘이 아닙니다. 시스템을 재부팅할 때 간단한 보안 및 버그 수정 업데이트를 적용하는 데 사용됩니다.
- 패치를 로드하거나 로드한 후 **SystemTap** 또는 **kprobe** 툴을 사용하지 마십시오. 이러한 프로브가 제거될 때까지 패치가 적용되지 않을 수 있습니다.

6.2. 타사 라이브 패치 지원

kpatch 유틸리티는 Red Hat 리포지토리에서 제공하는 RPM 모듈을 사용하여 Red Hat에서 지원하는 유일한 커널 라이브 패치 유틸리티입니다. Red Hat은 Red Hat 자체에서 제공하지 않은 실시간 패치를 지원하지 않습니다.

타사 라이브 패치를 지원하는 경우 패치를 제공한 공급 업체에 문의하십시오.

타사 라이브 패치로 실행되는 모든 시스템의 경우 Red Hat은 Red Hat 제품 및 지원 소프트웨어를 사용하여 재생을 요청할 수 있는 권한을 보유합니다. 이것이 가능하지 않은 경우 동일한 동작이 관찰되는지 확인하기 위해 실시간 패치를 적용하지 않고 테스트 환경에 유사한 시스템과 워크로드를 배포해야 합니다.

타사 소프트웨어 지원 정책에 대한 자세한 내용은 Red Hat 글로벌 지원 서비스에서 타사 소프트웨어, 드라이버 및/또는 인증되지 않은 하드웨어/하이퍼 바이저 또는 게스트 운영 체제를 처리하는 방법을참조하십시오.

6.3. 커널 라이브 패치 액세스

*커널 라이브 패치 기능은 RPM 패키지로 제공되는 커널 모듈(**.ko** 파일)으로 구현됩니다.*

모든 고객은 일반적인 채널을 통해 제공되는 커널 라이브 패치에 액세스할 수 있습니다. 그러나 연장된 지원 제공을 서브스크립션하지 않은 고객은 다음 마이너 릴리스가 출시되면 현재 마이너 릴리스의 새 패치에 액세스할 수 없습니다. 예를 들어 표준 서브스크립션이 있는 고객은 RHEL 8.3이 릴리스될 때까지만 RHEL 8.2 커널을 패치할 수 있습니다.

6.4. 커널 라이브 패치 구성 요소

커널 라이브 패치의 구성 요소는 다음과 같습니다.

커널 패치 모듈

- *커널 라이브 패치를 위한 전달 메커니즘입니다.*
- *패치되는 커널에 맞게 특별히 빌드된 커널 모듈입니다.*
- *patch 모듈에는 커널에 대해 원하는 수정 사항의 코드가 포함되어 있습니다.*
- *패치 모듈은 **livepatch** 커널 하위 시스템에 등록하고 원래 기능에 대한 정보를 제공하며 대체 기능에 대한 해당 포인터를 제공합니다. 커널 패치 모듈은 RPM으로 제공됩니다.*
- *이름 지정 규칙은 **kpatch_<kernel version>_<kpatch version>_<kpatch 릴리스>** 입니다. 이름의 "커널 버전" 부분에는 점과 대시 가 밑줄 로 교체되었습니다.*

***kpatch** 유틸리티*

패치 모듈을 관리하는 명령줄 유틸리티입니다.

***kpatch** 서비스*

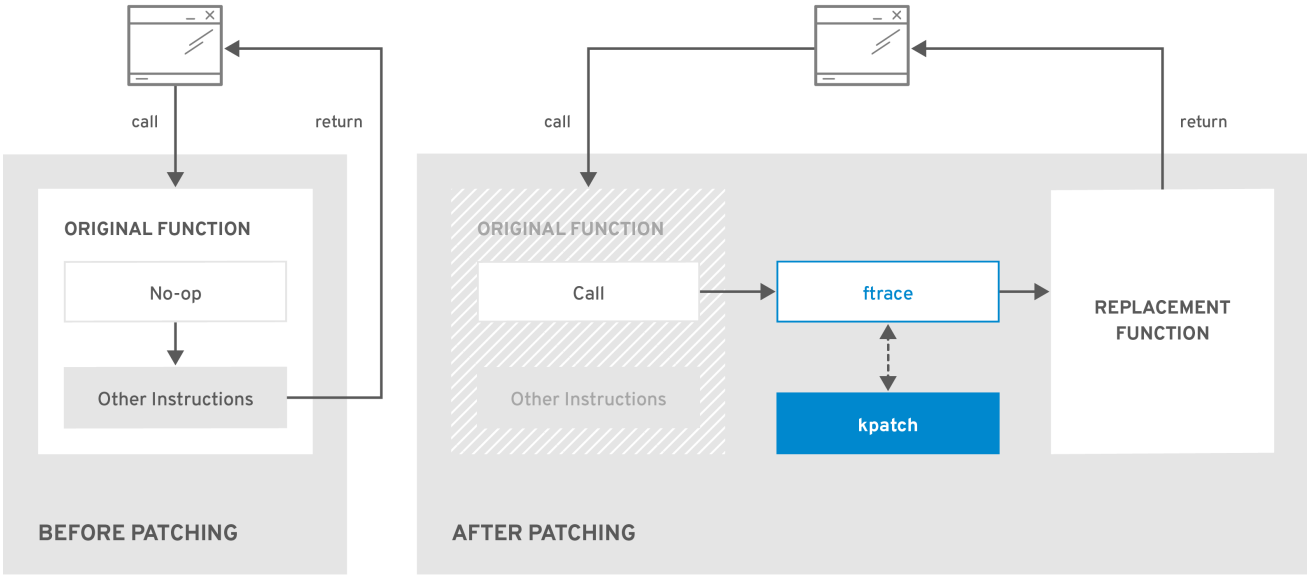
*multipath **.target**에 필요한 **systemd** 서비스입니다. 이 대상은 부팅 시 커널 패치 모듈을 로드합니다.*

6.5. 커널 라이브 패치 작동 방식

***kpatch** 커널 패치 솔루션에서는라이브 패치 커널 하위 시스템을 사용하여 이전 기능을 새 기능으로 리디렉션합니다. 라이브 커널 패치가 시스템에 적용되면 다음과 같은 일이 발생합니다.*

1. *커널 패치 모듈은 **/var/lib/kpatch/** 디렉토리에 복사되고 다음 부팅 시**systemd** 를 통해 커널에 재애플리케이션을 위해 등록됩니다.*
2. *kpatch 모듈은 실행 중인 커널에 로드되고 패치된 함수는 새 코드 메모리에 있는 위치에 대한 포인터와 함께 **ftrace** 메커니즘에 등록됩니다.*
3. *커널이 패치된 기능에 액세스하면 원래 기능을 무시하고 커널을 패치된 버전의 함수로 리디렉션하는 **ftrace** 메커니즘으로 리디렉션됩니다.*

그림 6.1. 커널 라이브 패치 작동 방식



6.6. 커널 라이브 패치 활성화

커널 패치 모듈은 패치되는 커널 버전과 관련된 RPM 패키지로 제공됩니다. 각 RPM 패키지는 시간이 지남에 따라 누적 업데이트됩니다.

다음 하위 섹션에서는 지정된 커널에 대한 모든 누적 라이브 패치 업데이트를 수신하는 방법을 설명합니다.

> *주의*
>
> *Red Hat은 Red Hat 지원 시스템에 적용되는 타사의 실시간 패치를 지원하지 않습니다.*

6.6.1. 라이브 패치 스트림 구독

이 절차에서는 특정 라이브 패치 패키지 설치에 대해 설명합니다. 이렇게 하면 지정된 커널의 라이브 패치 스트림을 구독하고 해당 커널에 대한 향후 누적 라이브 패치 업데이트를 모두 받을 수 있습니다.

> *주의*
>
> *라이브 패치는 누적되므로 지정된 커널에 배포되는 개별 패치를 선택할 수 없습니다.*

사전 요구 사항

- *root 권한*

절차

1. *선택적으로 커널 버전을 확인합니다.*

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

2. *커널 버전에 해당하는 라이브 패치 패키지를 검색합니다.*

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

3. *라이브 패치 패키지를 설치합니다.*

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

위의 명령은 해당 특정 커널에 대해서만 최신 누적 라이브 패치를 설치하고 적용합니다.

패키지 버전이 1-1 이상인 경우 라이브 패치 패키지에는 패치 모듈이 포함되어 있습니다. 이 경우 라이브 패치 패키지를 설치하는 동안 커널이 자동으로 패치됩니다.

*커널 patch 모듈은 향후 재부팅 중에 **systemd** 시스템 및 서비스 관리자가 로드할 **/var/lib/kpatch/** 디렉토리에도 설치됩니다.*

참고

*지정된 커널에서 사용 가능한 라이브 패치가 아직 없는 경우 빈 라이브 패치 패키지가 설치됩니다. 비어 있는 라이브 패치 패키지에는 0-0의 kpatch_version-kpatch_release 가 있습니다(예: **kpatch-patch-3_10_0-1062-0.el7.x86_64.rpm**). 빈 RPM을 설치하면 이후에 지정된 커널의 모든 라이브 패치를 사용할 수 있습니다.*

4. *선택적으로 커널이 패치되었는지 확인합니다.*

```
# 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)
...
```

*출력에서 커널 패치 모듈이 커널에 로드되었음을 보여줍니다. 이제 **kpatch-patch-3_10_0-1062-1.el7.x86_64.rpm** 패키지의 최신 수정 사항으로 패치됩니다.*

추가 리소스

- ***kpatch** 명령줄 유틸리티에 대한 자세한 내용은 **kpatch(1)** 매뉴얼 페이지를 참조하십시오.*
- *의 관련 섹션을 참조하십시오. System Administrator's Guide RHEL 7의 소프트웨어 패키지에 대한 자세한 내용은 다음을 참조하십시오.*

6.7. 커널 패치 모듈 업데이트

커널 패치 모듈이 RPM 패키지를 통해 전달 및 적용되므로 누적 커널 패치 모듈을 업데이트하는 것은 다른 RPM 패키지를 업데이트하는 것과 같습니다.

사전 요구 사항

- *root 권한*
- *6.6.1절. "라이브 패치 스트림 구독" 에 설명된 대로 실시간 패치 스트림에 시스템이 서브스크립션됩니다.*

절차

- *현재 커널의 새로운 누적 버전으로 업데이트합니다.*

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

위의 명령은 현재 실행 중인 커널에 사용할 수 있는 모든 업데이트를 자동으로 설치하고 적용합니다. 향후 릴리스된 누적 라이브 패치 포함.

- *또는 설치된 모든 커널 패치 모듈을 업데이트합니다.*

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

참고

*시스템이 동일한 커널로 재부팅되면 **kpatch.service** 서비스에 의해 커널이 자동으로 다시 패치됩니다.*

추가 리소스

- *소프트웨어 패키지 업데이트에 대한 자세한 내용은 관련 섹션을 참조하십시오. System Administrator's Guide.*

6.8. 커널 라이브 패치 비활성화

시스템 관리자가 Red Hat Enterprise Linux 커널 라이브 패치와 연결된 예기치 않은 부정적인 영향을 미치는 경우 이를 통해 메커니즘을 비활성화할 수 있습니다. 다음 섹션에서는 라이브 패치 솔루션을 비활성화하는 방법에 대해 설명합니다.

중요

현재 Red Hat은 시스템을 재부팅하지 않고 실시간 패치를 되돌리는 것을 지원하지 않습니다. 문제가 있는 경우 지원팀에 문의하십시오.

6.8.1. 실시간 패치 패키지 제거

다음 절차에서는 라이브 패치 패키지를 제거하여 Red Hat Enterprise Linux 커널 라이브 패치 솔루션을 비활성화하는 방법을 설명합니다.

사전 요구 사항

- *root 권한*
- *라이브 패치 패키지가 설치되어 있습니다.*

절차

1. *라이브 패치 패키지를 선택합니다.*

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

위의 예제 출력에는 설치한 라이브 패치 패키지가 나열됩니다.

2. 라이브 패치 패키지를 제거합니다.

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

실시간 패치 패키지가 제거되면 커널은 다음 재부팅까지 패치되지만 커널 패치 모듈은 디스크에서 제거됩니다. 다음 재부팅 후에는 해당 커널이 더 이상 패치되지 않습니다.

3. 시스템을 재부팅합니다.

4. 라이브 패치 패키지가 제거되었는지 확인합니다.

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

패키지가 성공적으로 제거된 경우 명령은 출력을 표시하지 않습니다.

5. 선택적으로 커널 실시간 패치 솔루션이 비활성화되어 있는지 확인합니다.

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

예제 출력에서는 커널이 패치되지 않았으며 현재 로드된 패치 모듈이 없기 때문에 실시간 패치 솔루션이 활성화되지 않음을 보여줍니다.

추가 리소스

- **kpatch** 명령줄 유틸리티에 대한 자세한 내용은**kpatch(1)** 매뉴얼 페이지를 참조하십시오.
- 소프트웨어 패키지 작업에 대한 자세한 내용은 관련 섹션을 참조하십시오. System Administrator's Guide.

6.8.2. 커널 패치 모듈 설치 제거

다음 절차에서는 Red Hat Enterprise Linux 커널 실시간 패치 솔루션이 후속 부팅에 커널 패치 모듈을 적용하지 않도록 하는 방법을 설명합니다.

사전 요구 사항

- root 권한
- 라이브 패치 패키지가 설치됩니다.
- 커널 패치 모듈이 설치 및 로드됩니다.

절차

1. 커널 패치 모듈을 선택합니다.

```
# 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. 선택한 커널 패치 모듈을 설치 제거합니다.

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

- 제거된 커널 패치 모듈이 계속 로드됩니다.

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

Installed patch modules:
<NO_RESULT>
```

선택한 모듈이 제거되면 다음 재부팅까지 커널이 패치되지만 커널 패치 모듈은 디스크에서 제거됩니다.

