Red Hat Enterprise Linux 10

위험 감소 및 복구 작업

데이터 백업, 로그 모니터링 및 보안 업데이트 관리

Last Updated: 2025-08-03

Red Hat Enterprise Linux 10 위험 감소 및 복구 작업

데이터 백업, 로그 모니터링 및 보안 업데이트 관리

Legal Notice

Abstract

Relax-and-Recover (ReaR) 재해 복구 도구를 사용하여 재해 이벤트의 영향을 최소화하고 오류 후 시스템 및 데이터를 복원하기위한 체계적인 계획을 가지고 있습니다. 시스템 안정성, 보안 및 성능을 개선하기 위해 보안 업데이트를 관리하고 모니터링하는 방법을 알아보십시오. 로그 파일을 사용하여 기록된 시스템 이벤트를 검사하여 문제를 식별하고 해결하며 시스템 기능을 모니터링합니다.

Table of Contents

RED HAT 문서에 관한 피드백 제공

문서 개선을 위한 의견에 감사드립니다. 어떻게 개선할 수 있는지 알려주십시오.

Jira를 통해 피드백 제출 (계정 필요)

1. Jira 웹 사이트에 로그인합니다.
2. 상단 탐색 바에서 **생성**을 클릭합니다.
3. **요약** 필드에 설명 제목을 입력합니다.
4. **설명** 필드에 개선을 위한 제안을 입력합니다. 문서의 관련 부분에 대한 링크를 포함합니다.
5. 대화 상자 하단에서 **생성** 을 클릭합니다.

1장. 시스템 복구 및 복원

Red Hat Enterprise Linux는 기존 백업을 사용하여 시스템을 복구 및 복원하는 Relax-and-Recover(ReaR) 유틸리티를 제공합니다. 유틸리티를 재해 복구 솔루션 및 시스템 마이그레이션으로 사용할 수도 있습니다.

유틸리티를 사용하면 다음 작업을 수행할 수 있습니다.

- 이미지를 사용하여 부팅 가능한 이미지를 생성하고 기존 백업에서 시스템을 복원합니다.
- 원래 스토리지 레이아웃을 복제합니다.
- 사용자 및 시스템 파일을 복원합니다.
- 시스템을 다른 하드웨어에 복원합니다.

또한 재해 복구를 위해 특정 백업 소프트웨어를 ReaR과 통합할 수도 있습니다.

1.1. REAR 설정 및 수동으로 백업 생성

다음 단계를 사용하여 Relax-and-Recover(ReaR) 유틸리티를 사용하여 패키지를 설치하고, 복구 시스템을 생성하고, 백업을 구성하고 생성합니다.

사전 요구 사항

- 백업 복원 계획에 따라 필요한 구성이 준비되었습니다.
  ReaR에서 완전히 통합된 기본 제공 방법인 **NETFS** 백업 방법을 사용할 수 있습니다.

프로세스

1. ReaR 유틸리티를 설치합니다.

   ```
   # dnf install rear
   ```

2. 선택한 편집기에서 ReaR 구성 파일을 수정합니다. 예를 들면 다음과 같습니다.

   ```
   # vi /etc/rear/local.conf
   ```

3. **/etc/rear/local.conf** 에 백업 설정 세부 정보를 추가합니다. 예를 들어 **NETFS** 백업 방법의 경우 다음 행을 추가합니다.

   ```
   BACKUP=NETFS
   BACKUP_URL=backup.location
   ```

   backup.location 을 백업 위치의 URL로 바꿉니다.

4. 새 파일이 생성될 때 이전 백업 아카이브를 유지하도록 ReaR을 구성하려면 구성 파일에 다음 행도 추가합니다.

   ```
   NETFS_KEEP_OLD_BACKUP_COPY=y
   ```

5. 백업을 늘리려면 각 실행 시 변경된 파일만 백업됩니다.

```
BACKUP_TYPE=incremental
```

6. 복구 시스템을 생성합니다.

```
# rear mkrescue
```

7. 복원 계획에 따라 백업을 생성합니다. 예를 들어 **NETFS** 백업 방법의 경우 다음을 입력합니다.

```
# rear mkbackuponly
```

또는 다음 명령을 실행하여 복구 시스템 및 백업을 단일 단계로 생성할 수 있습니다.

```
# rear mkbackup
```

이 명령은 **rear mkrescue** 의 기능을 결합하고 **mkbackuponly** 명령을 다시 수행합니다.

1.2. 64비트 IBM Z 아키텍처에서 REAR 복구 이미지 사용

기본 Relax 및 Recover(ReaR) 기능은 64비트 IBM Z 아키텍처에서 사용할 수 있으며 완전히 지원됩니다. z/VM 환경에서만 IBM Z에서 ReaR 복구 이미지를 생성할 수 있습니다. LPAR(Logical partition) 백업 및 복구는 테스트되지 않았습니다.

현재 사용할 수 있는 유일한 출력 방법은 초기 프로그램 로드(IPL)입니다. IPL은 **zIPL** 부트 로더와 함께 사용할 수 있는 커널 및 초기 RAM 디스크(initrd)를 생성합니다.

사전 요구 사항

- **rear** 패키지를 설치했습니다.

프로세스

1. 64비트 IBM Z 아키텍처에서 복구 이미지를 생성하도록 ReaR을 구성하려면 **/etc/rear/local.conf** 에 다음 변수를 추가합니다.

   a. **IPL** 출력 방법을 구성하려면 **OUTPUT=IPL** 을 추가합니다.

   b. 백업 방법과 대상을 구성하려면 Cryostat 및 Cryo **stat _URL** 변수를 추가합니다. 예를 들면 다음과 같습니다.

```
BACKUP=NETFS

BACKUP_URL=nfs://<nfsserver_name>/<share_path>
```

중요

로컬 백업 스토리지는 현재 64비트 IBM Z 아키텍처에서 지원되지 않습니다.

   c. 선택 사항: **OUTPUT_URL** 변수를 구성하여 커널 및 **initrd** 파일을 저장할 수도 있습니다. 기본적으로 **OUTPUT_URL** 은 Cryostat_URL **과** 정렬됩니다.

2. 백업 및 복구 이미지 생성을 수행하려면 다음을 수행합니다.

```
# rear mkbackup
```

3. 이렇게 하면 Cryostat _**URL 또는 OUTPUT_URL** (설정된 경우) 변수에서 지정한 위치에 kernel 및 initrd 파일과 지정된 백업 방법을 사용하여 백업이 생성됩니다.

주의

복구 프로세스는 시스템에 연결된 모든 DASD(Direct Attached Storage Devices)를 다시 포맷합니다. 시스템 스토리지 장치에 중요한 데이터가 있는 경우 시스템 복구를 시도하지 마십시오. 여기에는 복구 환경으로 부팅하는 데 사용된 zipl 부트 로더, ReaR 커널 및 initrd로 준비된 장치도 포함됩니다. 사본을 보관해야 합니다.

4. 시스템을 복구하려면 이전에 생성된 ReaR 커널 및 initrd 파일을 사용하고 Direct Attached Storage Device(DASD) 또는 **zipl** 부트 로더, kernel 및 **initrd** 와 함께 준비된 파이버 채널 프로토콜(FCP) 연결 SCSI 장치에서 부팅합니다.

5. rescue 커널 및 **initrd** 가 부팅되면 ReaR 복구 환경이 시작됩니다. 시스템 복구를 진행합니다.

추가 리소스

- z/VM에서 설치
- 준비된 DASD 사용

1.3. REAR EXCLUSIONS

ReaR 유틸리티는 복구 프로세스 중 **/var/lib/rear/layout/disklayout.conf** 레이아웃 파일의 설명에 따라 복구된 시스템의 디스크에서 복구 이미지가 생성된 원래 시스템의 스토리지 레이아웃을 다시 생성합니다. 스토리지 레이아웃에는 파티션, 볼륨 그룹, 논리 볼륨, 파일 시스템 및 기타 스토리지 구성 요소가 포함됩니다.

Rear는 복구 이미지를 생성할 때 레이아웃 파일을 생성하고 이 파일을 이미지에 포함합니다. **rear savelayout** 명령을 사용하여 레이아웃 파일을 생성할 수도 있습니다. 이렇게 하면 전체 복구 이미지를 생성하지 않고도 레이아웃 파일을 빠르게 생성하고 검사할 수 있습니다.

레이아웃 파일은 특정 예외를 제외하고 원래 시스템의 전체 스토리지 레이아웃을 설명합니다. ReaR은 레이아웃 파일에서 일부 스토리지 구성 요소를 제외하고 복구 중에 재생성되지 않습니다. 레이아웃에서 스토리지 구성 요소를 제외하는 것은 다음 구성 변수에 의해 제어됩니다.

- **AUTOEXCLUDE_DISKS**
- **AUTOEXCLUDE_MULTIPATH**
- **AUTOEXCLUDE_PATH**
- **EXCLUDE_RECREATE**

구성 변수로 레이아웃 파일에서 일부 파일 시스템을 제외하면 백업에서 해당 콘텐츠도 제외됩니다. 또한 Cryostat **_PROG_EXCLUDE** 구성 변수를 사용하여 레이아웃 파일에서 파일 시스템을 제외하지 않고 백업에서 파일 또는 디렉터리 트리를 제외할 수도 있습니다.

파일 시스템의 모든 파일 및 디렉터리가 이러한 방식으로 제외되면 복구 중에 파일 시스템을 다시 생성하지만 백업에 복원할 데이터가 포함되어 있지 않기 때문에 비어 있습니다. 이는 임시 데이터가 포함되어 있지 않은 파일 시스템이나 ReaR과 무관한 방법을 사용하여 백업되는 데이터에 유용합니다.

Cryo **stat_PROG_EXCLUDE** 변수는 tar 또는 rsync로 전달되는 glob 형식의 와일드카드 패턴의 배열입니다. 구성 파일을 읽을 때 쉘이 확장되지 않도록 패턴을 인용해야 합니다. 이 변수의 기본값은 **/usr/share/rear/conf/default.conf** 파일에 설정됩니다. 기본값에는 **/tmp** 디렉토리 아래의 모든 파일과 디렉터리를 제외하지만 **/tmp** 디렉토리 자체는 제외하는 **/tmp/*** 패턴이 포함됩니다.

다른 파일 및 디렉터리를 제외해야 하는 경우 기본값을 유지하기 위해 재정의하지 않고 **+** 문자가 있는 패턴을 변수에 추가합니다. 예를 들어 기본값 외에도 **/data/temp** 디렉토리 아래의 모든 파일 및 디렉터리를 제외하려면 다음을 사용합니다.

```
BACKUP_PROG_EXCLUDE+=( '/data/temp/*' )
```

/usr/share/rear/conf/default.conf 파일에서 구성 변수의 기본값을 보고 로컬 **/etc/rear/local.conf** 구성 파일에서 이러한 값을 변경할 수 있습니다.

또한 내부 **NETFS** 및 **RSYNC** 백업 방법으로 백업되는 파일을 구성할 수도 있습니다. 파일 시스템이 레이아웃 파일에 포함된 경우 기본적으로 마운트된 모든 로컬 디스크 기반 파일 시스템은 **rear mkbackup** 또는 **rear mkbackuponly** 명령으로 백업됩니다.

rear mkbackup 명령은 로그에 백업 제외 패턴을 나열합니다. 로그 파일은 **/var/log/rear** 디렉토리에서 찾을 수 있습니다. 이는 전체 시스템 복구를 수행하기 전에 제외된 규칙을 확인하는 데 사용할 수 있습니다. 예를 들어 로그에 다음 항목이 포함될 수 있습니다.

```
2025-04-29 10:17:41.312431050 Making backup (using backup method NETFS)
2025-04-29 10:17:41.314369109 Backup include list (backup-include.txt contents):
2025-04-29 10:17:41.316197323   /
2025-04-29 10:17:41.318052001 Backup exclude list (backup-exclude.txt contents):
2025-04-29 10:17:41.319857125   /tmp/*
2025-04-29 10:17:41.321644442   /dev/shm/*
2025-04-29 10:17:41.323436363   /var/lib/rear/output/*
```

이전 출력에서 전체 루트 파일 시스템은 백업에 포함되며 **/tmp**,**/dev/shm**, **/var/lib/rear/output** 디렉터리 아래의 모든 파일과 디렉터리를 제외하고 백업에 포함됩니다.

추가 리소스

- **/usr/share/doc/ rear /relax-and-recover-user-guide.html**에 ReaR 사용자 가이드의 **레이아웃 구성** 장
- 시스템에 있는 **glob** Cryostat 도움말 페이지

2장. 로그 파일을 사용하여 문제 해결

로그 파일의 정보를 사용하여 시스템 기능 문제 해결 및 모니터링할 수 있습니다. 로그 파일에는 커널 및 실행 중인 서비스 및 애플리케이션을 포함하여 시스템에 대한 메시지가 포함되어 있습니다. Red Hat Enterprise Linux의 로깅 시스템은 기본 제공 **syslog** 프로토콜을 기반으로 합니다. 그런 다음 다양한 프로그램이 **syslog** 를 사용하여 이벤트를 기록하고 로그 파일로 구성합니다.

2.1. SYSLOG 메시지를 처리하는 서비스

다음 서비스는 **syslog** 메시지를 처리합니다.

systemd-journald 데몬

다음 소스에서 메시지를 수집하여 추가 처리를 위해 **Rsyslog** 에 전달합니다.

- 커널
- 부팅 프로세스의 초기 단계
- 데몬을 시작하고 실행할 때 표준 및 오류 출력
- **syslog**

Rsyslog 서비스

syslog 메시지를 유형 및 우선 순위에 따라 정렬하고 **/var/log** 디렉터리의 파일에 씁니다. **/var/log** 디렉터리는 로그 메시지를 영구적으로 저장합니다.

2.2. SYSLOG 메시지를 저장하는 하위 디렉터리

/var/log 디렉토리 아래의 다음 하위 디렉터리는 **syslog** 메시지를 저장합니다.

/var/log/messages

다음을 제외한 모든 **syslog** 메시지

/var/log/secure

보안 및 인증 관련 메시지 및 오류

/var/log/maillog

메일 서버 관련 메시지 및 오류

/var/log/cron

주기적으로 실행되는 작업과 관련된 로그 파일

/var/log/boot.log

시스템 시작과 관련된 로그 파일

2.3. 로그를 보는 명령

systemd 의 구성 요소인 Journal을 사용하여 로그 파일을 보고 관리할 수 있습니다. 기존 로깅과 관련된 문제를 해결하고 나머지 시스템과 밀접하게 통합되어 있으며 로그 파일에 대한 다양한 로깅 기술 및 액세스 관리를 지원합니다.

journalctl 명령을 사용하여 시스템 저널의 메시지를 볼 수 있습니다. 예를 들면 다음과 같습니다.

```
$ journalctl -b | grep kvm
May 15 11:31:41 localhost.localdomain kernel: kvm-clock: Using msrs 4b564d01 and 4b564d00
May 15 11:31:41 localhost.localdomain kernel: kvm-clock: cpu 0, msr 76401001, primary cpu clock
```

시스템 정보 보기

journalctl

수집된 모든 저널 항목을 표시합니다.

journalctl FILEPATH

는 특정 파일과 관련된 로그를 표시합니다. 예를 들어 **journalctl /dev/sda** 명령은 /dev/sda 파일 시스템과 관련된 로그를 표시합니다.

journalctl -b

현재 부팅에 대한 로그를 표시합니다.

journalctl -k -b -1

현재 부팅에 대한 커널 로그를 표시합니다.

특정 서비스에 대한 정보 보기

journalctl -b _SYSTEMD_UNIT=*<name.service>*

로그를 필터링하여 **systemd** 서비스와 일치하는 항목을 표시합니다.

journalctl -b _SYSTEMD_UNIT=*<name.service>* _PID=*<number>*

일치 항목 결합. 예를 들어 이 명령은 < ***name.service*** > 및 PID < ***number*** >와 일치하는 **systemd-units** 에 대한 로그를 표시합니다.

journalctl -b _SYSTEMD_UNIT= <*name.service* > _PID= <*number* > + _SYSTEMD_UNIT= *<name2.service>*

더하기 기호(+) 구분 기호는 논리 OR로 두 표현식을 결합합니다. 예를 들어 이 명령은 < ***name.service*** > 서비스 프로세스의 모든 메시지와 < ***name2.service*** > 서비스의 모든 메시지(프로세스 중 하나)를 표시합니다.

journalctl -b _SYSTEMD_UNIT=*<name.service>* _SYSTEMD_UNIT=*<name2.service>*

이 명령은 동일한 필드를 참조하여 두 식과 일치하는 모든 항목을 표시합니다. 여기에서 이 명령은 systemd-unit < *name.service* > 또는 systemd-unit < *name2.service* >와 일치하는 로그를 표시합니다.

특정 부팅과 관련된 로그 보기

journalctl --list-boots

부팅 번호, ID 및 부팅과 관련된 첫 번째 및 마지막 메시지의 타임스탬프에 대한 표 형식 목록을 표시합니다. 다음 명령에서 ID를 사용하여 자세한 정보를 볼 수 있습니다.

journalctl --boot=ID _SYSTEMD_UNIT=*<name.service>*

지정된 부팅 ID에 대한 정보를 표시합니다.