3. 시스템을 재부팅합니다.

4. 선택적으로 커널 패치 모듈이 제거되었는지 확인합니다.

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

위의 출력 예제에서는 로드되거나 설치된 커널 패치 모듈이 표시되어 커널이 패치되지 않고 커널 라이브 패치 솔루션이 활성화되지 않습니다.

추가 리소스

- **kpatch** 명령줄 유틸리티에 대한 자세한 내용은**kpatch(1)** 매뉴얼 페이지를 참조하십시오.

6.8.3. kpatch.service 비활성화

다음 절차에서는 Red Hat Enterprise Linux 커널 실시간 패치 솔루션이 모든 커널 패치 모듈을 후속 부팅에 전역적으로 적용하지 않도록 하는 방법을 설명합니다.

사전 요구 사항

- root 권한
- 라이브 패치 패키지가 설치됩니다.
- 커널 패치 모듈이 설치 및 로드됩니다.

절차

1. **kpatch.service** 가 활성화되어 있는지 확인합니다.

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

2. **kpatch.service** 를 비활성화합니다.

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

- 적용된 커널 패치 모듈이 계속 로드됩니다.

```
# 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. 시스템을 재부팅합니다.

4. 필요한 경우 **kpatch.service** 의 상태를 확인합니다.

```
# 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)
```

예제 출력에서는 **kpatch.service** 가 비활성화되었으며 실행되지 않음을 테스트합니다. 따라서 커널 실시간 패치 솔루션이 활성화되지 않습니다.

5. 커널 패치 모듈이 언로드되었는지 확인합니다.

```
# kpatch list
Loaded patch modules:

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

위의 출력 예제에서는 커널 패치 모듈이 계속 설치되어 있지만 커널은 패치되지 않음을 보여줍니다.

추가 리소스

- **kpatch** 명령줄 유틸리티에 대한 자세한 내용은**kpatch(1)** 매뉴얼 페이지를 참조하십시오.
- **systemd** 시스템 및 서비스 관리자, 장치 구성 파일, 해당 위치 및 전체**systemd** 장치 유형에 대한 자세한 내용은 의 관련 섹션을 참조하십시오. System Administrator's Guide.

7장. 커널 크래시 덤프 가이드

7.1. KDUMP 소개

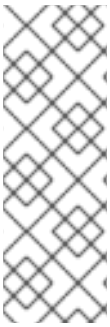
7.1.1. kdump 및 kexec 정보

kdump 는 크래시 덤프 메커니즘을 제공하는 서비스입니다. 이 서비스를 사용하면 분석을 위해 시스템 메모리의 내용을 저장할 수 있습니다.

kdump 는 **kexec** 시스템 호출을 사용하여 재부팅하지 않고 두 번째커널(커널 캡처)으로 부팅한 다음 충돌된 커널 메모리 (crash dump 또는vmcore)의 콘텐츠를 캡처하여 파일에 저장합니다. 두 번째 커널은 시스템 메모리의 예약된 부분에 있습니다.

중요

커널 크래시 덤프는 실패 시 사용할 수 있는 유일한 정보일 수 있으며, 비즈니스에 중요한 환경에서 이 데이터를 갖는 것의 중요성은 과소평가할 수 없습니다. Red Hat은 시스템 관리자가 일반 커널 업데이트 주기에서 **kexec-tools** 를 정기적으로 업데이트하고 테스트하는 것이 좋습니다. 이는 새로운 커널 기능을 구현할 때 특히 중요합니다.



참고

HP Watchdog 타이머(hpwdt) 드라이버는 RHEV 하이퍼바이저로 실행되는 HP 시스템에 사전 로드되므로 이러한 시스템은 NMI 워치독을 사용할 수 있습니다. kexec-tools-2.0.15-33.el7.x86_64 로 시작하는 업데이트된 kexec-tools 패키지는hpwdt 드라이버를 미리 로드했습니다.

kdump 커널에서 bnx2x 및 bmx2fc 드라이버가 블랙리스트로 지정되지 않으면 두 번째 커널에서 패닉 상태가 되고 덤프가 캡처되지 않습니다.

7.1.2. 메모리 요구 사항

kdump가 커널 크래시 덤프를 캡처하여 추가 분석을 위해 저장할 수 있으려면 캡처 커널에 대해 시스템 메모리의 일부를 영구적으로 예약해야 합니다. 예약된 경우 시스템 메모리의 이 부분은 기본 커널에서 사용할 수 없습니다.

메모리 요구 사항은 특정 시스템 매개변수에 따라 다릅니다. 주요 요인 중 하나는 시스템의 하드웨어 아키텍처입니다. 시스템 아키텍처의 정확한 이름(예: **x86_64**)을 검색하고 표준 출력에 출력하려면 쉘 프롬프트에서 다음 명령을 입력합니다.

```
uname -m
```

예약할 메모리 크기에 영향을 주는 또 다른 요소는 설치된 시스템 메모리의 총 양입니다. 예를 들어 x86_64 아키텍처에서 예약된 메모리 크기는 4KB의 RAM마다 160MB + 2비트입니다. 총 물리 메모리가 1TB인 시스템에서는 224MB(160MB + 64MB)를 의미합니다. 시스템 아키텍처 및 실제 메모리 양에 따라 kdump에 대한 전체 메모리 요구 사항 목록은 7.8.1절. "kdump에 대한 메모리 요구 사항"에서 참조하십시오.

많은 시스템에서 kdump는 필요한 메모리 양을 추정하고 자동으로 예약할 수 있습니다. 이 동작은 기본적으로 활성화되어 있지만 특정 양의 사용 가능한 메모리를 초과하는 시스템에서만 작동하며 시스템 아키텍처에 따라 다릅니다. 시스템 아키텍처에 따라 자동 메모리 예약에 대한 최소 요구 사항 목록은 7.8.2절. "자동 메모리 예약에 대한 최소 임계값" 을 참조하십시오.

시스템에 자동 할당이 작동하는 데 필요한 최소 메모리 양보다 적거나 사용 사례에 다른 값이 필요한 경우 수

동으로 예약된 메모리 양을 구성할 수 있습니다. 명령줄에서 이 작업을 수행하는 방법에 대한 자세한 내용은 7.2.2.1절. "메모리 사용량 구성" 을 참조하십시오. 그래픽 사용자 인터페이스에서 예약된 메모리 양을 구성하는 방법에 대한 자세한 내용은 7.2.3.1절. "메모리 사용량 구성" 을 참조하십시오.

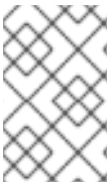
중요

자동 메모리 예약을 사용하는 경우에도 kdump 서비스를 설정한 후 설정을 테스트하는 것이 좋습니다. 구성을 테스트하는 방법에 대한 지침은 7.4절. "kdump 설정 테스트" 을 참조하십시오.

7.2. KDUMP 설치 및 구성

7.2.1. kdump 설치

대부분의 경우 **kdump** 서비스는 새로운 Red Hat Enterprise Linux 7 설치에 기본적으로 설치되고 활성화됩니다. The Anaconda 설치 프로그램은 그래픽 또는 텍스트 인터페이스를 사용하여 대화형 설치를 수행할 때 kdump 설정 화면을 제공합니다. 설치 프로그램 화면에는 **Kdump** 라는 제목이 있으며 기본설치 요약 화면에서 사용할 수 있으며 제한된 구성만 사용할 수 있습니다. kdump가 활성화되어 있는지 여부와 예약된 메모리 양만 선택할 수 있습니다. kdump의 메모리 요구 사항에 대한 정보는 7.8.1절. "kdump에 대한 메모리 요구 사항" 에서 확인할 수 있습니다. 설치 프로그램의 Kdump 구성 화면은 에 문서화되어 있습니다Red Hat Enterprise Linux 7 Installation Guide.



참고

이전 Red Hat Enterprise Linux 릴리스에서 kdump 설정을 사용할 수 있었습니다. Firstboot 설치가 완료된 후 자동으로 실행되고 시스템이 처음으로 재부팅된 유틸리티. Red Hat Enterprise Linux 7.1부터 kdump 설정이 설치 프로그램으로 이동되었습니다.

사용자 정의 Kickstart 설치와 같은 일부 설치 옵션은 기본적으로 kdump를 설치하거나 활성화할 필요가 없습니다. 시스템의 경우 이 경우 시스템에 해당하고 kdump를 추가로 설치하려는 경우 쉘 프롬프트에서 **root** 로 다음 명령을 실행합니다.

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

위의 명령은 시스템에 활성 서브스크립션 또는 가 포함된 사용자 정의 리포지터리가 있다고 가정하여 kdump 및 기타 필요한 모든 패키지를 설치할 수 있습니다. kexec-tools 시스템 아키텍처를 위한 패키지.

참고

kdump가 시스템에 설치되어 있는지 여부를 모르는 경우 **rpm** 을 사용하여 확인할 수 있습니다.

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

또한 위에서 설명한 명령을 사용하는 경우 그래픽 구성 툴을 사용할 수 있지만 기본적으로 설치되지 않습니다. 7.2.3절. "그래픽 사용자 인터페이스에서 kdump 구성" 에 설명된 이 유틸리티를 설치하려면**root** 로 다음 명령을 사용하십시오.

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

Red Hat Enterprise Linux 7에 새 패키지를 설치하는 방법에 대한 자세한 내용은. Yum 패키지 관리자, 참조 Red Hat Enterprise Linux 7 System Administrator's Guide

중요

Red Hat Enterprise Linux 7.4부터 **Intel IOMMU** 드라이버는 **kdump** 에서 지원됩니다. 버전 7.3 이하에서 커널을 실행하는 경우 **Intel IOMMU** 지원을 사용하지 않는 것이 좋습니다.

7.2.2. 명령줄에서 kdump 설정

7.2.2.1. 메모리 사용량 구성

kdump 커널용으로 예약된 메모리는 항상 시스템 부팅 중에 예약되어 있습니다. 즉, 시스템의 부트 로더 구성에 메모리 크기가 지정됩니다.

kdump 커널용으로 예약된 메모리를 지정하려면 **crashkernel=** 옵션을 필수 값으로 설정합니다. 예를 들어 128MB의 메모리를 예약하려면 다음을 사용합니다.

```
crashkernel=128M
```

AMD64 및 Intel 64 시스템 및 IBM Power Systems 서버의 **crashkernel=** 옵션을 변경하는 방법에 대한 자세한 내용을 참조하십시오. GRUB2 부트 로더 및 IBM Z에서 사용zipl3.1.1절. "커널 명령줄 매개변수 설정" 을 참조하십시오.

crashkernel= 옵션은 여러 가지 방법으로 정의할 수 있습니다.**auto** 값을 사용하면7.8.1절. "kdump에 대한 메모리 요구 사항" 에 설명된 지침에 따라 시스템의 총 메모리 크기에 따라 예약된 메모리를 자동으로 구성할 수 있습니다. 더 큰 메모리 시스템은, 운영 체제의 설정된 한도까지 **crashkernel=auto** 옵션으로 아키텍처에 따라 계산됩니다.

이 동작을 변경하려면 **auto** 값을 특정 메모리 양으로 바꿉니다.

crashkernel= 옵션은 작은 메모리 시스템에서 특히 유용할 수 있습니다. 예를 들어 128MB의 메모리를 예약하려면 다음을 사용합니다.

```
crashkernel=128M
```

설치된 메모리의 총 양에 따라 예약된 메모리의 양을 변수로 설정할 수도 있습니다. 변수 메모리 예약 구문은 **crashkernel= <range1> : <size1 > , <range2 > : <size2 >** 입니다. 예를 들어 다음과 같습니다.

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

위 예제에서는 총 시스템 메모리 크기가 512MB 이상이고 2GB보다 낮은 경우 64MB의 메모리를 예약합니다. 총 메모리 크기가 2GB를 초과하는 경우 대신 kdump용으로 128MB가 예약되어 있습니다.

일부 시스템은 정해진 오프셋으로 메모리를 예약해야 합니다. 오프셋이 설정된 경우 예약된 메모리가 시작됩니다. 예약된 메모리를 오프셋하려면 다음 구문을 사용합니다.

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

위의 예제에서는 kdump가 16MB(물리적 주소 0x01000000)에서 시작하는 128MB의 메모리를 예약함을 의미합니다. offset 매개 변수가 0으로 설정하거나 완전히 생략되면 kdump가 예약된 메모리를 자동으로 오프셋합니다. 이 구문은 위에서 설명한 대로 변수 메모리 예약을 설정할 때 사용할 수도 있습니다. 이 경우 오프셋은 항상 마지막입니다(예: **crashkernel=512M-2G:64M,2G-:128M@16M**).

7.2.2.2. kdump 대상 구성

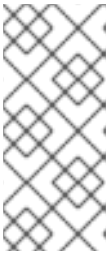
커널 충돌을 캡처하면 코어 덤프를 로컬 파일 시스템에 파일로 저장하거나 장치에 직접 작성되거나 **NFS**

(Network File System) 또는 **SSH** (Secure Shell) 프로토콜을 사용하여 네트워크를 통해 전송할 수 있습니다. 이러한 옵션 중 하나만 현재 설정할 수 있습니다. 기본 옵션은 **vmcore** 파일을 로컬 파일 시스템의 **/var/crash** 디렉터리에 저장하는 것입니다.

- 로컬 파일 시스템의 **/var/crash/** 디렉터리에 **vmcore** 파일을 저장하려면 **/etc/kdump.conf** 파일을 편집하고 경로를 지정합니다.

```
path /var/crash
```

옵션 경로 **/var/crash** 는 **kdump** 가 **vmcore** 파일을 저장하는 파일 시스템 경로를 나타냅니다.