2.4. 추가 리소스

- **journalctl(1)** 시스템의 도움말 페이지
- 원격 로깅 솔루션 구성

3장. 웹 콘솔에서 로그 검토 및 필터링

Red Hat Enterprise Linux 웹 콘솔은 로그 액세스, 검토 및 필터링을 위한 그래픽 인터페이스를 제공합니다. 해당 명령 및 옵션을 암기하지 않고 가장 일반적인 함수를 사용할 수 있습니다.

3.1. 웹 콘솔에서 로그 검토

RHEL 웹 콘솔 로그 섹션은 **journalctl** 유틸리티의 **UI**입니다. 웹 콘솔 인터페이스에서 시스템 로그에 액세스할 수 있습니다.

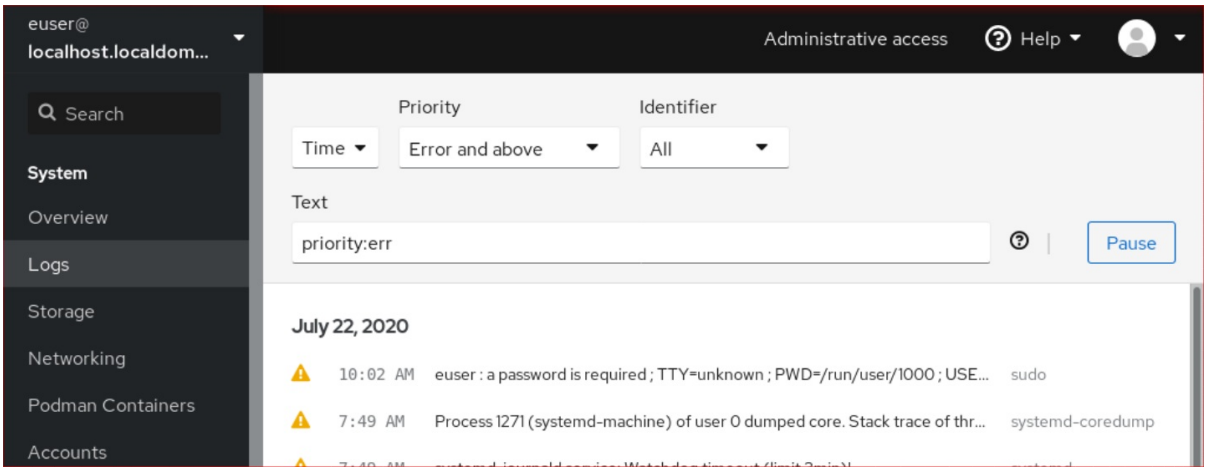
사전 요구 사항

- **RHEL 10** 웹 콘솔을 설치했습니다.

  자세한 내용은 웹 콘솔 설치 및 활성화를 참조하십시오.

프로세스

1. **RHEL 10** 웹 콘솔에 로그인합니다.
2. 로그를 클릭합니다.



3. 목록에서 선택한 로그 항목을 클릭하여 로그 항목 세부 정보를 엽니다.

참고

일시 중지 버튼을 사용하여 새 로그 항목이 표시되지 않도록 일시 중지할 수 있습니다. 새 로그 항목을 다시 시작하면 웹 콘솔은 일시 정지 버튼을 사용한 후 보고된 모든 로그 항목을 로드합니다.

로그는 시간, 우선 순위 또는 식별자별로 필터링할 수 있습니다. 자세한 내용은 웹 콘솔의 로그 검토 및 필터링을 참조하십시오.

3.2. 웹 콘솔에서 로그 필터링

웹 콘솔에서 로그 항목을 필터링할 수 있습니다.

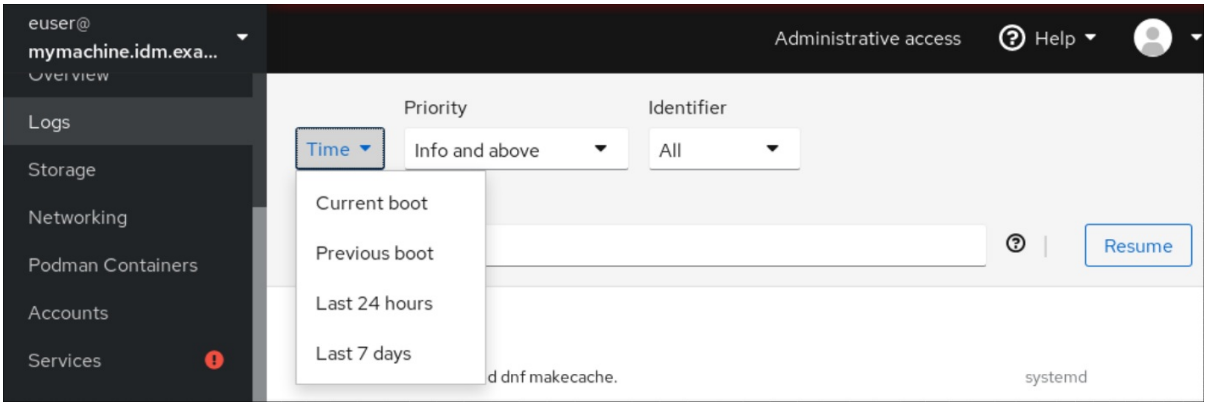
사전 요구 사항

- RHEL 10 웹 콘솔을 설치했습니다.

  자세한 내용은 웹 콘솔 설치 및 활성화를 참조하십시오.

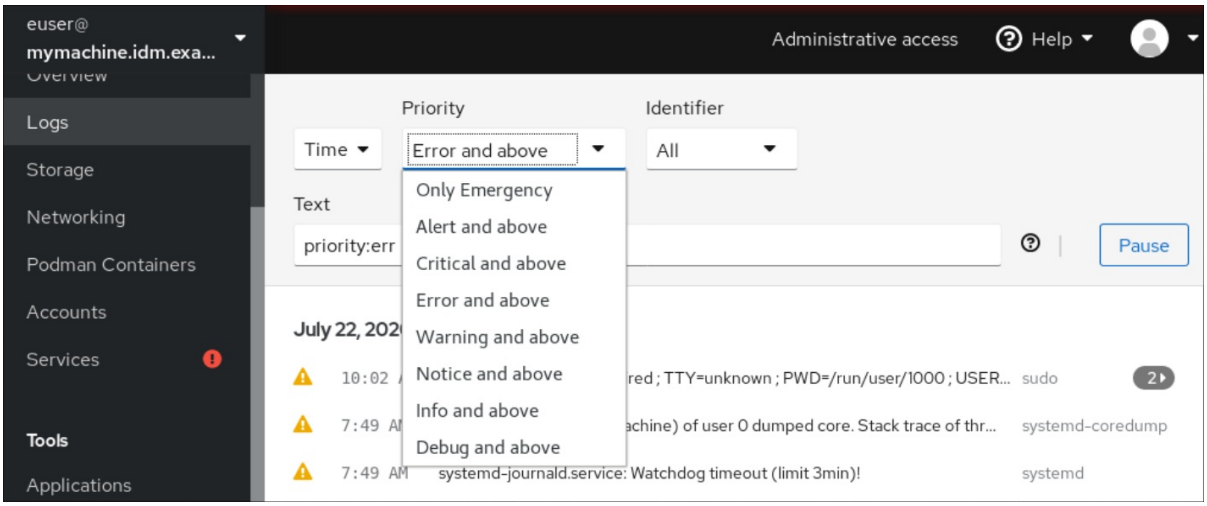
프로세스

1. RHEL 10 웹 콘솔에 로그인합니다.
2. 로그를 클릭합니다.
3. 기본적으로 웹 콘솔은 최신 로그 항목을 표시합니다. 특정 시간 범위별로 필터링하려면 시간 드롭다운 메뉴를 클릭하고 선호하는 옵션을 선택합니다.



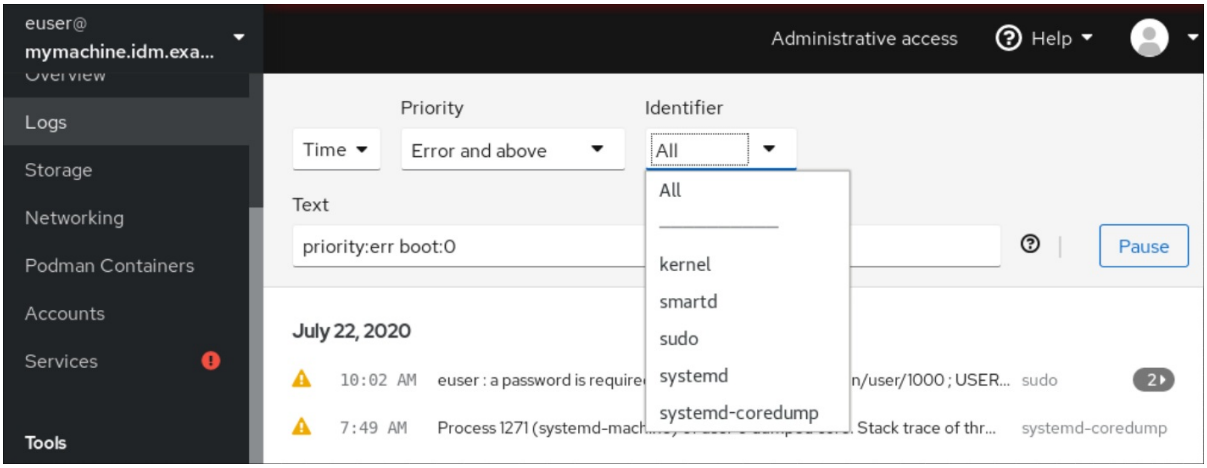
4.

오류 및 심각도 로그 목록이 기본적으로 표시됩니다. 다른 우선 순위로 필터링하려면 **Error and above** 드롭다운 메뉴를 클릭하고 선호하는 우선 순위를 선택합니다.



5.

기본적으로 웹 콘솔은 모든 식별자에 대한 로그를 표시합니다. 특정 식별자에 대한 로그를 필터링하려면 모두 드롭다운 메뉴를 클릭하고 식별자를 선택합니다.



6. 로그 항목을 열려면 선택한 로그를 클릭합니다.

3.3. 웹 콘솔에서 로그 필터링을 위한 텍스트 검색 옵션

텍스트 검색 옵션 기능은 로그 필터링을 위한 다양한 옵션을 제공합니다. 텍스트 검색을 사용하여 로그를 필터링하려면 세 드롭다운 메뉴에 정의된 사전 정의된 옵션을 사용하거나 전체 검색을 직접 입력할 수 있습니다.

드롭다운 메뉴

검색의 기본 매개변수를 지정하는 데 사용할 수 있는 세 가지 드롭다운 메뉴가 있습니다.

- **Time:**이 드롭다운 메뉴에는 다양한 검색 시간 범위에 대한 사전 정의된 검색이 포함되어 있습니다.
- **Priority:**이 드롭다운 메뉴에서는 다양한 우선 순위 수준의 옵션을 제공합니다. **journalctl --priority** 옵션에 해당합니다. 기본 우선순위 값은 **Error** 이상입니다. 다른 우선 순위를 지정하지 않을 때마다 설정됩니다.
- **ID:** 이 드롭다운 메뉴에서는 필터링할 식별자를 선택할 수 있습니다. **journalctl --identifier** 옵션에 해당합니다.

Cryostatifiers

검색을 지정하는 데 사용할 수 있는 **6**가지 한정자가 있습니다. 로그 필터링 테이블에는 옵션이 포함되

어 있습니다.

로그 필드

특정 로그 필드를 검색하려면 해당 콘텐츠와 함께 필드를 지정할 수 있습니다.

로그 메시지에서 자유형 텍스트 검색

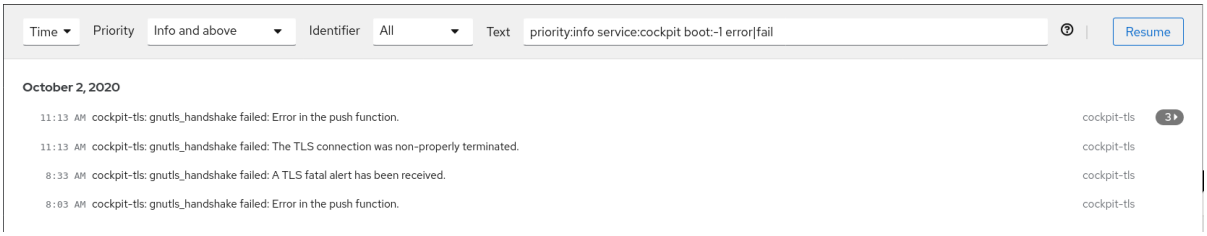
로그 메시지에서 선택한 텍스트 문자열을 필터링할 수 있습니다. 문자열은 정규식 형식일 수도 있습니다.

고급 로그 필터링 I

2020년 10월 22일 자정 이후 발생한 'systemd'로 식별된 모든 로그 메시지를 필터링하고 'JOB_TYPE' 필드는 'start' 또는 'restart'입니다.

1. type identifier:systemd since:2020-10-22 JOB_TYPE=start,restart to search field.

2. 결과를 확인하십시오.

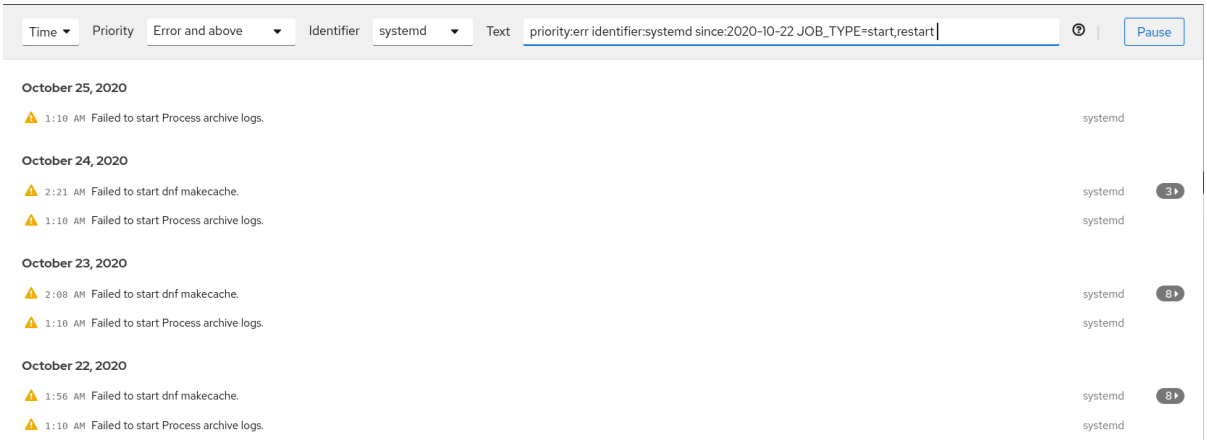


고급 로그 필터링 II

마지막 전에 부팅에 발생한 'cockpit.service' systemd 장치에서 제공되는 모든 로그 메시지를 필터링하고 메시지 본문에는 "error" 또는 "fail"가 포함됩니다.

1. service:cockpit boot:-1 error|fail to the search field를 입력합니다.

2. 결과를 확인하십시오.



3.4. 텍스트 검색 상자를 사용하여 웹 콘솔에서 로그를 필터링

웹 콘솔의 텍스트 검색 상자를 사용하여 다양한 매개변수에 따라 로그를 필터링할 수 있습니다. 검색에서는 필터링 드롭다운 메뉴, 정량자, 로그 필드 및 자유 형식 문자열 검색의 사용을 결합합니다.

사전 요구 사항

- **RHEL 10** 웹 콘솔을 설치했습니다.

  자세한 내용은 웹 콘솔 설치 및 활성화를 참조하십시오.

프로세스

1. **RHEL 10** 웹 콘솔에 로그인합니다.
2. 로그를 클릭합니다.
3. 드롭다운 메뉴를 사용하여 필터링하려는 세 가지 기본 한정자(시간 범위, 우선 순위 및 식별자)를 지정합니다.

   우선 순위 한정자는 항상 값이 있어야 합니다. 이를 지정하지 않으면 **Error** 및 **above** 우선 순위를 자동으로 필터링합니다. 설정한 옵션은 텍스트 검색 상자에 반영됩니다.

4.

필터링할 로그 필드를 지정합니다.

여러 로그 필드를 추가할 수 있습니다.

5.

자유 형식 문자열을 사용하여 다른 항목을 검색할 수 있습니다. 검색 상자에는 정규식도 사용할 수 있습니다.

3.5. 로그 필터링 옵션

웹 콘솔에서 로그를 필터링하는 데 journalctl 옵션을 사용할 수 있습니다. 이러한 옵션 중 일부는 웹 콘솔 인터페이스에서 드롭다운 메뉴의 일부로 제공됩니다.

표 3.1. 표

옵션 이름	사용법	참고
priority	메시지 우선 순위에 따라 출력을 필터링합니다. 단일 숫자 또는 텍스트 로그 수준을 사용합니다. 로그 수준은 일반적인 syslog 로그 수준입니다. 단일 로그 수준이 지정되면 이 로그 수준이 있는 모든 메시지 또는 하위(가장 중요함) 로그 수준이 표시됩니다.	**우선 순위** 드롭다운 메뉴에서 다룹니다.
identifier	지정된 syslog 식별자 SYSLOG_IDENTIFIER에 대한 메시지를 표시합니다. 여러 번 지정할 수 있습니다.	**ID** 드롭다운 메뉴에서 다룹니다.
팔로우	가장 최근의 저널 항목만 표시하고 저널에 추가되는 대로 새 항목을 지속적으로 출력합니다.	드롭다운에는 적용되지 않습니다.
서비스	지정된 **systemd** 장치에 대한 메시지를 표시합니다. 여러 번 지정할 수 있습니다.	드롭다운에 포함되지 않습니다. **journalctl --unit** 매개변수에 해당합니다.