참고

- **/etc/kdump.conf** 파일에서 덤프 대상을 지정하면 경로는 지정된 덤프 대상을기준으로 합니다.
- **/etc/kdump.conf** 파일에 덤프 대상을 지정하지 않으면 경로는 루트 디렉터리의 절대 경로를 나타냅니다.

현재 시스템에 마운트된 항목에 따라 덤프 대상 및 조정된 덤프 경로가 자동으로 수행됩니다.

예 7.1. kdump 대상 설정

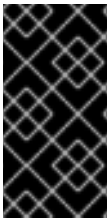
```
grep -v ^# etc/kdump.conf | grep -v ^$
ext4 /dev/mapper/vg00-varcrashvol
path /var/crash
core_collector makedumpfile -c --message-level 1 -d 31
```

여기서 덤프 대상은 (**ext4 /dev/mapper/vg00-varcrashvol**)로 지정되어 있으므로 **/var/crash** 에 마운트된 경로 옵션도 **/var/crash** 로 설정되므로 **kdump** 는 **/var/crash/var/crash** 디렉터리에 **vmcore** 파일을 저장합니다.

루트 권한으로 덤프 위치를 변경하려면 텍스트 편집기에서 **/etc/kdump.conf** 구성 파일을 열고 아래에 설명된 대로 옵션을 편집합니다.

코어 덤프를 저장할 로컬 디렉터리를 변경하려면 **#path /var/crash** 줄의 시작 부분에서 해시 기호("#")를 제거하고 해당 값을 원하는 디렉토리 경로로 바꿉니다.

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



중요

Red Hat Enterprise Linux 7에서는 kdump systemd 서비스가 시작될 때 **path** 지시문을 사용하여 **kdump** 대상으로 정의된 디렉터리가 있어야 합니다. 그렇지 않으면 서비스가 실패합니다. 이 동작은 서비스를 시작할 때 존재하지 않는 경우 디렉터리가 자동으로 생성된 Red Hat Enterprise Linux의 이전 릴리스와 다릅니다.

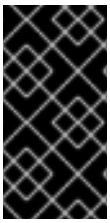
선택적으로 파일을 다른 파티션에 작성하려는 경우 **#ext4** 로 시작하는 줄 중 하나를 사용하여 동일한 절차를 따르십시오. 여기에서 장치 이름 (**#ext4 /dev/vg/lv_kdump** 라인), 파일 시스템 레이블 (**#ext4 LABEL=/boot** 라인) 또는 UUID (**#ext4 UUID=03138356-5e6-4ab3-b58e-27507ac41937** 라인)를 사용할 수 있습니다. 파일 시스템 유형 및 장치 이름, 레이블 또는 UUID를 원하는 값으로 변경합니다.

참고

UUID 값을 지정하는 올바른 구문은 **UUID="correct-uuid"** 및 **UUID=correct-uuid** 입니다.

예를 들어 다음과 같습니다.

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



중요

LABEL= 또는 **UUID=** 을 사용하여 스토리지 장치를 지정하는 것이 좋습니다. **/dev/sda3** 과 같은 디스크 장치 이름은 재부팅 시 일관되게 보장되지 않습니다. 참조 Red Hat Enterprise Linux 7 Storage Administration Guide 영구 디스크 장치 이름 지정에 대한 자세한 내용은 다음을 참조하십시오.

중요

s390x 하드웨어의 DASD로 덤프할 때는 계속하기 전에 덤프 장치를 **/etc/dasd.conf** 에 올바르게 지정해야 합니다.

덤프를 장치에 직접 작성하려면 **#raw /dev/vg/lv_kdump** 행의 시작 부분에서 해시 기호("#")를 제거하고 해당 값을 원하는 장치 이름으로 교체합니다. 예를 들어 다음과 같습니다.

```
raw /dev/sdb1
```

NFS 프로토콜을 사용하여 원격 머신에 덤프를 저장하려면**#nfs my.server.com:/export/tmp** 행의 시작 부분에서 해시 기호("#")를 제거하고 해당 값을 유효한 호스트 이름 및 디렉터리 경로로 교체합니다. 예를 들어 다음과 같습니다.

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

SSH 프로토콜을 사용하여 덤프를 원격 머신에 저장하려면**#ssh user@my.server.com** 행의 시작 부분에서 해시 기호("#")를 제거하고 해당 값을 유효한 사용자 이름 및 호스트 이름으로 교체합니다. SSH 키를 구성에도 포함하려면 **#sshkey /root/.ssh/kdump_id_rsa** 행의 시작 부분에서 해시 기호를 제거하고 덤프하려는 서버에 유효한 키 위치로 값을 변경합니다. 예를 들어 다음과 같습니다.

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

SSH 서버를 구성하고 키 기반 인증을 설정하는 방법에 대한 자세한 내용은 을 참조하십시오. Red Hat Enterprise Linux 7 System Administrator's Guide.

현재 지원되는 대상 및 유형별로 정렬된 지원되지 않는 대상의 전체 목록은 표 7.3. "지원되는 kdump 대상"를 참조하십시오.

7.2.2.3. 코어 수집기 구성

vmcore dump 파일의 크기를 줄이기 위해**kdump** 를 사용하면 외부 애플리케이션(코어 수집기)을 지정하여 데이터를 압축하고 선택적으로 모든 관련 정보를 남겨둘 수 있습니다. 현재는 완전히 지원되는 코어 수집기만 **makedumpfile** 입니다.

루트 로서 루트 수집기를 활성화하려면 텍스트 편집기에서**/etc/kdump.conf** 구성 파일을 열고 **#core_collector makedumpfile -l --message-level 1 -d 31** 행의 시작 부분에서 해시 기호("#")를 제거하고, 아래에 설명된 대로 명령행 옵션을 편집합니다.

덤프 파일 압축을 활성화하려면 **-l** 매개 변수를 추가합니다. 예를 들어 다음과 같습니다.

```
core_collector makedumpfile -l
```

덤프에서 특정 페이지를 제거하려면 **-d value** 매개변수를 추가합니다. 여기서 value 는 표 7.4. "지원되는 필터링 수준" 에 설명된 대로 생략하려는 페이지의 합계입니다. 예를 들어 0과 사용 가능한 페이지를 모두 제거하려면 다음을 사용합니다.

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

사용 가능한 전체 옵션 목록은 **makedumpfile(8)** 매뉴얼 페이지를 참조하십시오.

7.2.2.4. 기본 작업 구성

기본적으로 **kdump** 는 **kexec** 시스템 호출을 사용하여 재부팅하지 않고 두 번째커널(커널 캡처)으로 부팅한 다음 충돌된 커널 메모리(crash dump 또는 vmcore)의 내용을 캡처하여 파일에 저장합니다. 저장 후 **kdump** 는 시스템을 재부팅합니다.

그러나 **kdump** 가 7.2.2.2 절. "kdump 대상 구성" 에 지정된 대상 위치에 코어 덤프를 생성하지 못하면 **kdump** 는 vmcore 를 저장하지 않고 시스템을 재부팅합니다. 이 동작을 변경하려면 루트 로서 텍스트 편집기에서 **/etc/kdump.conf** 구성 파일을 열고 **#default** 쉘 행의 시작 부분에서 해시 기호(**"#"**)를 제거하고, 표 7.5. "지원되는 기본 작업" 에 설명된 대로 값을 원하는 작업으로 교체합니다.

예를 들어 다음과 같습니다.

```
default reboot
```

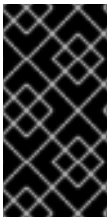
7.2.2.5. 서비스 활성화

부팅 시 **kdump** 데몬을 시작하려면 쉘 프롬프트에 **root** 로 다음을 입력합니다.

```
systemctl enable kdump.service
```

이를 통해 **multi-user.target** 에 대해 서비스를 사용할 수 있습니다. 마찬가지로 **systemctl disable kdump** 를 입력하면 **kdump** 가 비활성화됩니다. 현재 세션에서 서비스를 시작하려면 **root** 로 다음 명령을 사용하십시오.

```
systemctl start kdump.service
```



중요

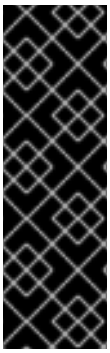
Red Hat Enterprise Linux 7에서 kdump 대상으로 정의된 디렉터리는 **kdump** systemd 서비스를 시작할 때 있어야 합니다. 그렇지 않으면 서비스가 실패합니다. 이 동작은 서비스를 시작할 때 존재하지 않는 경우 디렉터리가 자동으로 생성된 Red Hat Enterprise Linux의 이전 릴리스와 다릅니다.

systemd 및 서비스 구성에 대한 자세한 내용은 을 참조하십시오. Red Hat Enterprise Linux 7 System Administrator's Guide.

7.2.3. 그래픽 사용자 인터페이스에서 kdump 구성

시작 방법 Kernel Dump Configuration utility, 패널에서 → Other → **Kernel crash dumps** 선택하거나 쉘 프롬프트에서 **system-config-kdump** 를 입력합니다. 그러면 그림 7.1. "기본 설정" 에 창이 표시됩니다.

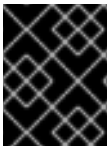
유틸리티를 사용하면 부팅 시 서비스 시작을 활성화하거나 비활성화할 뿐만 아니라 **kdump** 를 구성할 수 있습니다. 완료되면 적용을 클릭하여 변경 사항을 저장합니다. 이미 인증되어 있지 않으면 슈퍼유저 암호를 입력합니다. 유틸리티는 구성 변경 사항을 적용하려면 시스템을 재부팅해야 한다는 알림을 제공합니다.



중요

Enforcing 모드에서 **SELinux** 가 실행되는 IBM Z 또는 PowerPC 시스템에서 커널 Dump Configuration 유틸리티를 시작하기 전에 kdumpgui_run_bootloader 부울을 활성화해야 합니다. 이 부울을 사용하면 system-config-kdump가 bootloader_t SELinux 도메인에서 부트 로더를 실행할 수 있습니다. 부울을 영구적으로 활성화하려면 root로 다음 명령을 실행하십시오.

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



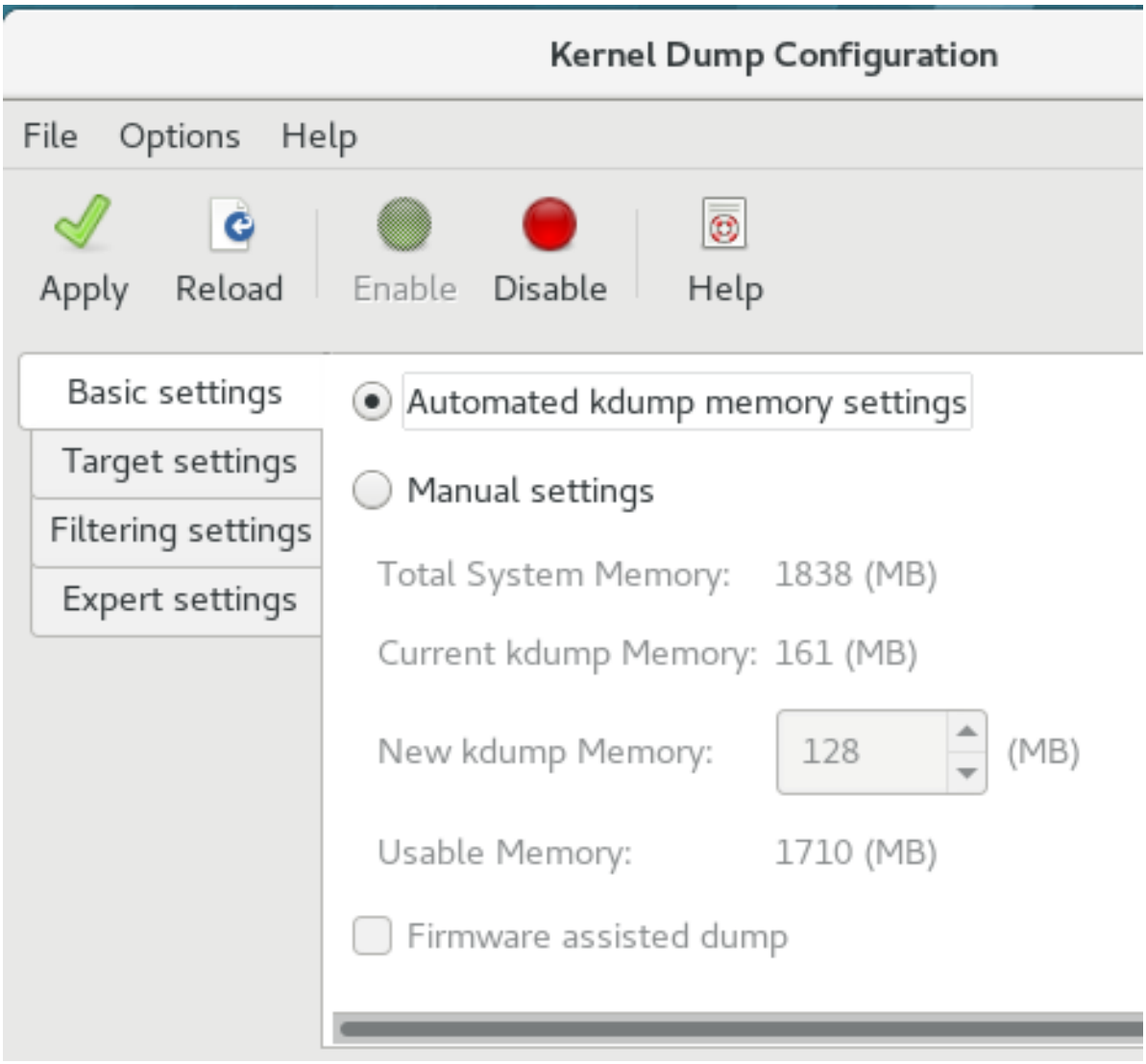
중요

s390x 하드웨어의 DASD로 덤프할 때는 계속하기 전에 덤프 장치를 **/etc/dasd.conf** 에 올바르게 지정해야 합니다.