옵션 이름	사용법	참고
부팅	특정 부팅의 메시지를 표시합니다. 양의 정수는 저널의 시작부터 부팅을 조회하고, 동일하게 또는 무탄한 0 정수는 저널이 끝날 때부터 부팅을 조회합니다. 따라서 1은 저널에 있는 첫 번째 부팅을 시간순으로, 2번 이상 두 번째 부팅을 의미합니다. -0은 마지막 부팅이고, 마지막 부팅은 -1이 됩니다.	**현재 부팅** 또는 **시간** 드롭다운 메뉴에서 **이전 부팅** 으로만 적용됩니다. 다른 옵션은 수동으로 작성해야 합니다.
since	지정된 날짜보다 크거나 지정된 날짜보다 항목을 각각 표시하거나 그 이전 날짜를 표시합니다. 날짜 사양은 "2012-10-30 18:17:16" 형식이어야 합니다. 시간 부분이 생략되면 "00:00:00"이 사용됩니다. 초 구성 요소만 생략하면 ":00"이 사용됩니다. 날짜 구성 요소를 생략하면 현재 날짜로 가정합니다. 또는 문자열 "yesterday", "today", "tomorrow"가 이해되며, 이는 현재 날 전날, 현재 날짜 또는 하루 전의 00:00:00을 나타냅니다. "현재"는 현재 시간을 나타냅니다. 마지막으로, 현재 시간 전이나 후에 각각 "-" 또는 "+" 접두사를 지정하여 상대 시간을 지정할 수 있습니다.	드롭다운에는 적용되지 않습니다.

4장. RHEL 시스템 역할을 사용하여 SYSTEMD 저널 구성

journald RHEL 시스템 역할을 사용하면 Red Hat Ansible Automation Platform을 사용하여 systemd 저널을 자동화하고 영구 로깅을 구성할 수 있습니다.

4.1. JOURNALD RHEL 시스템 역할을 사용하여 영구 로깅 구성

기본적으로 systemd 저널은 영구적이지 않은 /run/log/journal 의 작은 링 버퍼에만 로그를 저장합니다. 시스템을 재부팅하면 저널 데이터베이스 로그도 제거됩니다. journald RHEL 시스템 역할을 사용하여 여러 시스템에서 영구 로깅을 일관되게 구성할 수 있습니다.

사전 요구 사항

- 컨트롤 노드 및 관리형 노드를 준비했습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리 노드에 연결하는 데 사용하는 계정에는 sudo 권한이 있습니다.

프로세스

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: ~/playbook.yml)을 생성합니다.

```
---
- name: Configure journald
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure persistent logging
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.journald
      vars:
        journald_persistent: true
        journald_max_disk_size: <size>
        journald_per_user: true
        journald_sync_interval: <interval>
```

예제 플레이북에 지정된 설정은 다음과 같습니다.

journald_persistent: true

영구 로깅을 활성화합니다.

journald_max_disk_size: *<size>*

저널 파일의 최대 디스크 크기를 **MB(예: 2048)**로 지정합니다.

journald_per_user: true

로그 데이터를 각 사용자에 대해 별도로 유지하도록 **journald** 를 구성합니다.

journald_sync_interval: *<interval>*

동기화 간격을 분 단위로 설정합니다**(예: 1).**

플레이북에 사용되는 모든 변수에 대한 자세한 내용은 제어 노드의 **/usr/share/ansible/roles/rhel-system-roles.journald/README.md** 파일을 참조하십시오.

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. **Playbook**을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

5장. 원격 로깅 솔루션 구성

환경의 다양한 시스템의 로그가 로깅 서버에 중앙에 기록되도록 하려면 클라이언트 시스템의 특정 기준에 맞는 로그를 기록하도록 **Rsyslog** 애플리케이션을 구성할 수 있습니다.

5.1. RSYSLOG 로깅 서비스

Rsyslog 애플리케이션은 **systemd-journald** 서비스와 함께 **Red Hat Enterprise Linux**의 로컬 및 원격 로깅 지원을 제공합니다. **rsyslogd** 데몬은 저널에서 **systemd-journald** 서비스에서 수신한 **syslog** 메시지를 지속적으로 읽습니다. 그런 다음 **rsyslogd** 는 이러한 **syslog** 이벤트를 필터링 및 처리하고 **rsyslog** 로그 파일에 기록하거나 구성에 따라 다른 서비스로 전달합니다.

rsyslogd 데몬은 확장된 필터링, 메시지 암호화 보호, 입력 및 출력 모듈, **TCP** 및 **UDP** 프로토콜을 사용한 전송 지원도 제공합니다.

rsyslog 의 기본 구성 파일인 **/etc/rsyslog.conf** 에서는 **rsyslogd** 가 메시지를 처리하는 규칙에 따라 규칙을 지정할 수 있습니다. 일반적으로 메시지를 소스 및 주제(facility) 및 긴급(우선 순위)별로 분류한 다음 메시지가 이러한 기준에 맞는 경우 수행해야 하는 작업을 할당할 수 있습니다.

/etc/rsyslog.conf 에서는 **rsyslogd** 에서 유지 관리하는 로그 파일 목록도 볼 수 있습니다. 대부분의 로그 파일은 **/var/log/** 디렉터리에 있습니다. **httpd** 및 **samba** 와 같은 일부 애플리케이션은 로그 파일을 **/var/log/** 내의 하위 디렉터리에 저장합니다.

추가 리소스

- **rsyslogd(8)** 및 **rsyslog.conf(5)** 시스템의 도움말 페이지
- **/usr/share/doc/rsyslog/html/index.html** 파일에 **rsyslog-doc** 패키지로 설치된 문서

5.2. RSYSLOG 문서 설치

Rsyslog 애플리케이션에는 **https://www.rsyslog.com/doc/** 에서 사용할 수 있는 광범위한 온라인 문서가 있지만 **rsyslog-doc** 문서 패키지를 로컬로 설치할 수도 있습니다.

사전 요구 사항

- 시스템에서 **AppStream** 리포지토리를 활성화했습니다.
- **sudo** 를 사용하여 새 패키지를 설치할 수 있는 권한이 있습니다.

프로세스

- **rsyslog-doc** 패키지를 설치합니다.

```
# dnf install rsyslog-doc
```

검증

- 선택한 브라우저에서 **/usr/share/doc/rsyslog/html/index.html** 파일을 엽니다. 예를 들면 다음과 같습니다.

```
$ firefox /usr/share/doc/rsyslog/html/index.html &
```

5.3. TCP를 통한 원격 로깅을 위한 서버 구성

Rsyslog 애플리케이션을 사용하면 로깅 서버를 실행하고 개별 시스템을 구성하여 로그 파일을 로깅 서버로 보낼 수 있습니다. **TCP**를 통해 원격 로깅을 사용하려면 서버와 클라이언트를 모두 구성합니다. 서버는 하나 이상의 클라이언트 시스템에서 전송한 로그를 수집하고 분석합니다.

Rsyslog 애플리케이션을 사용하면 로그 메시지가 네트워크를 통해 서버로 전달되는 중앙 집중식 로깅 시스템을 유지 관리할 수 있습니다. 서버를 사용할 수 없을 때 메시지 손실을 방지하기 위해 전달 작업에 대한 작업 큐를 구성할 수 있습니다. 이렇게 하면 전송하지 못한 메시지는 서버에 다시 연결할 때까지 로컬에 저장됩니다. 이러한 대기열은 **UDP** 프로토콜을 사용하는 연결에 대해 구성할 수 없습니다.

omfwd 플러그인은 **UDP** 또는 **TCP**를 통한 전달 기능을 제공합니다. 기본 프로토콜은 **UDP**입니다. 플러그인이 내장되어 있으므로 로드할 필요가 없습니다.

기본적으로 **rsyslog** 는 포트 **514** 에서 **TCP**를 사용합니다.

사전 요구 사항

- **rsyslog**는 서버 시스템에 설치됩니다.
- 서버에서 **root** 로 로그인했습니다.
- **semanage** 명령을 사용하여 선택적 단계에 대해 **policycoreutils-python-utils** 패키지가 설치됩니다.
- **firewalld** 서비스가 실행 중입니다.

프로세스

1. 선택 사항: **rsyslog** 트래픽에 다른 포트를 사용하려면 **syslogd_port_t SELinux** 유형을 **port** 에 추가합니다. 예를 들어 포트 **30514** 를 활성화합니다.

```
# semanage port -a -t syslogd_port_t -p tcp 30514
```

2. 선택 사항: **rsyslog** 트래픽에 다른 포트를 사용하려면 해당 포트에서 들어오는 **rsyslog** 트래픽을 허용하도록 **firewalld** 를 구성합니다. 예를 들어 포트 **30514** 에서 **TCP** 트래픽을 허용합니다.

```
# firewall-cmd --zone=<zone_name> --permanent --add-port=30514/tcp
success
# firewall-cmd --reload
```

3. **/etc/rsyslog.d/** 디렉터리에 (예: **remotelog.conf**) 새 파일을 생성하고 다음 내용을 삽입합니다.

```
# Define templates before the rules that use them
# Per-Host templates for remote systems
template(name="TmplAuthpriv" type="list") {
    constant(value="/var/log/remote/auth/")
    property(name="hostname")
    constant(value="/")
    property(name="programname" SecurePath="replace")
    constant(value=".log")
    }

template(name="TmplMsg" type="list") {
    constant(value="/var/log/remote/msg/")
```

```
    property(name="hostname")
    constant(value="/")
    property(name="programname" SecurePath="replace")
    constant(value=".log")
    }

# Provides TCP syslog reception
module(load="imtcp")

# Adding this ruleset to process remote messages
ruleset(name="remote1"){
     authpriv.*   action(type="omfile" DynaFile="TmplAuthpriv")
      *.info;mail.none;authpriv.none;cron.none
action(type="omfile" DynaFile="TmplMsg")
}

input(type="imtcp" port="30514" ruleset="remote1")
```

4. **/etc/rsyslog.d/remotelog.conf 파일에 변경 사항을 저장합니다.**

5. **/etc/rsyslog.conf 파일의 구문을 테스트합니다.**

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-2.el8, config validation run...
rsyslogd: End of config validation run. Bye.
```

6. **로깅 서버에서 rsyslog 서비스가 실행 중이고 활성화되어 있는지 확인합니다.**

```
# systemctl status rsyslog
```

7. **rsyslog 서비스를 다시 시작합니다.**

```
# systemctl restart rsyslog
```

8. **선택 사항: rsyslog 가 활성화되지 않은 경우 재부팅 후 rsyslog 서비스가 자동으로 시작되는지 확인합니다.**

```
# systemctl enable rsyslog
```

이제 환경의 다른 시스템에서 로그 파일을 수신하고 저장하도록 로그 서버가 구성되어 있습니다.

추가 리소스

- **rsyslogd(8), rsyslog.conf(5), semanage(8), semanage(8)** 및 **firewall-cmd(1)** 매뉴얼 페이지
- **/usr/share/doc/rsyslog/html/index.html** 파일에 **rsyslog-doc** 패키지로 설치된 문서

5.4. TCP를 통한 원격 로깅 서버 구성

TCP 프로토콜을 통해 서버로 로그 메시지를 전달하도록 시스템을 구성할 수 있습니다. **omfwd** 플러그인은 **UDP** 또는 **TCP**를 통한 전달 기능을 제공합니다. 기본 프로토콜은 **UDP**입니다. 플러그인이 내장되어 있으므로 로드하지 않아도 됩니다.

사전 요구 사항

- **rsyslog** 패키지는 서버에 보고해야 하는 클라이언트 시스템에 설치됩니다.
- 원격 로깅을 위해 서버를 구성했습니다.
- 지정된 포트는 **SELinux**에서 허용되며 방화벽에서 열려 있습니다.
- 시스템에는 **SELinux** 구성에 비표준 포트를 추가하기 위한 **semanage** 명령을 제공하는 **policycoreutils-python-utils** 패키지가 포함되어 있습니다.

프로세스

1. **/etc/rsyslog.d/** 디렉터리에 **(예: 10-remotelog.conf)** 를 생성하고 다음 내용을 삽입합니다.

```
*.* action(type="omfwd"
     queue.type="linkedlist"
     queue.filename="example_fwd"
     action.resumeRetryCount="-1"
     queue.saveOnShutdown="on"
     target="example.com" port="30514" protocol="tcp"
    )
```

다음과 같습니다.

- **queue.type="linkedlist"** 설정은 **LinkedList in-memory** 큐를 활성화합니다.
- **queue.filename** 설정은 디스크 스토리지를 정의합니다. 백업 파일은 이전 **global workDirectory** 지시문에서 지정한 작업 디렉터리에 **example_fwd** 접두사를 사용하여 생성됩니다.
- **action.resumeRetryCount -1** 설정은 서버가 응답하지 않는 경우 연결을 다시 시도할 때 **rsyslog** 가 메시지를 삭제하지 않도록 합니다.
- **rsyslog** 가 종료되면 **queue.saveOnShutdown="on"** 설정은 메모리 내 데이터를 저장합니다.
- 마지막 줄은 수신된 모든 메시지를 로깅 서버로 전달합니다. 포트 사양은 선택 사항입니다.

이 구성을 사용하면 **rsyslog** 가 서버에 메시지를 전송하지만 원격 서버에 연결할 수 없는 경우 메시지를 메모리에 유지합니다. 디스크의 파일은 **rsyslog** 가 구성된 메모리 대기열 공간이 부족하거나 종료되어야 하는 경우에만 생성되므로 시스템 성능에 도움이 됩니다.

참고

rsyslog는 구성 파일 **/etc/rsyslog.d/** 를 사전순으로 처리합니다.

2. **rsyslog** 서비스를 다시 시작합니다.

```
# systemctl restart rsyslog
```

검증

클라이언트 시스템이 서버에 메시지를 전송하는지 확인하려면 다음을 수행합니다.

1. 클라이언트 시스템에서 테스트 메시지를 보냅니다.

```
# logger test
```

2. 서버 시스템에서 **/var/log/messages** 로그를 확인합니다. 예를 들면 다음과 같습니다.

```
# cat /var/log/remote/msg/hostname/root.log
Feb 25 03:53:17 hostname root[6064]: test
```

여기서 ***hostname*** 은 클라이언트 시스템의 호스트 이름입니다. 로그에는 **logger** 명령을 입력한 사용자의 사용자 이름이 포함되어 있습니다(이 경우 **root**).

추가 리소스

- **rsyslogd(8)** 및 **rsyslog.conf(5)** 시스템의 도움말 페이지
- **/usr/share/doc/rsyslog/html/index.html** 파일에 **rsyslog-doc** 패키지로 설치된 문서

5.5. TLS 암호화 원격 로깅 구성

기본적으로 **Rsyslog**는 일반 텍스트 형식으로 **remote-logging** 통신을 보냅니다. 시나리오에 이 통신 채널을 보안해야 하는 경우 **TLS**를 사용하여 암호화할 수 있습니다.

TLS를 통해 암호화된 전송을 사용하려면 서버와 클라이언트를 모두 구성합니다. 서버는 하나 이상의 클라이언트 시스템에서 전송한 로그를 수집하고 분석합니다.

ossl 네트워크 스트림 드라이버(**OpenSSL**) 또는 **gtls** 스트림 드라이버(**GnuTLS**)를 사용할 수 있습니다.

참고

예를 들어 보안이 높은 별도의 시스템(예: 네트워크에 연결되지 않았거나 더 엄격한 권한이 있는 시스템)이 있는 경우 별도의 시스템을 인증 기관(**CA**)으로 사용합니다.

글로벌 ,모듈 및 입력 수준의 서버 측과 글로벌 및 작업 수준의 클라이언트 측의 스트림 드라이버를 사용하여 연결 설정을 사용자 지정할 수 있습니다. 보다 구체적인 구성은 보다 일반적인 구성을 덮어씁니다. 예를 들어 대부분의 연결에서 **ossl** 을 사용하고 특정 연결에 대해서만 **gtls** 를 사용할 수 있습니다.

사전 요구 사항

- 클라이언트 및 서버 시스템 모두에 대한 루트 액세스 권한이 있습니다.
- 다음 패키지는 서버 및 클라이언트 시스템에 설치됩니다.
  - **rsyslog** 패키지입니다.
  - **ossl** 네트워크 스트림 드라이버의 경우 **rsyslog-openssl** 패키지입니다.
  - **gtls** 네트워크 스트림 드라이버의 경우 **rsyslog-gnutls** 패키지입니다.
  - **certtool** 명령을 사용하여 인증서를 생성하는 경우 **gnutls-utils** 패키지를 사용합니다.
- 로깅 서버에서 다음 인증서는 **/etc/pki/ca-trust/source/anchors/** 디렉터리에 있으며 **update-ca-trust** 명령을 사용하여 시스템 구성이 업데이트됩니다.
  - **ca-cert.pem -** 로깅 서버 및 클라이언트에서 키와 인증서를 확인할 수 있는 **CA** 인증서입니다.
  - **server-cert.pem -** 로깅 서버의 공개 키입니다.
  - **server-key.pem -** 로깅 서버의 개인 키입니다.
- 로깅 클라이언트에서 다음 인증서는 **/etc/pki/ca-trust/source/anchors/** 디렉터리에 있으며 **update-ca-trust:**을 사용하여 시스템 구성이 업데이트됩니다.

- **ca-cert.pem -** 로깅 서버 및 클라이언트에서 키와 인증서를 확인할 수 있는 **CA** 인증서입니다.

- **client-cert.pem -** 클라이언트의 공개 키입니다.

- **client-key.pem -** 클라이언트의 개인 키입니다.

- 서버가 **RHEL 9.2** 이상을 실행하고 **FIPS** 모드가 활성화된 경우 클라이언트는 확장 마스터 시크릿**(Extended Master Secret)** 확장을 지원하거나 **TLS 1.3**을 사용해야 합니다. **TLS 1.2** 연결이 없는 경우 실패합니다. 자세한 내용은 **TLS** 확장 **"Extended Master Secret" enforced article (Red Hat Knowledgebase)**을 참조하십시오.

프로세스

1. 클라이언트 시스템에서 암호화된 로그를 수신하도록 서버를 구성합니다.

a. **/etc/rsyslog.d/** 디렉터리에 새 파일을 만듭니다(예: **securelogser.conf**).

b. 통신을 암호화하려면 구성 파일에 서버의 인증서 파일 경로, 선택한 인증 방법, **TLS** 암호화를 지원하는 스트림 드라이버가 포함되어야 합니다. **/etc/rsyslog.d/securelogser.conf** 파일에 다음 행을 추가합니다.