7.2.3.1. 메모리 사용량 구성

기본 설정 탭을 사용하면 **kdump** 커널에 예약된 메모리 양을 구성할 수 있습니다. 이렇게 하려면 **Manual settings** (수동 설정) 라디오 버튼을 선택하고 **New kdump Memory** 필드 옆에 있는 위쪽 및 아래쪽 화살표 버튼을 클릭하여 예약할 메모리 양을 늘리거나 줄입니다. **Usable Memory** (원하는 메모리) 필드가 변경되어 시스템에서 사용할 수 있는 나머지 메모리가 표시됩니다. kdump의 메모리 요구 사항에 대한 자세한 내용은 7.1.2절. "메모리 요구 사항" 를 참조하십시오.

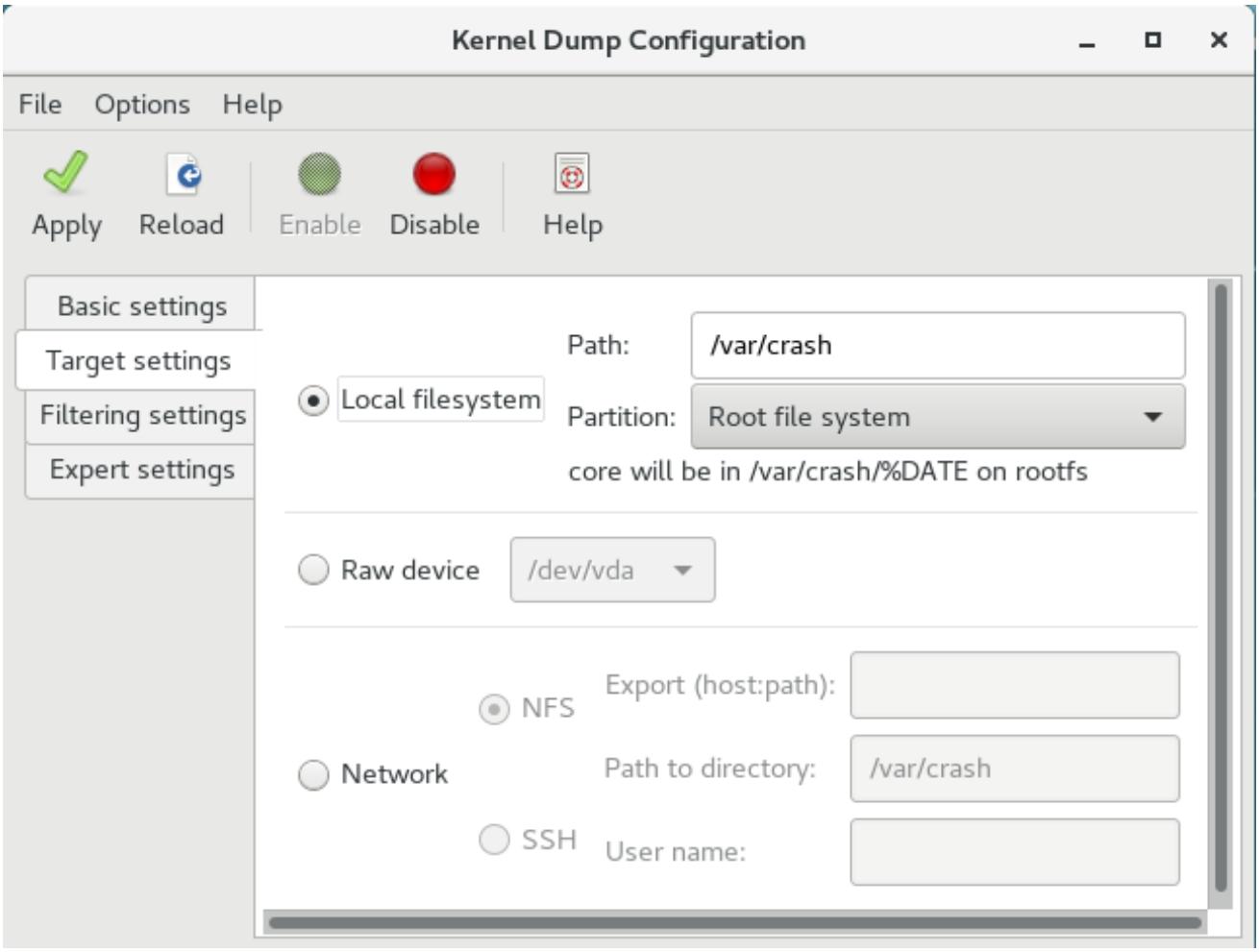
그림 7.1. 기본 설정



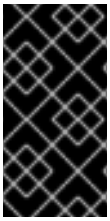
7.2.3.2. kdump 대상 구성

*대상 설정 탭에서 **vmcore** 덤프의 대상 위치를 지정할 수 있습니다. 덤프는 로컬 파일 시스템에 파일로 저장되거나 장치에 직접 작성되거나 **NFS** (Network File System) 또는 **SSH** (Secure Shell) 프로토콜을 사용하여 네트워크를 통해 전송할 수 있습니다.*

그림 7.2. 대상 설정



덤프를 로컬 파일 시스템에 저장하려면 로컬 파일 시스템 라디오 버튼을 선택합니다. 필요한 경우 **Partition** *드롭다운 목록에서 다른 파티션을 선택하고* **Path** *필드를 사용하여 대상 디렉터리를 선택하여 설정을 사용자 지정할 수 있습니다.*



중요

Red Hat Enterprise Linux 7에서 kdump 대상으로 정의된 디렉터리는 **kdump** *systemd 서비스를 시작할 때 있어야 합니다. 그렇지 않으면 서비스가 실패합니다. 이 동작은 서비스를 시작할 때 존재하지 않는 경우 디렉터리가 자동으로 생성된 Red Hat Enterprise Linux의 이전 릴리스와 다릅니다.*

덤프를 장치에 직접 작성하려면 원시 장치 라디오 버튼을 선택하고 드롭다운 목록에서 원하는 대상 장치를 선택합니다.

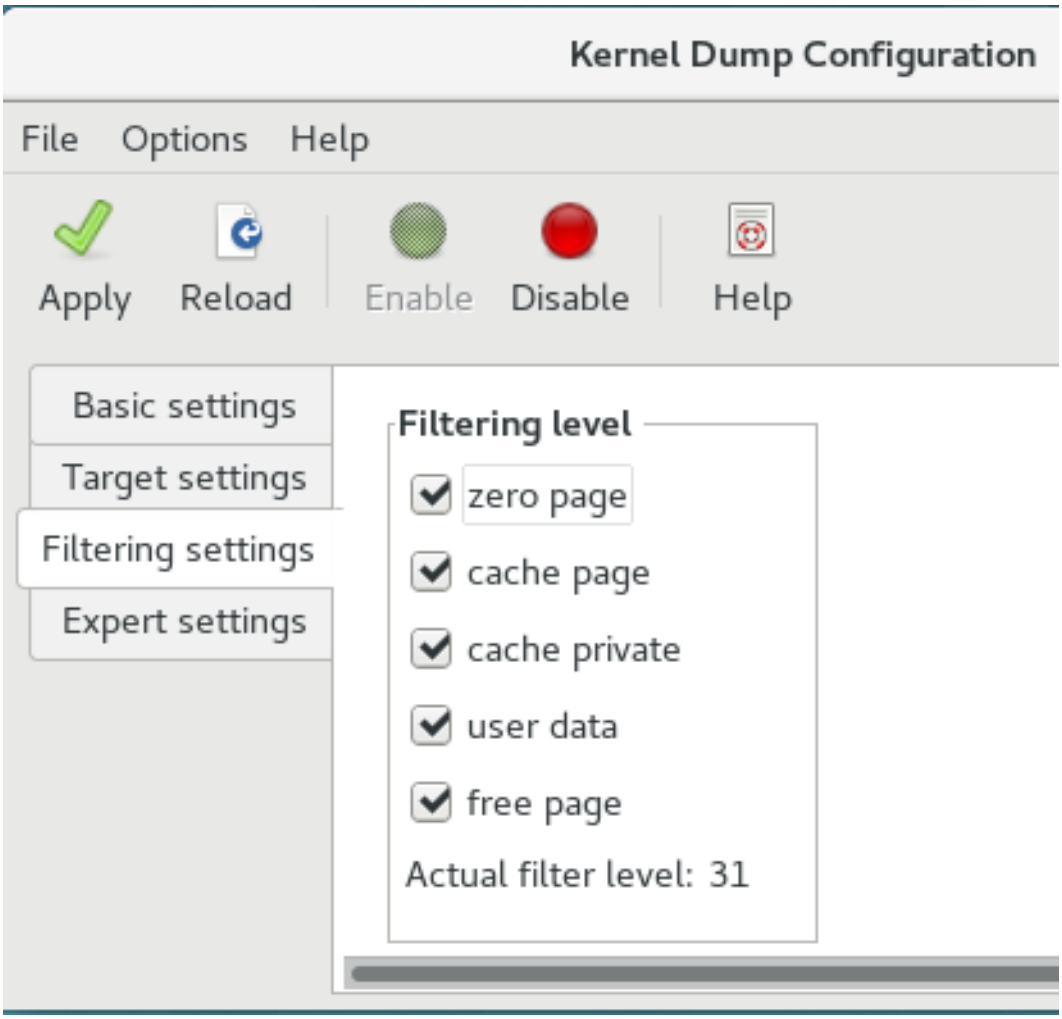
네트워크 연결을 통해 원격 시스템으로 덤프를 보내려면 네트워크 라디오 버튼을 선택합니다. **NFS** *프로토콜을 사용하려면* **NFS** *라디오 버튼을 선택하고서버 이름과* **Path to directory** *필드를 작성합니다.* **SSH** *프로토콜을 사용하려면* **SSH** *라디오 버튼을 선택하고 각각서버 이름,경로, 디렉터리, 사용자 이름 필드를 원격 서버 주소, 대상 디렉터리, 유효한 사용자 이름을 입력합니다.*

SSH 서버를 구성하고 키 기반 인증을 설정하는 방법에 대한 자세한 내용은 을 참조하십시오. Red Hat Enterprise Linux 7 System Administrator's Guide. 현재 지원되는 대상의 전체 목록은표 7.3. "지원되는 kdump 대상" 를 참조하십시오.

7.2.3.3. 코어 수집기 구성

*필터링 설정 탭에서 **vmcore** dump의 필터링 수준을 선택할 수 있습니다.*

그림 7.3. 설정 필터링



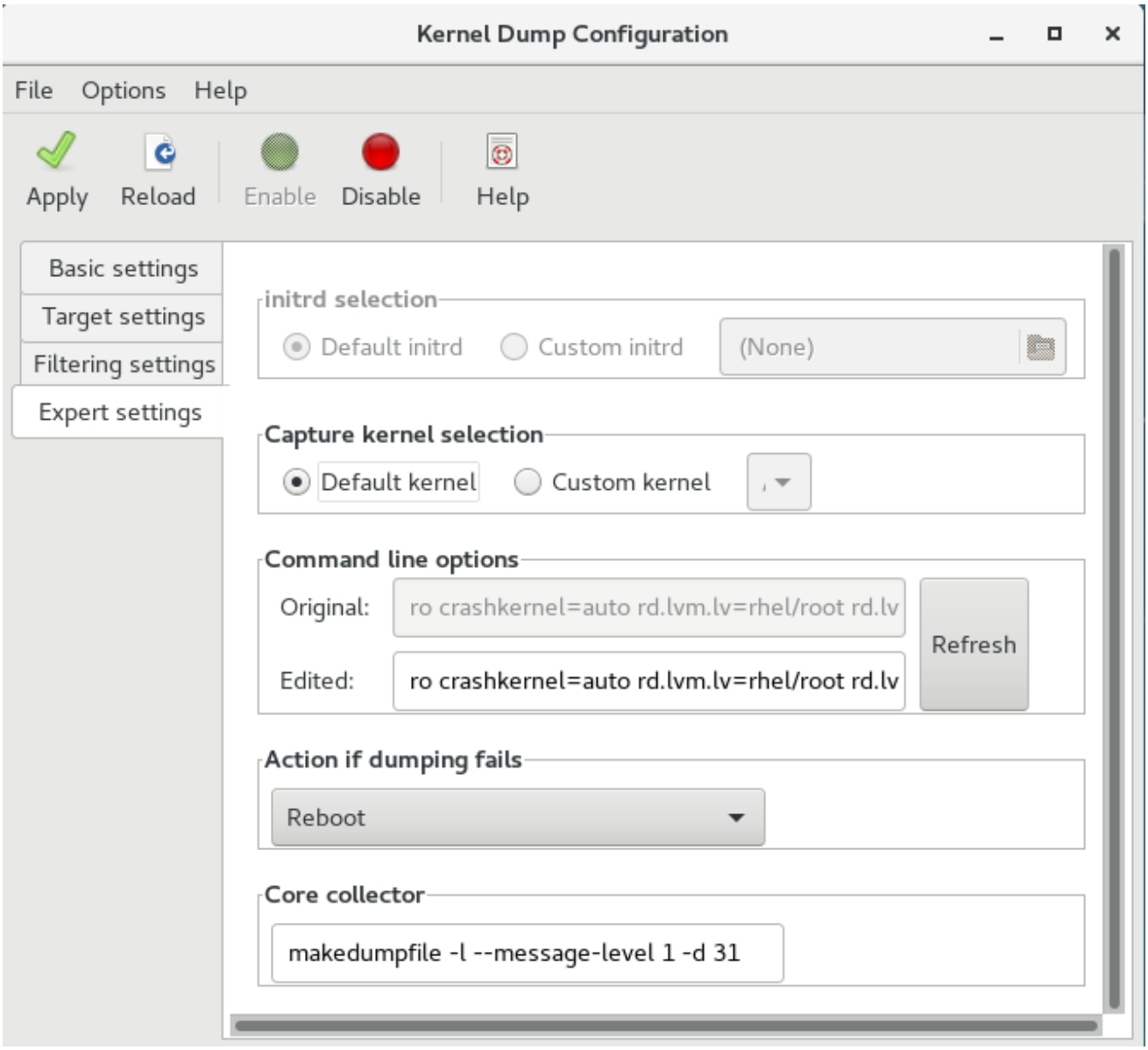
제로 페이지,캐시 페이지,캐시 개인 데이터,사용자 데이터 또는 덤프에서 사용 가능한페이지를 제외하려면 적절한 레이블 옆에 있는 확인란을 선택합니다.

7.2.3.4. 기본 작업 구성

***kdump** 가 코어 덤프를 생성하지 못할 때 수행해야 하는 작업을 선택하려면 덤프가실패한 경우 작업에서 적절한 옵션을 선택합니다. 사용 가능한 옵션은 다음과 같습니다.*

- *rootfs로 덤프하고 재부팅 시도에서 코어를 로컬에 저장한 다음 시스템을 재부팅합니다.*
- *시스템을 재부팅 하는 기본 작업 재부팅*
- *쉘을 시작하여 활성 쉘 프롬프트가 있는 사용자를 제공합니다.*
- *시스템을 중단하는 것을 중지합니다.*
- *시스템 전원을 끄려면 전원 끄기*

그림 7.4. 설정 필터링



makedumpfile 코어 수집기에 전달되는 옵션을 사용자 지정하려면**Core** 수집기 텍스트 필드를 편집합니다. 자세한 내용은 7.2.2.3절. "코어 수집기 구성" 을 참조하십시오.

7.2.3.5. 서비스 활성화

부팅 시 **kdump** 서비스를 시작하려면 도구 모음에서**Enable** 버튼을 클릭한 다음**Apply** 버튼을 클릭합니다. 이를 통해 **multi-user.target** 에 대한 서비스를 활성화 및 활성화합니다.**Disable** 버튼을 클릭하고**Apply** 버튼을 클릭하여 서비스를 즉시 비활성화합니다.

중요

Red Hat Enterprise Linux 7에서 kdump 대상으로 정의된 디렉터리는 **kdump** systemd 서비스를 시작할 때 있어야 합니다. 그렇지 않으면 서비스가 실패합니다. 이 동작은 서비스를 시작할 때 존재하지 않는 경우 디렉터리가 자동으로 생성된 Red Hat Enterprise Linux의 이전 릴리스와 다릅니다.

systemd 대상 및 일반적인 서비스 구성에 대한 자세한 내용은 을 참조하십시오. *Red Hat Enterprise Linux 7 System Administrator's Guide*.

7.3. KDUMP용 커널 드라이버 블랙리스트 지정

*커널 드라이버의 블랙리스트 지정은 커널 드라이버가 로드되고 사용되지 않도록 하는 메커니즘입니다. **/etc/sysconfig/kdump** 파일에 드라이버를 추가하여 kdump **initramfs** 가 블랙리스트 모듈을 로드하지 못하도록 합니다.*

*커널 드라이버를 블랙리스트로 지정하면 **oom** 킬러 또는 기타 충돌 커널 오류가 발생하지 않습니다. 커널 드라이버를 블랙리스트로 지정하려면 **/etc/sysconfig/kdump** 파일에서 **KDUMP_COMMANDLINE_APPEND=** 변수를 업데이트하고 다음 블랙리스트 옵션 중 하나를 지정할 수 있습니다.*

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

절차

1. *블랙리스트에 있는 커널 모듈을 선택합니다.*

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

   ***lsmod** 명령은 현재 실행 중인 커널에 로드된 모듈 목록을 표시합니다.*

2. ***/etc/sysconfig/kdump** 파일에서 **KDUMP_COMMANDLINE_APPEND=** 행을 다음과 같이 업데이트합니다.*

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

3. ***/etc/sysconfig/kdump** 파일에서 **KDUMP_COMMANDLINE_APPEND=** 행을 다음과 같이 업데이트할 수도 있습니다.*

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

4. ***kdump** 서비스를 다시 시작합니다.*

```
$ systemctl restart kdump
```

7.4. KDUMP 설정 테스트

주의

아래 명령으로 인해 커널이 충돌합니다. 이러한 단계를 수행할 때는 주의해야 하며 프로덕션 시스템에서는 사용하지 마십시오.

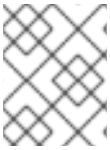
*설정을 테스트하려면 **kdump** 가 활성화된 시스템을 재부팅하고 서비스가 실행 중인지 확인합니다.*

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

쉘 프롬프트에 다음 명령을 입력합니다.

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

*이렇게 하면 Linux 커널이 충돌하게 되며 주소**-YYYY-MM-DD-HH:MM:SS/vmcore** 파일이 구성에서 선택한 위치에 복사됩니다(즉, 기본적으로 **/var/crash/** 로 변경).*



참고

이 작업을 사용하여 구성의 유효성을 확인하는 것 외에도 대표 테스트 로드에서 수행하는 경우 크래시 덤프에 걸리는 시간을 기록할 수도 있습니다.

7.4.1. 추가 리소스

7.4.1.1. 설치된 문서

- *kdump.conf(5) - 사용 가능한 옵션에 대한 전체 문서가 포함된**/etc/kdump.conf** 설정 파일의 수동 페이지입니다.*
- *zipl.conf(5) - **/etc/zipl.conf** 구성 파일의 설명서 페이지.*
- *zipl(8) - IBM Z용 **zipl** 부트 로더 유틸리티의 설명서 페이지.*
- *makedumpfile(8) - **makedumpfile** 코어 수집기의 수동 페이지.*
- *kexec(8) - 도움말 페이지kexec.*
- *crash(8) - 의 도움말 페이지crash 유틸리티.*
- ***/usr/share/doc/kexec-tools-**버전**/kexec-kdump-howto.txt - kdump** 및 개요kexec 설치 및 사용.*

7.4.1.2. 온라인 문서

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

***kexec** 및 **kdump** 설정에 대한 Red Hat 지식베이스 문서입니다.*

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

*지원되는 **kdump** 대상에 대한 Red Hat 지식베이스 문서*

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

The crash 유틸리티 Git 리포지토리.

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

The GRUB2 부트 로더 홈페이지 및 설명서.

7.5. 펌웨어에서 덤프 메커니즘 지원

7.5.1. 펌웨어에서 지원하는 덤프의 경우

kexec 및 **kdump** 메커니즘은 AMD64 및 Intel 64 시스템의 코어 덤프를 캡처하는 신뢰할 수 있고 검증된 방법입니다. 그러나 더 긴 내역, 특히 미니 및 메인프레임 시스템이 있는 일부 하드웨어를 사용하면 온보드 펌웨어를 활용하여 메모리 영역을 분리하고 충돌 분석에 중요한 데이터의 실수로 덮어 쓰기를 방지할 수 있습니다.

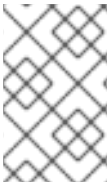
이 장에서는 덤프 방법을 지원하는 일부 펌웨어와 Red Hat Enterprise Linux와의 통합 방법에 대해 설명합니다.

7.5.2. IBM PowerPC 하드웨어에서 fadump 사용

펌웨어 지원 덤프(**fadump**)는 IBM PowerPC LPARS에서 사용 가능한 **kexec-kdump** 의 안정적인 대안입니다. PCI 및 I/O 장치가 있는 완전 재설정 시스템에서 vmcore를 캡처합니다. 이 메커니즘은 펌웨어를 사용하여 충돌이 발생할 경우 메모리를 보존하지만 **kdump** 사용자 공간 스크립트를 사용하여 vmcore를 저장합니다.

이를 위해 **fadump** 는 시스템 펌웨어와의 충돌 발생 시 유지해야 하는 메모리 영역을 등록합니다. 이러한 리전은 부팅 메모리, 시스템 레지스터 및 하드웨어 페이지 테이블 항목(PTE)을 제외한 모든 시스템 메모리 콘텐츠로 구성됩니다.

상기에 대한 보다 자세한 내용은 fadump 하드웨어를 재설정하는 PowerPC 특정 방법을 포함한 메커니즘은 **/usr/share/doc/kexec-tools-X.y.z/fadump-howto.txt** 를 참조하십시오. kexec-tools 시스템에 설치되어 있어야 합니다.



참고

메모리 영역이 보존되지 않고 부팅 메모리 라고 알려진 것은 충돌 이벤트 후 커널을 성공적으로 부팅하는 데 필요한 RAM의 양입니다. 기본적으로 부팅 메모리 크기는 총 시스템 RAM의 256MB 또는 5 %이며 이는 더 큽니다.

kexec-initiated 이벤트와 달리 **fadump** 프로세스는 프로덕션 커널을 사용하여 크래시 덤프를 복구합니다. 충돌 후 부팅 시 PowerPC 하드웨어는 장치 노드 **/proc/device-tree/rtas/ibm,kernel-dump** 를 사용할 수 있도록 fadump-aware **kdump** 스크립트에서 vmcore를 저장하도록 확인했습니다. 이 작업이 완료되면 시스템이 정상적으로 재부팅됩니다.

fadump 활성화

1. 7.2절. "kdump 설치 및 구성" 에 설명된 대로 **kdump** 를 설치하고 구성합니다.

2. **/etc/default/grub** 의 **GRUB_CMDLINE_LINUX** 행에 **fadump=on** 을 추가합니다.

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

3. *(선택 사항) 기본값을 수락하지 않고 예약된 부팅 메모리를 지정하려면* **crashkernel=xxM** *을* **GRUB_CMDLINE_LINUX** *로 구성합니다. 여기서* **xx** *는 메가바이트에 필요한 메모리 양입니다.*

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

중요

모든 부팅 구성 옵션과 마찬가지로 필요한 구성을 테스트하는 것이 좋습니다. 크래시 커널에서 부팅할 때 **OOM(Out of Memory)** *오류가 발생하는 경우 크래시 커널이 정상적으로 부팅할 수 있을 때까지* **crashkernel=** *에 지정된 값을 늘립니다. 이 경우 일부 시험 및 오류가 필요할 수 있습니다.*

7.5.3. IBM Z에서 지원되는 펌웨어

***IBM Z**에는 두 개의 펌웨어가 지원됩니다. 독립 실행형* **Dump** *와* **VMDUMP** *입니다.*

***The kdump** 인프라는 이러한 시스템에서 지원 및 활용되고* **Red Hat Enterprise Linux**의 구성에는 *7.2절. "kdump 설치 및 구성" 에 설명되어 있습니다. 그러나* **IBM Z** *하드웨어에서 제공하는 펌웨어 지원 방법 중 하나를 사용할 때 몇 가지 이점이 있습니다.*

***Stand-alone Dump(SADMP)** 메커니즘은 시스템 콘솔에서 시작 및 제어되며* **IPL** *부팅 가능 장치에 저장해야 합니다.*

***SADMP**와 비슷한 경우는* **VMDUMP**입니다. *이 툴은 시스템 콘솔에서 시작되지만, 하드웨어에서 결과 덤프를 검색하여 분석을 위해 시스템에 복사하는 메커니즘이 있습니다.*

이러한 방법의 한 가지 장점은 (및 다른 하드웨어 기반 덤프 메커니즘과 유사하게) **Early Boot** *단계에서 시스템의 상태를 캡처하는 기능입니다 (* **kdump** *서비스가 시작되기 전에)*

***VMDUMP**에는* **Red Hat Enterprise Linux** *시스템으로 덤프 파일을 수신하는 메커니즘이 포함되어 있지만* **SADMP**와 **VMDUMP**의 *구성 및 제어는* **IBM Z Hardware** *콘솔에서 모두 관리됩니다.*

***IBM**은* **SADMP**에 *대해 자세히 설명하고 독립 실행형 덤프 프로그램 문서 및* **VMDUMP** *문서의* **VMDUMP**에 *대해 자세히 설명합니다.*

IBM에는 Red Hat Enterprise Linux에서 Dump Tools on Red Hat Enterprise Linux 문서 사용 시 Red Hat Enterprise Linux 7의 덤프 툴 사용에 대한 문서도 포함되어 있습니다.

7.5.4. Fujitsu PRIMEQUEST 시스템에서 sadump 사용

Fujitsu sadump 메커니즘은 이벤트에서 대체 덤프 캡처를 제공하도록 설계되었습니다. kdump 를 성공적으로 완료할 수 없습니다.

sadump 프로세스는 시스템 ManageMent Board(MMB) 인터페이스에서 수동으로 호출됩니다.

이 시스템을 사용하여 X86_64 서버에 대해 kdump를 정상적으로 구성한 다음 다음 추가 단계를 수행하여 sadump 를 활성화합니다.

/etc/sysctl.conf 에서 다음 행을 추가하거나 편집하여 확인합니다. kdump sadump 에 대해 예상대로 시작됩니다.

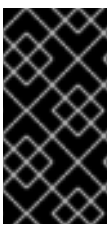
```
kernel.panic=0
kernel.unknown_nmi_panic=1
```

위 항목 외에도 /etc/kdump.conf 에 일부 옵션을 추가하여 해당 상태를 확인해야 합니다. kdump sadump에 대해 올바르게 작동합니다.

특히 다음 사항을 확인하십시오. kdump시스템이 재부팅되지 않습니다. 시스템이 재부팅되는 경우 kdump 는 코어를 저장하지 못했습니다. 그러면 호출할 기회가 없습니다. sadump.