```
# Set certificate files
global(
  DefaultNetstreamDriverCAFile="/etc/pki/ca-trust/source/anchors/ca-cert.pem"
  DefaultNetstreamDriverCertFile="/etc/pki/ca-trust/source/anchors/server-cert.pem"
  DefaultNetstreamDriverKeyFile="/etc/pki/ca-trust/source/anchors/server-key.pem"
)

# TCP listener
module(
  load="imtcp"
  PermittedPeer=["client1.example.com", "client2.example.com"]
  StreamDriver.AuthMode="x509/name"
  StreamDriver.Mode="1"
  StreamDriver.Name="ossl"
)

# Start up listener at port 514
input(
```

```
  type="imtcp"
  port="514"
)
```

참고

GnuTLS 드라이버를 선호하는 경우 **StreamDriver.Name="gtls"** 구성 옵션을 사용합니다. **x509/name** 보다 덜 엄격한 인증 모드에 대한 자세한 내용은 **rsyslog-doc** 패키지로 설치된 설명서를 참조하십시오.

c. 선택 사항: 연결 구성을 사용자 지정하려면 **input** 섹션을 다음으로 바꿉니다.

```
input(
  type="imtcp"
  Port="50515"
  StreamDriver.Name="<driver>"
  streamdriver.CAFile="/etc/rsyslog.d/<ca1>.pem"
  streamdriver.CertFile="/etc/rsyslog.d/<server1_cert>.pem"
  streamdriver.KeyFile="/etc/rsyslog.d/<server1_key>.pem"
)
```

- 사용하려는 *드라이버에 따라* ***<driver>***를 **ossl** 또는 **gtls** 로 바꿉니다.
- ***<ca1>***을 **CA** 인증서로, ***<server1_cert>***를 인증서로, ***<server1_key>***를 사용자 지정된 연결 키로 바꿉니다.

d. **/etc/rsyslog.d/securelogser.conf** 파일에 변경 사항을 저장합니다.

e. **/etc/rsyslog.conf** 파일 및 **/etc/rsyslog.d/** 디렉터리에 있는 모든 파일의 구문을 확인합니다.

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-2.el8, config validation run (level 1)...
rsyslogd: End of config validation run. Bye.
```

f. 로깅 서버에서 **rsyslog** 서비스가 실행 중이고 활성화되어 있는지 확인합니다.

```
# systemctl status rsyslog
```

g. **rsyslog** 서비스를 다시 시작하십시오.

```
# systemctl restart rsyslog
```

h. 선택 사항: **Rsyslog**가 활성화되지 않은 경우 재부팅 후 **rsyslog** 서비스가 자동으로 시작되는지 확인합니다.

```
# systemctl enable rsyslog
```

2. 암호화된 로그를 서버로 전송하도록 클라이언트를 구성합니다.

a. 클라이언트 시스템에서 **/etc/rsyslog.d/** 디렉터리에 (예: **securelogcli.conf**) 새 파일을 만듭니다.

b. **/etc/rsyslog.d/securelogcli.conf** 파일에 다음 행을 추가합니다.

```
# Set certificate files
global(
  DefaultNetstreamDriverCAFile="/etc/pki/ca-trust/source/anchors/ca-cert.pem"
  DefaultNetstreamDriverCertFile="/etc/pki/ca-trust/source/anchors/client-cert.pem"
  DefaultNetstreamDriverKeyFile="/etc/pki/ca-trust/source/anchors/client-key.pem"
)


# Set up the action for all messages
*.* action(
  type="omfwd"
  StreamDriver="ossl"
  StreamDriverMode="1"
  StreamDriverPermittedPeers="server.example.com"
  StreamDriverAuthMode="x509/name"
  target="server.example.com" port="514" protocol="tcp"
)
```

참고

GnuTLS 드라이버를 선호하는 경우 **StreamDriver.Name="gtls"** 구성 옵션을 사용합니다.

c. **선택 사항: 연결 구성을 사용자 지정하려면 action 섹션을 다음으로 교체합니다.**

```
local1.* action(
  type="omfwd"
  StreamDriver="<driver>"
  StreamDriverMode="1"
  StreamDriverAuthMode="x509/certvalid"
  streamDriver.CAFile="/etc/rsyslog.d/<ca1>.pem"
  streamDriver.CertFile="/etc/rsyslog.d/<client1_cert>.pem"
  streamDriver.KeyFile="/etc/rsyslog.d/<client1_key>.pem"
  target="server.example.com" port="514" protocol="tcp"
  )
```

- **사용하려는 *드라이버에 따라* *<driver>*를 ossl 또는 gtls 로 바꿉니다.**
- **< *ca1* >을 CA 인증서로, < *client1_cert* >를 인증서로, < client1_key >를 사용자 지정 연결의 키로 바꿉니다.**

d. **/etc/rsyslog.d/securelogcli.conf 파일에 변경 사항을 저장합니다.**

e. **/etc/rsyslog.conf 파일 및 기타 파일의 구문을 /etc/rsyslog.d/ 디렉터리에 확인합니다.**

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-2.el8, config validation run (level 1)...
rsyslogd: End of config validation run. Bye.
```

f. **로깅 서버에서 rsyslog 서비스가 실행 중이고 활성화되어 있는지 확인합니다.**

```
# systemctl status rsyslog
```

g. **rsyslog 서비스를 다시 시작하십시오.**

```
# systemctl restart rsyslog
```

h. **선택 사항: Rsyslog가 활성화되지 않은 경우 재부팅 후 rsyslog 서비스가 자동으로 시작되는지 확인합니다.**

```
# systemctl enable rsyslog
```

검증

클라이언트 시스템이 서버에 메시지를 전송하는지 확인하려면 다음 단계를 따르십시오.

1. 클라이언트 시스템에서 테스트 메시지를 보냅니다.

```
# logger test
```

2. 서버 시스템에서 **/var/log/messages** 로그를 확인합니다. 예를 들면 다음과 같습니다.