이를 위해 /etc/kdump.conf 에서 기본 작업을 halt 또는 shell 로 설정합니다.

```
default shell
```



중요

하드웨어 구성에 대한 자세한 내용은 sadumpFUJITSU Server PRIMEQUEST 2000 설치 설명서를 참조하십시오.

7.6. 코어 덤프 분석

시스템 충돌의 원인을 확인하려면 다음을 사용하십시오. **crash** GDB(GNU Debugger)와 매우 유사한 대화형 프롬프트를 제공하는 유틸리티. 이 유틸리티를 사용하면 **netdump,diskdump,xendump** 또는 **kdump** 로 생성된 코어 덤프뿐만 아니라 실행 중인 **Linux** 시스템을 대화형으로 분석할 수 있습니다.

7.6.1. 크래시 유틸리티 설치

설치 방법 **crash** 도구를 분석하고 쉘 프롬프트에서 루트로 다음 명령을 실행합니다.

```
yum install crash
```

기타 등등 (**In addition to crash**또한 설치해야합니다. **kernel-debuginfo** 덤프 분석에 필요한 데이터를 제공하는 실행 중인 커널에 해당하는 패키지입니다. 설치 방법 **kernel-debuginfo debuginfo-install** 명령을 **root** 로 사용합니다.

```
debuginfo-install kernel
```

Red Hat Enterprise Linux에 새 패키지를 설치하는 방법에 대한 자세한 내용은. **Yum** 패키지 관리자, 참조 **Red Hat Enterprise Linux 7 System Administrator's Guide**.

7.6.2. 크래시 유틸리티 실행

유틸리티를 시작하려면 쉘 프롬프트에서 다음 양식에 명령을 입력합니다.

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

kdump 에서 캡처한 것과 동일한 **& lt;kernel>** 버전을 사용합니다. 현재 실행 중인 커널을 확인하려면 **uname -r** 명령을 사용합니다.

예 7.2. 크래시 유틸리티 실행

```
~]# 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. 메시지 버퍼 표시

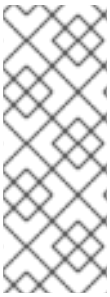
커널 메시지 버퍼를 표시하려면 대화형 프롬프트에서 **log** 명령을 입력합니다.

예 7.3. 커널 메시지 버퍼 표시

```
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
```

명령 사용법에 대한 자세한 내용은 도움말 로그 를 입력합니다.



참고

*커널 메시지 버퍼에는 시스템 충돌에 대한 가장 중요한 정보가 포함되어 있으므로 항상 **vmcore-dmesg.txt** 파일에서 먼저 덤프됩니다. 이는 예를 들어 대상 위치의 공간이 부족하여 전체 **vmcore** 파일을 가져오려고 할 때 유용합니다. 기본적으로 **vmcore-dmesg.txt** 는 **/var/crash/** 디렉터리에 있습니다.*

7.6.4. 역추적 표시

*커널 스택 추적을 표시하려면 대화형 프롬프트에서 **bt** 명령을 입력합니다. **bt < pid>** 를 사용하여 단일 프로세스의 역추적을 표시할 수 있습니다.*

예 7.4. 커널 스택 추적 표시

```
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: fffffffda  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
```

*명령 사용법에 대한 자세한 내용은 **help bt** 를 입력합니다.*

7.6.5. 프로세스 상태 표시

*시스템의 프로세스 상태를 표시하려면 대화형 프롬프트에서 **ps** 명령을 입력합니다. **ps < pid>** 를 사용하여 단일 프로세스의 상태를 표시할 수 있습니다.*

예 7.5. 시스템에 프로세스 상태 표시

```
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
```

*명령 사용법에 대한 자세한 내용은 **help ps** 를 입력합니다.*

7.6.6. 가상 메모리 정보 표시

*기본 가상 메모리 정보를 표시하려면 대화형 프롬프트에서 **vm** 명령을 입력합니다. **vm < pid>** 를 사용하여 단일 프로세스에 정보를 표시할 수 있습니다.*

예 7.6. 현재 컨텍스트의 가상 메모리 정보 표시

```
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 ...
```

*명령 사용법에 대한 자세한 내용은 **help vm** 를 입력합니다.*

7.6.7. 열려 있는 파일 표시

*열려 있는 파일에 대한 정보를 표시하려면 대화형 프롬프트에서 **files** 명령을 입력합니다. **< pid>**파일을 사용하여 선택한 하나의 프로세스에서 열린 파일을 표시할 수 있습니다.*

예 7.7. 현재 컨텍스트의 열려 있는 파일에 대한 정보 표시

```
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
```

명령 사용법에 대한 자세한 내용은 도움말 파일을 입력합니다.

7.6.8. 유틸리티 종료

대화형 프롬프트를 종료하고 종료합니다. **crashexit** 또는 **q** 를 입력합니다.

> 예 7.8. 크래시 유틸리티 종료
>
> ```
> crash> exit
> ~]#
> ```

7.7. 자주하는 질문

클러스터형 환경에서 Kdump를 사용하려면 어떤 고려 사항을 수행해야 합니까?

*RHEL 6, 7 고가용성 애드온과 함께 사용할 수 있도록 kdump를 구성하려면 어떻게 해야 합니까?*에서는 고가용성 애드온 을 사용하여 시스템 관리자가 사용할 수 있는 옵션을 보여줍니다.

초기 부팅 과정에서 kdump가 실패합니다. 부팅 로그를 캡처하는 방법은 무엇입니까?

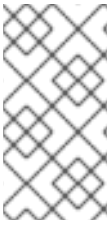
두 번째 커널을 부팅하는 데 문제가 있는 경우 초기 부팅 로그를 검토해야 합니다. 이러한 부팅은 영향을 받는 시스템에 직렬 콘솔을 활성화하여 얻을 수 있습니다.

RHEL7에서 직렬 콘솔을 설정하려면 어떻게 해야 합니까? 는 초기 부팅 메시지에 대한 액세스를 활성화하는 데 필요한 구성을 보여줍니다.

디버깅을 위해 makedumpfile에서 메시징을 늘리려면 어떻게 해야 합니까?

makedumpfile 이 실패하는 경우 로그 수준을 높여 무엇이 잘못되는지 파악해야 합니다. 이는 덤프 수준을 설정하는 것과 다르며 **/etc/kdump.conf** 를 편집하고 **core_collector** 행 항목에서 **dumpfile**을 만들기 위해 **message_level** 옵션을 늘려서 달성됩니다.

기본적으로 **makedumpfile** 은 출력 출력 내용을 진행률 표시로 제한하는 **level 1**로 설정됩니다. 이 메시지 수준을 **31**로 설정하여 모든 디버깅 정보를 활성화합니다. 메시지 수준 **31**은 진행 상태 표시, 일반적인 메시지, 오류 메시지, 디버그 메시지 및 보고서 메시지에 대한 세부 정보를 출력합니다.



참고

메시지 수준 옵션에 대한 자세한 내용은 **makedumpfile(8)** 매뉴얼 페이지를 참조하십시오.

설정된 경우 **core_collector** 구성 줄이 다음과 유사하게 표시되는지 확인합니다.

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

Dracut을 어떻게 디버깅할 수 있습니까?

때로는 **dracut** 이 **initramfs**를 빌드하지 못할 수 있습니다. 이 경우 문제를 격리하려면 **dracut** 에서 로그 수준을 늘립니다.

/etc/kdump.conf 를 편집하고 필요한 다른 **dracut** 인수 외에 **-L 5** 옵션을 포함하도록 **dracut_args** 행을 변경합니다.

dracut_args 에 다른 옵션이 구성되어 있지 않은 경우 결과는 다음과 유사합니다.

```
dracut_args -L 5
```

가상 머신에 사용할 수 있는 덤프 방법은 무엇입니까?

대부분의 경우, **kdump** 메커니즘은 충돌 또는 패닉 발생 후 시스템에서 메모리 덤프를 얻는 데 충분합니다. 이는 베어 메탈에 설치와 동일한 방식으로 설정할 수 있습니다.

그러나 크래시 덤프를 얻으려면 하이퍼바이저와 직접 작업해야 하는 경우도 있습니다. 이 작업을 수행하기 위해 **libvirt** 에서 사용할 수 있는 두 가지 메커니즘은 **pvpanic** 및 **virsh dump** 입니다. 이러한 두 가지 방법은 모두 *가상화 배포 및 관리 가이드*에서 설명합니다.

pvpanic 메커니즘은 *가상화 배포 및 관리 가이드* - *창에서 장치 설정을* 참조하십시오.

virsh dump 명령은 *가상화 배포 및 관리 가이드* - *도메인 코어의 덤프 파일 생성*에 설명되어 있습니다.

Red Hat 지원 서비스에 대용량 덤프 파일을 업로드하려면 어떻게 해야 합니까?

경우에 따라 분석을 위해 커널 크래시 덤프 파일을 Red Hat 글로벌 지원 서비스로 보내야 할 수도 있습니다. 그러나 덤프 파일은 필터링 된 후에도 매우 클 수 있습니다. 새 지원 케이스를 열 때 250MB보다 큰 파일은 Red Hat 고객 포털을 통해 직접 업로드할 수 없으므로 Red Hat은 대용량 파일을 업로드하기 위해 FTP 서버를 제공합니다.

FTP 서버의 주소는 dropbox.redhat.com 이며 파일은 /incoming/ 디렉토리에 업로드됩니다. FTP 클라이언트를 패시브 모드로 설정해야 합니다. 방화벽에서 이 모드를 허용하지 않는 경우 활성 모드를 사용하여 origin-dropbox.redhat.com 서버를 사용하십시오.

업로드 된 파일이 다음과 같은 프로그램을 사용하여 압축되어 있는지 확인하십시오. gzip 그리고 적절하고 설명적으로 이름이 지정되어 있습니다. 파일 이름에 지원 케이스 번호를 사용하는 것이 좋습니다. 필요한 모든 파일을 성공적으로 업로드한 후 정확한 파일 이름과 SHA1 또는 MD5 체크섬을 사용하여 지원 케이스를 담당 엔지니어를 제공하십시오.

자세한 내용은 Red Hat 지원에 파일을 제공하는 방법을 참조하십시오.

크래시 덤프를 완료하는 데 시간이 얼마나 필요합니까?

종종 덤프를 완료하는 데 걸리는 시간을 확인하기 위해 재해 복구 계획의 목적을 위해 필요합니다. 그러나 걸리는 시간은 디스크에 복사되는 메모리 양과 RAM과 스토리지 간의 인터페이스 속도에 따라 크게 달라집니다.

타이밍 테스트의 경우 대표적 로드 하에서 시스템이 작동해야 합니다. 그렇지 않으면 페이지 제외 선택 사항이 완전히 로드된 프로덕션 시스템과 함께 kdump 동작에 대한 잘못된 보기를 제공할 수 있습니다. 이러한 불일치는 특히 대량의 RAM과 함께 작업할 때 존재합니다.

또한 덤프할 시간을 평가할 때 계획에서 스토리지 인터페이스를 고려합니다. 예를 들어 네트워크 제약 조건으로 인해 ssh 를 통해 덤프하는 연결에서 로컬로 연결된 SATA 디스크보다 완료하는 데 시간이 오래 걸릴 수 있습니다.

설치 중에 Kdump가 어떻게 구성되어 있습니까?

다음을 구성할 수 있습니다. kdump Kickstart의 옵션 세트 또는 대화형 GUI를 사용하여 설치하는 동안.

The kdump 를 사용한 구성 anaconda 설치 GUI는 설치 가이드의 KDUMP 섹션에 설명되어 있습니다.

The kickstart 구문은 다음과 같습니다.

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

Kickstart에 대한 이 애드온을 사용하면 kdump 기능을 비활성화하거나 활성화할 수 있으며 선택적으로 auto의 기본 옵션(전체 스위치가 생략되는 경우) 또는 메가바이트 단위로 숫자 값을 지정하여 예약된 메모리 크기를 정의할 수 있습니다.

Kickstart를 사용하여 시스템 배포를 자동화하는 방법을 알아보려면 설치 가이드에서 Kickstart 설치를 참조하십시오.

Kickstart 애드온 구문에 대한 자세한 내용은 설치 가이드의 Kickstart 구문 참조 를 참조하십시오.

7.8. 지원되는 KDUMP 구성 및 대상

7.8.1. kdump에 대한 메모리 요구 사항

kdump가 커널 크래시 덤프를 캡처하여 추가 분석을 위해 저장할 수 있으려면 캡처 커널에 대해 시스템 메모리의 일부를 영구적으로 예약해야 합니다.

명령줄에서 메모리 설정을 변경하는 방법에 대한 자세한 내용은 7.2.2.1절. “메모리 사용량 구성” 을 참조하십시오. 그래픽 사용자 인터페이스에서 예약된 메모리 양을 설정하는 방법에 대한 지침은 7.2.3.1절. “메모리 사용량 구성” 을 참조하십시오.

Table 7.1 은 커널 및 kernel -alt 패키지에서 kdump 에 의해 자동으로 예약된 메모리를 나열합니다. kdump 자동은 CPU 아키텍처 및 사용 가능한 총 실제 메모리를 기반으로 메모리를 예약합니다.

표 7.1. kdump에 의해 자동으로 예약된 총 충돌 메모리

CPU 아키텍처	사용 가능한 메모리	자동으로 예약된 메모리 충돌
AMD64 및 Intel 64(x86_64)	2GB 이상	161MB + 1TB당 64MB

CPU 아키텍처	사용 가능한 메모리	자동으로 예약된 메모리 충돌
64비트 ARM 아키텍처(arm64)	2GB 이상	512MB
IBM POWER ppc64/ppc64le	2GB ~ 4GB	384MB
	4GB ~ 16GB	512MB
	16GB ~ 64GB	1GB
	64GB ~ 128GB	2GB
	128GB 이상	4GB
IBM Z (s390x)	4GB 이상	161MB + 1TB 당 64MB [1] 160MB RHEL-ALT-7.6

*다양한 **Red Hat Enterprise Linux** 기술 기능 및 제한에 대한 자세한 내용은 https://access.redhat.com/articles/rhel-limits 을 참조하십시오.*

7.8.2. 자동 메모리 예약에 대한 최소 임계값

*일부 시스템에서는 부트로더의 설정 파일에서 **crashkernel=auto** 매개 변수를 사용하거나 그래픽 구성 유틸리티에서 이 옵션을 활성화하여 **kdump**에 메모리를 자동으로 할당할 수 있습니다. 이 자동 예약이 작동하려면 시스템에서 특정 양의 전체 메모리를 사용할 수 있어야 합니다. 이 금액은 시스템의 아키텍처에 따라 다릅니다.*

*아래 표에는 커널 및 **kernel-alt** 패키지에 대한 자동 메모리 할당 임계값이 나열되어 있습니다. 시스템이 테이블에서 지정한 메모리보다 적은 경우 메모리를 수동으로 예약해야 합니다.*

명령줄에서 이러한 설정을 변경하는 방법에 대한 자세한 내용은 7.2.2.1절. "메모리 사용량 구성" 을 참조하십시오. 그래픽 사용자 인터페이스에서 예약된 메모리 양을 변경하는 방법에 대한 지침은 7.2.3.1절. "메모리 사용량 구성" 을 참조하십시오.

표 7.2. 자동 메모리 예약에 필요한 최소 메모리 양

아키텍처	필수 메모리
AMD64 및 Intel 64(**x86_64**)	2GB
IBM POWER (**ppc64**)	2GB

아키텍처	필수 메모리
IBM Z (**s390x**)	4GB
64비트 ARM 아키텍처(m**d64**)	2GB

7.8.3. 지원되는 kdump 대상

*커널 크래시를 캡처하면 코어 덤프를 장치에 직접 쓰거나 로컬 파일 시스템에 파일로 저장되거나 네트워크를 통해 전송할 수 있습니다. 아래 표에는 현재 지원되거나 **kdump**에서 명시적으로 지원되지 않는 전체 덤프 대상 목록이 포함되어 있습니다.*

명령줄에서 대상 유형을 구성하는 방법에 대한 자세한 내용은 7.2.2.2절. "kdump 대상 구성" 을 참조하십시오. 그래픽 사용자 인터페이스에서 이를 수행하는 방법에 대한 자세한 내용은 7.2.3.2절. "kdump 대상 구성" 을 참조하십시오.

표 7.3. 지원되는 kdump 대상

유형	지원되는 대상	지원되지 않는 대상
원시 장치	로컬에서 연결된 모든 원시 디스크 및 파티션.	
로컬 파일 시스템	직접 연결된 디스크 드라이브, 하드웨어 RAID 논리 드라이브, LVM 장치 및 **mdraid** 어레이의 **ext2**,**ext3**,**ext4**, **xfs** 파일 시스템.	자동 유형(자동 파일 시스템 탐지)을 포함하여 이 표에서 지원되는 것으로 명시적으로 나열되지 않은 모든 로컬 파일 시스템입니다.
원격 디렉터리	**IPv4** 를 통해 **NFS** 또는 **SSH** 프로토콜을 사용하여 액세스하는 원격 디렉터리.	**NFS** 프로토콜을 사용하여 액세스한 **rootfs** 파일 시스템의 원격 디렉터리입니다.
FCoE (*Fibre Channel over Ethernet*) 프로토콜을 사용하여 액세스하는 원격 디렉터리입니다.	**qla2xxx**,**lpfc** 및 **bfa** 하드웨어 **FCoE** 대상. **bnx2fc** 및 **ixgbe** 소프트웨어 **FCoE** 대상.	
하드웨어 및 소프트웨어 초기자 둘 다에서 **iSCSI** 프로토콜을 사용하여 액세스하는 원격 디렉터리입니다.	**be2iscsi** 하드웨어에서 **iSCSI** 프로토콜을 사용하여 액세스하는 원격 디렉터리.	다중 경로 기반 스토리지.
		IPv6 를 통해 액세스하는 원격 디렉터리.

유형	지원되는 대상	지원되지 않는 대상
		SMB 또는 **CIFS** 프로토콜을 사용하여 액세스하는 원격 디렉터리입니다.
		무선 네트워크 인터페이스를 사용하여 액세스하는 원격 디렉터리.

참고

*소프트웨어 **FCoE** 대상으로 덤프할 때 **OOM**(메모리 부족) 문제가 발생할 수 있습니다. 이 경우 기본 **crashkernel=auto** 매개변수 값을 늘립니다. 이 커널 부팅 매개변수를 설정하는 방법에 대한 자세한 내용은 7.2.2.1절. "메모리 사용량 구성" 을 참조하십시오.*

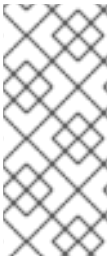
7.8.4. 지원되는 kdump 필터링 수준

*덤프 파일의 크기를 줄이기 위해 **kdump**는 **makedumpfile** 코어 수집기를 사용하여 데이터를 압축하고 선택적으로 관련 없는 정보를 남겨 둡니다. 아래 표에는 현재 **makedumpfile** 유틸리티에서 지원하는 필터링 수준의 전체 목록이 포함되어 있습니다.*

명령줄에서 코어 수집기를 구성하는 방법에 대한 지침은 7.2.2.3절. "코어 수집기 구성" 을 참조하십시오. 그래픽 사용자 인터페이스에서 이를 수행하는 방법에 대한 자세한 내용은 7.2.3.3절. "코어 수집기 구성" 을 참조하십시오.

표 7.4. 지원되는 필터링 수준

옵션	설명
1	0 페이지
2	페이지 캐시
4	캐시 프라이빗
8	사용자 페이지
16	사용 가능한 페이지



참고

makedumpfile 명령은 **Red Hat Enterprise Linux 7.3** 이상에서 투명한 대규모 페이지 및 **hugetlbfs** 페이지 제거를 지원합니다. 이러한 유형의 **hugepages** 사용자 페이지를 모두 고려하여 **-8** 수준을 사용하여 제거합니다.

7.8.5. 지원되는 기본 작업

기본적으로 **kdump**가 코어 덤프를 생성하지 못하면 운영 체제가 재부팅됩니다. 그러나 코어 덤프를 기본 대상에 저장하지 못하는 경우 다른 작업을 수행하도록 **kdump**를 구성할 수 있습니다. 아래 표에는 현재 **kdump**에서 지원하는 모든 기본 작업이 나열되어 있습니다.

명령줄에서 기본 작업을 설정하는 방법에 대한 자세한 내용은 **7.2.2.4절. "기본 작업 구성"** 을 참조하십시오. 그래픽 사용자 인터페이스에서 이를 수행하는 방법에 대한 자세한 내용은 **7.2.3.4절. "기본 작업 구성"** 을 참조하십시오.

표 **7.5.** 지원되는 기본 작업

옵션	설명
dump_to_rootfs	코어 덤프를 루트 파일 시스템에 저장합니다. 이 옵션은 특히 네트워크 대상과 함께 유용합니다. 네트워크 대상에 연결할 수 없는 경우 이 옵션을 사용하면 코어 덤프를 로컬에 저장하도록 kdump를 구성합니다. 나중에 시스템이 재부팅됩니다.
reboot	시스템을 다시 부팅하고 프로세스에서 코어 덤프를 손실합니다.
halt	시스템을 중단하고 프로세스에서 코어 덤프를 잃게 됩니다.
poweroff	시스템의 전원을 끄고 프로세스에서 코어 덤프를 잃습니다.
shell	initramfs 내에서 쉘 세션을 실행하여 사용자가 코어 덤프를 수동으로 기록할 수 있습니다.

7.8.6. kdump 크기 추정

kdump 환경을 계획 및 빌드할 때는 덤프 파일에 필요한 공간이 하나씩 생산되기 전에 얼마나 많은 공간이 필요한지 알아야 합니다. **makedumpfile** 명령은 이에 도움이 될 수 있습니다.

--mem-usage 기능을 사용하여 덤프 파일에 필요한 공간을 다음과 같이 추정합니다.

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

참고

-f 옵션을 사용하는 **--mem-usage** 기능은 커널 버전 **v4.11** 이상에서 작동합니다.

v4.11 이전 커널 버전의 경우 **-f** 옵션과 함께 **--mem-usage** 를 사용하기 전에 커널이 업스트림 커밋 **464920104bf7**로 패치되었는지 확인합니다.

--mem-usage 옵션은 **excludable** 페이지에 대한 유용한 보고서를 제공합니다. 이 페이지를 사용하여 할당할 덤프 수준을 결정합니다. 시스템이 대표적 로드 상태에 있는 경우 이 명령을 실행합니다. 그러지 않으면 **makedumpfile** 에서 프로덕션 환경에서 예상되는 값보다 작은 값을 반환합니다.

```
[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
```

중요

makedumpfile 명령은 페이지의 보고입니다. 즉, **Red Hat Enterprise Linux** 커널에서 사용할 커널 페이지 크기에 대해 사용 중인 메모리 크기를 계산해야 하며 **AMD64** 및 **Intel 64** 아키텍처의 경우 **4**킬로바이트, **IBM POWER** 아키텍처용 **64**킬로바이트입니다.

7.8.7. 커널 및 커널 -alt 패키지에서 아키텍처 지원

다음 표에서는 커널 및 커널 **-alt** 패키지에서 아키텍처 및 사용 가능한 메모리 지원에 대한 개요를 제공합니다.

표 **7.6.** 커널 및 커널 **-alt** 패키지에서 지원되는 아키텍처

CPU 아키텍처	사용 가능한 메모리	RHEL 7.5 및 이전	RHEL 7.6 이상	RHEL-ALT-7.4	RHEL-ALT-7.5 이상
AMD64 및 Intel 64(x86_64)	2GB 이상	161MB + 1TB당 64MB	161MB + 64MB/1TB	2GB ~ 160MB	2GB ~ 160MB
64비트 ARM 아키텍처 (arm64)	2GB 이상	해당 없음	해당 없음	2GB에서 512MB	2GB에서 512MB
IBM POWER ppc64/ppc64le (Upto POWER8)	2GB ~ 4GB	384MB	384MB	해당 없음	해당 없음
	4GB ~ 16GB	512MB	512MB	해당 없음	해당 없음
	16GB ~ 64GB	1GB	1GB	해당 없음	해당 없음
	64GB ~ 128GB	2GB	2GB	해당 없음	해당 없음
	128GB 이상	4GB	4GB	해당 없음	해당 없음
IBM POWER ppc64/ppc64le (Upto POWER9)	2GB ~ 4GB	384MB	384MB	384MB	384MB
	4GB ~ 16GB	512MB	512MB	512MB	512MB
	16GB ~ 64GB	1GB	1GB	1GB	1GB
	64GB ~ 128GB	2GB	2GB	2GB	2GB
	128GB 이상	4GB	4GB	4GB	4GB
IBM POWER ppc64le (POWER9)	2GB ~ 4GB	해당 없음	해당 없음	384MB	384MB
	4GB ~ 16GB	해당 없음	해당 없음	512MB	512MB
	16GB ~ 64GB	해당 없음	해당 없음	1GB	1GB
	64GB ~ 128GB	해당 없음	해당 없음	2GB	2GB
	128GB 이상	해당 없음	해당 없음	4GB	4GB
IBM Z (s390x)	4GB 이상	161MB + 1TB당 64MB	161MB + 1TB당 64MB	161MB + 1TB당 64MB	160MB

7.9. KEXEC를 사용하여 커널 재부팅

7.9.1. kexec를 사용하여 커널 재부팅

kexec 시스템 호출을 사용하면 현재 실행 중인 커널에서 다른 커널로 로드 및 부팅될 수 있으므로 커널 내에서 부트 로더의 기능을 수행할 수 있습니다.

kexec 유틸리티는 kexec 시스템 호출의 kernel 및 initramfs 이미지를 로드하여 다른 커널로 부팅합니다.

다음 섹션에서는'kexec의 유틸리티를 사용하여 다른 커널로 재부팅할 때 kexec 시스템 호출을 수동으로 호출하는 방법을 설명합니다.

1. *kexec 유틸리티를 실행합니다.*

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

명령은 kexec 시스템 호출의 kernel 및 initramfs 이미지를 수동으로 로드합니다.

2. *시스템을 재부팅합니다.*

```
# reboot
```

명령은 커널을 감지하고 모든 서비스를 종료한 다음 kexec 시스템 호출을 호출하여 이전 단계에서 제공한 커널에 재부팅합니다.

주의

kexec -e 명령을 사용하여 커널을 재부팅할 때 다음 커널을 시작하기 전에 시스템이 표준 종료 시퀀스를 통과하지 않으므로 데이터가 손실되거나 응답하지 않는 시스템이 발생할 수 있습니다.

7.10. KDUMP와 관련된 포털 랩

Portal Labs 는 시스템 관리자가 여러 시스템 작업을 수행하는 데 도움이 되는 작은 웹 애플리케이션입니다. 현재 Kdump에 대한 두 가지 랩이 있습니다. Kdump Helper 및 커널 Oops Analyzer.

7.10.1. kdump 도우미

Kdump Helper 는 kdump 구성 파일을 준비하는 데 도움이 되는 일련의 질문 및 작업입니다.

랩의 워크플로에는 클러스터형 환경과 독립 실행형 환경 모두에 대한 단계가 포함됩니다.

7.10.2. 커널 oops Analyzer

Kernel Oops Analyzer 는 크래시 덤프 스택을 해제하지 않고도 Oops 메시지를 처리하고 알려진 솔루션을 검색하는 툴입니다.

커널 Oops Analyzer는 makedumpfile 의 정보를 사용하여 충돌된 시스템의 oops 메시지를 기술 자료의 알려진 문제와 비교합니다. 이를 통해 시스템 관리자는 예기치 않은 중단 후 추가 분석을 위해 지원 티켓을 열기 전에 알려진 문제를 신속하게 배제할 수 있습니다.

8장. 커널 무결성 하위 시스템으로 보안 강화

8.1. 커널 무결성 하위 시스템

무결성 하위 시스템은 시스템의 데이터 무결성을 유지 관리하는 커널의 일부입니다. 이 하위 시스템은 사용자가 특정 시스템 파일을 원치 않는 변경을 방지하여 시스템을 초기 상태로 유지하는 데 도움이 됩니다.

커널 무결성 하위 시스템은 다음 두 가지 주요 구성 요소로 구성됩니다.

IMA(Integrity Measurement Architecture)

- 실행 또는 열 때마다 파일의 콘텐츠를 측정합니다. 사용자는 사용자 지정 정책을 적용하여 이 동작을 변경할 수 있습니다.
- 측정된 값을 커널의 메모리 공간 내에 배치하여 시스템 사용자가 수정할 수 없습니다.
- 로컬 및 원격 당사자가 측정 값을 확인할 수 있도록 합니다.

EVM(Extended Verification Module)

- **IMA** 측정 및 **SELinux** 속성과 같은 시스템의 보안과 관련된 파일의 확장된 속성(**xattr**라고도 함)을 해당 값을 암호화하여 보호합니다.

IMA 및 **EVM** 모두 추가 기능을 가져오는 다양한 기능 확장 기능이 포함되어 있습니다. 예를 들어 다음과 같습니다.

IMA-Appraisal

- 커널 메모리 내의 측정 파일에 이전에 저장된 값에 대해 현재 파일의 내용을 로컬에서 검증합니다. 이 확장에서는 현재 측정이 이전 측정과 일치하지 않는 경우 특정 파일을 통해 수행할 모든 작업을 금지합니다.

EVM 디지털 서명

- 커널 키링에 저장된 암호화 키를 통해 디지털 서명을 사용할 수 있습니다. **EVM Digital Signature**는 콘텐츠 해시(**security.ima**)가 포함된 파일의 출처 및 무결성을 보장합니다.

참고

기능 확장 기능은 서로를 보완하지만 서로 독립적으로 구성하고 사용할 수 있습니다.

커널 무결성 하위 시스템은 신뢰할 수 있는 플랫폼 모듈(TPM)을 사용하여 시스템 보안을 더욱 강화할 수 있습니다. TPM은 중요한 암호화 기능에 대한 Trusted Computing Group(TCG)의 사양입니다. TPM은 일반적으로 플랫폼의 마더보드에 연결된 전용 하드웨어로 구현되며 하드웨어 칩의 보호 및 변조 영역에서 암호화 기능을 제공하여 소프트웨어 기반 공격을 방지합니다. TPM 기능 중 일부는 다음과 같습니다.

- random-number 생성기
- 암호화 키를 위한 Generator 및 secure storage
- 해시 생성기
- 원격 인증

8.2. 무결성 측정 아키텍처

IMA(Integrity Measurement Architecture)는 커널 무결성 하위 시스템의 구성 요소입니다. IMA는 액세스 하기 전에 파일의 해시를 측정, 저장 및 앱 올리는 방식으로 로컬 파일의 내용을 유지 하는 것을 목표로 합니다. 이렇게 하면 신뢰할 수 없는 데이터의 읽기 및 실행을 방지하고 시스템 보안이 향상됩니다.

8.3. 확장 검증 모듈

EVM(Extended Verification Module)은 파일의 확장 속성(xattr)의 변경 사항을 모니터링하는 커널 무결성 하위 시스템의 구성 요소입니다. 무결성 측정 아키텍처(IMA(Integrity Measurement Architecture)를 포함한 많은 보안 지향 기술은 콘텐츠 해시와 같은 중요한 파일 정보를 확장된 특성에 저장합니다. EVM은 이러한 확장 속성과 부팅 시 로드되는 특수 키에서 또 다른 해시를 생성합니다. 확장된 속성이 사용될 때마다 결과 해시의 유효성이 검사됩니다. 예를 들어 IMA에서 파일을 예상하는 경우입니다.

8.4. 신뢰할 수 있는 암호화된 키

신뢰할 수 있고 암호화된 키는 커널 키링 서비스에서 사용하는 커널에서 생성되는 변수 길이의 대칭 키입니다. 이러한 유형의 키는 암호화되지 않은 형식으로 사용자 공간에 표시되지 않으므로 무결성을 확인할 수 있습니다. 따라서 예를 들어 확장 확인 모듈(EVM)을 사용하여 실행 중인 시스템의 무결성을 확인하고 확인할 수 있습니다. 사용자 수준 프로그램은 암호화된 Blob 형식의 키만 액세스할 수 있습니다.

신뢰할 수 있는 키에는 키를 만들고 암호화(seal)하는 데 사용되는 신뢰할 수 있는 플랫폼 모듈(TPM) 칩의 하드웨어 구성 요소가 필요합니다. TPM은 스토리지 루트 키(SRK)라는 2048비트 RSA 키를 사용하여 키를 봉인합니다.

신뢰할 수 있고 암호화된 키에 대한 자세한 내용은 다음을 참조하십시오. Trusted and Encrypted Keys 섹션 RHEL 7 보안 가이드.

8.5. 무결성 측정 아키텍처 및 확장 검증 모듈 활성화

무결성 측정 아키텍처(IMA) 및 EVM(Extended verification module)은 다양한 방식으로 시스템 보안을 향상시키는 커널 무결성 하위 시스템의 구성 요소입니다. IMA 및 EVM을 구성하면 파일에 서명하여 시스템 보안을 강화할 수 있습니다.

사전 요구 사항

- ima-evm-utils 및 keyutils 패키지가 시스템에 설치됩니다.
- securityfs 파일 시스템은 /sys/kernel/security/ 디렉토리에 마운트됩니다.
- /sys/kernel/security/ima/ 디렉토리가 있습니다.

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

절차

1. IMA 및 EVM을 사용하도록 시스템을 준비합니다.

   a. 다음 커널 명령줄 매개변수를 추가합니다.

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

명령을 사용하면 현재 부팅 항목에 대한 수정 모드에서 **IMA** 및 **EVM**을 사용할 수 있으며 사용자가 **IMA** 측정을 수집 및 업데이트할 수 있습니다.

ima_appraise_tcb 커널 명령행 매개변수를 사용하면 커널에서 기본 **Trusted Computing Base(TCB)** 측정 정책 및 **appraisal** 단계를 사용합니다. 이전 측정 및 현재 측정이 일치하지 않는 파일에 대한 접근을 금지하는 단계 **(Arraisal step forbids access to files whose prior and current measurements do not match.)**

b. 변경 사항을 적용하려면 재부팅합니다.

c. 선택적으로 커널 명령줄에 새 매개변수가 포함되어 있는지 확인합니다.

```
# 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**의 공용 및 개인 키 쌍을 만들고 설정합니다.

a. **EVM**의 새 인증 키를 만듭니다.

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

이 명령은 **_evm** 인증 키를 생성하여 **@u** 시스템 사용자 인증 키에 연결합니다. 그런 다음 나중에 더 편리하게 처리하기 위해 **_evm** 의 인증 키 **ID**가 **evm_kr_id** 변수에 할당됩니다.

b. 선택적으로 새로 생성된 인증 키를 확인합니다.

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

c.

키의 디렉터리를 생성합니다.

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

d.

***/etc/keys/privkey.pem** 파일에 **1024**비트 **RSA** 개인 키를 생성합니다.*

```
# openssl genrsa -out /etc/keys/privkey.pem 1024
Generating RSA private key, 1024 bit long modulus
.......++++++
...++++++
e is 65537 (0x10001)
```

e.

*이전에 생성된 **/etc/keys/privkey.pem** 개인 키를 사용하여 해당 **RSA** 공개 키를 **/etc/keys/pubkey.pem** 파일로 가져옵니다.*

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

f.

*공개 키를 전용 **EVM** 인증 키로 가져옵니다.*

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

*명령은 **/etc/keys/pubkey.pem** 공개 키를 **_evm** 인증 키로 가져옵니다. 그런 다음 **_evm** 인증 키가 커널 인증 키에 연결됩니다. 키 일련 번호는 이전 예제의 두 번째 줄에 있습니다.*

g.

선택적으로 새로 가져온 키를 확인합니다.

```
# 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
```

참고

*이 **symmetric** 키 쌍을 사용하여 **eRuntimeConfig sign** 명령을 사용하여 파일의 확장된 속성의 내용을 디지털 서명하는 데 사용할 수 있습니다. 확장 속성은 나중에 커널에서 확인합니다.*

h.

EVM 키를 보호하기 위해 커널 마스터 키를 생성합니다.

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

커널 마스터 키(kmk)는 전적으로 커널 공간 메모리에 유지됩니다. 커널 마스터 키 kmk 의 32바이트 긴 값은 /dev/urandom 파일에서 임의의 바이트에서 생성되며 사용자(@u) 인증 키에 배치됩니다.

3.

kmk 키를 기반으로 암호화된 EVM 키를 만듭니다.

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

명령은 kmk 를 사용하여 64바이트 사용자 키(evm-key)를 생성하고 암호화하며 사용자(@u) 인증 키에 배치합니다. 키 일련 번호는 이전 예제의 두 번째 줄에 있습니다.

중요

이는 EVM 하위 시스템에서 예상하고 함께 작동하는 이름이므로 사용자 키 evm-key 의 이름을 지정해야 합니다.

4.

EVM을 활성화합니다.

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

5.

EVM이 초기화되었는지 확인합니다.

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

8.6. 무결성 측정 아키텍처를 사용하여 파일 해시 수집

무결성 측정 아키텍처(IMA)의 첫 번째 작업 수준은 파일 해시를 생성하고 해당 파일의 확장 속성 (xattrs)으로 저장할 수 있는 측정 단계입니다. 다음 섹션에서는 파일 해시를 생성하고 검사하는 방법을 설명합니다.

사전 요구 사항

- *ima-evm-utils,attr 및 keyutils 패키지가 시스템에 설치되어 있습니다.*

- *무결성 측정 아키텍처(IMA) 및 EVM(Extended verification module)은 8.5절. "무결성 측정 아키텍처 및 확장 검증 모듈 활성화" 에 설명된 대로 활성화됩니다.*

절차

1. *테스트 파일을 생성합니다.*

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

2. *개인 키로 파일에 서명합니다.*

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

test_file 파일의 해시를 생성하면 IMA는 파일이 손상되지 않은 상태로 남아 있는지 확인합니다. EVM은 test_file 의 확장 속성에 저장된 해시 콘텐츠에 서명하여 IMA 해시를 정품으로 보장합니다.

3. *선택적으로 서명된 파일의 확장 속성을 확인합니다.*

```
# getfattr -m . -d test_file
file: test_file
security.evm=0sAwICztVdCQCAZLtD7qAezGl8nGLgqFZzMzQp7Fm1svUet2Hy7Tyl2vtT
9/9ZfBTKMK6Mjoyk0VX+DciS85XOCYX6WnV1LF2P/pmPRfputSEq9fVD4SWfKKj2rl7qw
pndC1UqRX1BbN3aRUYeoKQdPdl6Cz+cX4d7vS56FJkFhPGlhq/UQbBnd80=
security.ima=0sAfgqQ5/05X4w/ltZEfbogdl9+KM5
security.selinux="unconfined_u:object_r:admin_home_t:s0"
```

이 예제 출력에서는 SELinux 및 IMA 및 EVM 해시 값과 관련된 확장된 속성을 보여줍니다. EVM은 security.evm 확장 속성을 적극적으로 추가하고 파일 콘텐츠 무결성과 직접 관련된 security.ima 와 같은 다른 파일의 xattrs에 대한 오프라인 변조를 감지합니다. security.evm 필드의 값은 개인 키로 생성된 Hash-based Message Authentication Code(HMAC-SHA1)에 있습니다.

추가 리소스

- *커널 무결성 하위 시스템에 대한 자세한 내용은 을 참조하십시오. official upstream wiki page.*

- *TPM에 대한 자세한 내용은 다음을 참조하십시오. Trusted Computing Group resources.*

- *암호화된 키 생성에 대한 자세한 내용은 을 참조하십시오. Trusted and Encrypted Keys 섹션 RHEL 7 보안 가이드.*

9장. 개정 내역

0.1-8

Tue 2020년 9월 29일 Jaroslav Klech (jklech@redhat.com)

- *7.9 GA 게시에 대한 문서 버전입니다.*

0.1-7

Tue Mar 31 2020, Jaroslav Klech (jklech@redhat.com)

- *7.8 정식 출시일 (GA)*

0.1-6

2019년 8월 6일, Jaroslav Klech (jklech@redhat.com)

- *7.7 GA 게시에 대한 문서 버전.*

0.1-5

2018년 10월 19일 Jaroslav Klech (jklech@redhat.com)

- *7.6 GA 게시를 위한 문서 버전입니다.*

0.1-4

2018년 3월 26일 Marie DoleReviewelov receiving (mdolezel@redhat.com)

- *7.5 GA 게시를 위한 문서 버전.*

0.1-3

2018년 1월 5일 Mark Flitter (mflitter@redhat.com)

- *7.5 베타 게시를 위한 문서 버전.*

0.1-2

2017년 7월 31일, Mark Flitter (mflitter@redhat.com)

- *7.4 GA 게시의 문서 버전입니다.*

0.1-0

2017년 4월 20일, Mark Flitter (mflitter@redhat.com)

- *검토를 위한 초기 빌드*