```
# cat /var/log/remote/msg/<hostname>/root.log
Feb 25 03:53:17 <hostname> root[6064]: test
```

여기서 ***<hostname>***은 클라이언트 시스템의 호스트 이름입니다. 로그에는 **logger** 명령을 입력한 사용자의 사용자 이름이 포함되어 있습니다(이 경우 **root**).

추가 리소스

- **certtool(1), openssl(1), update-ca-trust(8), rsyslogd(8), rsyslog.conf(5)** 도움말 페이지
- **/usr/share/doc/rsyslog/html/index.html**에 **rsyslog-doc** 패키지로 설치된 문서
- TLS에서 로깅 시스템 역할 사용

5.6. UDP를 통해 원격 로깅 정보를 수신하기 위한 서버 구성

Rsyslog 애플리케이션을 사용하면 원격 시스템에서 로깅 정보를 수신할 수 있는 시스템을 구성할 수 있습니다. **UDP**를 통한 원격 로깅을 사용하려면 서버와 클라이언트를 모두 구성합니다. 수신 서버는 하나 이상의 클라이언트 시스템에서 전송한 로그를 수집하고 분석합니다. 기본적으로 **rsyslog** 는 포트 **514** 에서 **UDP**를 사용하여 원격 시스템에서 로그 정보를 수신합니다.

다음 절차에 따라 **UDP** 프로토콜을 통해 하나 이상의 클라이언트 시스템에서 전송한 로그를 수집하고 분석하도록 서버를 구성합니다.

사전 요구 사항

- **rsyslog**는 서버 시스템에 설치됩니다.
- 서버에서 **root** 로 로그인했습니다.
- **semanage** 명령을 사용하는 선택적 단계에 대해 **policycoreutils-python-utils** 패키지가 설치됩니다.
- **firewalld** 서비스가 실행 중입니다.

프로세스

1. 선택 사항: 기본 포트 **514** 이외의 **rsyslog** 트래픽에 다른 포트를 사용하려면 다음을 수행합니다.

   a. **syslogd_port_t SELinux** 유형을 **SELinux** 정책 구성에 추가하고 ***portno*** 를 **rsyslog** 에서 사용할 포트 번호로 바꿉니다.

   ```
   # semanage port -a -t syslogd_port_t -p udp portno
   ```

   b. 들어오는 **rsyslog** 트래픽을 허용하고 ***portno*** 를 포트 번호 및 영역을 **rsyslog** 에서 사용할 영역으로 대체하도록 **firewalld** 를 구성합니다.

   ```
   # firewall-cmd --zone=zone --permanent --add-port=portno/udp
   success
   # firewall-cmd --reload
   ```

   c. 방화벽 규칙을 다시 로드합니다.

   ```
   # firewall-cmd --reload
   ```

2. **/etc/rsyslog.d/** 디렉터리에 새 **.conf** 파일을 만들고 (예: **remotelogserv.conf**) 다음 내용을

삽입합니다.

```
# Define templates before the rules that use them
# Per-Host templates for remote systems
template(name="TmplAuthpriv" type="list") {
    constant(value="/var/log/remote/auth/")
    property(name="hostname")
    constant(value="/")
    property(name="programname" SecurePath="replace")
    constant(value=".log")
    }

template(name="TmplMsg" type="list") {
    constant(value="/var/log/remote/msg/")
    property(name="hostname")
    constant(value="/")
    property(name="programname" SecurePath="replace")
    constant(value=".log")
    }

# Provides UDP syslog reception
module(load="imudp")

# This ruleset processes remote messages
ruleset(name="remote1"){
     authpriv.*   action(type="omfile" DynaFile="TmplAuthpriv")
      *.info;mail.none;authpriv.none;cron.none
action(type="omfile" DynaFile="TmplMsg")
}

input(type="imudp" port="514" ruleset="remote1")
```

여기서 **514** 는 기본적으로 **rsyslog** 에서 사용하는 포트 번호입니다. 대신 다른 포트를 지정할 수 있습니다.

3. **/etc/rsyslog.conf** 파일과 **/etc/rsyslog. d/** 디렉터리의 모든 **. conf** 파일의 구문을 확인합니다.

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-2.el8, config validation run...
```

4. **rsyslog** 서비스를 다시 시작합니다.

```
# systemctl restart rsyslog
```

5. 선택 사항**: rsyslog** 가 활성화되지 않은 경우 재부팅 후 **rsyslog** 서비스가 자동으로 시작되는지 확인합니다.

```
# systemctl enable rsyslog
```

추가 리소스

- **rsyslogd(8) , rsyslog.conf(5), semanage(8), semanage(8)** 및 **firewall-cmd(1)** 매뉴얼 페이지
- **/usr/share/doc/rsyslog/html/index.html** 파일에 **rsyslog-doc** 패키지로 설치된 문서

5.7. UDP를 통해 서버에 원격 로깅 구성

UDP 프로토콜을 통해 서버로 로그 메시지를 전달하도록 시스템을 구성할 수 있습니다. **omfwd** 플러그인은 **UDP** 또는 **TCP**를 통한 전달 기능을 제공합니다. 기본 프로토콜은 **UDP**입니다. 플러그인이 내장되어 있으므로 로드하지 않아도 됩니다.

사전 요구 사항

- **rsyslog** 패키지는 서버에 보고해야 하는 클라이언트 시스템에 설치됩니다.
- **UDP**를 통해 원격 로깅 정보를 수신하기 위해 서버 구성에 설명된 대로 원격 로깅을 위해 서버를 구성했습니다.

프로세스

1. **/etc/rsyslog.d/** 디렉터리에 새 **.conf** 파일을 만들고(예: **10-remotelogcli.conf**) 다음 내용을 삽입합니다.

```
*.* action(type="omfwd"
     queue.type="linkedlist"
     queue.filename="example_fwd"
     action.resumeRetryCount="-1"
     queue.saveOnShutdown="on"
     target="example.com" port="portno" protocol="udp"
    )
```

다음과 같습니다.

- **queue.type="linkedlist"** 설정은 **LinkedList in-memory** 큐를 활성화합니다.

- **queue.filename** 설정은 디스크 스토리지를 정의합니다. 백업 파일은 이전 **global workDirectory** 지시문에서 지정한 작업 디렉터리에 ***example_fwd*** 접두사를 사용하여 생성됩니다.

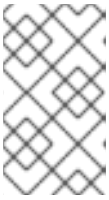
- **action.resumeRetryCount -1** 설정은 서버가 응답하지 않는 경우 연결을 다시 시도할 때 **rsyslog** 가 메시지를 삭제하지 않도록 합니다.

- **rsyslog** 가 종료되면 활성화된 **queue.saveOnShutdown="on"** 설정은 메모리 내 데이터를 저장합니다.

- ***portno*** 값은 **rsyslog** 에서 사용할 포트 번호입니다. 기본값은 **514** 입니다.

- 마지막 줄은 수신된 모든 메시지를 로깅 서버로 전달하며 포트 사양은 선택 사항입니다.

이 구성을 사용하면 **rsyslog** 가 서버에 메시지를 전송하지만 원격 서버에 연결할 수 없는 경우 메시지를 메모리에 유지합니다. 디스크의 파일은 **rsyslog** 가 구성된 메모리 대기열 공간이 부족하거나 종료되어야 하는 경우에만 생성되므로 시스템 성능에 도움이 됩니다.



참고

rsyslog는 구성 파일 **/etc/rsyslog.d/** 를 사전순으로 처리합니다.

2. **rsyslog** 서비스를 다시 시작합니다.

```
# systemctl restart rsyslog
```

3. 선택 사항: **rsyslog** 가 활성화되지 않은 경우 재부팅 후 **rsyslog** 서비스가 자동으로 시작되는지 확인합니다.

```
# systemctl enable rsyslog
```

검증

클라이언트 시스템이 서버에 메시지를 전송하는지 확인하려면 다음 단계를 따르십시오.

1. 클라이언트 시스템에서 테스트 메시지를 보냅니다.

```
# logger test
```

2. 서버 시스템에서 **/var/log/remote/msg/***호스트 이름***/root.log** 로그를 확인합니다. 예를 들면 다음과 같습니다.

```
# cat /var/log/remote/msg/hostname/root.log
Feb 25 03:53:17 hostname root[6064]: test
```

여기서 **hostname** 은 클라이언트 시스템의 호스트 이름입니다. 로그에는 **logger** 명령을 입력한 사용자의 사용자 이름이 포함되어 있습니다(이 경우 **root**).

추가 리소스

- **rsyslogd(8)** 및 **rsyslog.conf(5)** 도움말 페이지.
- **/usr/share/doc/rsyslog/html/index.html** 에 **rsyslog-doc** 패키지로 설치된 문서 .

5.8. RSYSLOG의 로드 밸런싱 도우미

클러스터에서 사용하는 경우 **RebindInterval** 설정을 수정하여 **Rsyslog** 로드 밸런싱을 개선할 수 있습니다.

RebindInterval 은 현재 연결이 끊어지고 다시 설정된 간격을 지정합니다. 이 설정은 **TCP, UDP** 및 **RELP** 트래픽에 적용됩니다. 로드 밸런서는 이를 새로운 연결로 인식하고 메시지를 다른 물리적 대상 시스템으로 전달합니다.

RebindInterval 은 대상 시스템이 **IP** 주소를 변경한 경우 유용합니다. **Rsyslog** 애플리케이션은 연결이 설정될 때 **IP** 주소를 캐시하므로 메시지가 동일한 서버로 전송됩니다. **IP** 주소가 변경되면 **Rsyslog** 서비

스가 다시 시작될 때까지 **UDP** 패킷이 손실됩니다. 연결을 다시 설정하면 **IP**가 **DNS**에서 다시 확인됩니다.

TCP, UDP 및 **RELP** 트래픽에 대한 **RebindInterval** 사용 예

```
action(type="omfwd" protocol="tcp" RebindInterval="250" target="example.com" port="514" …)
action(type="omfwd" protocol="udp" RebindInterval="250" target="example.com" port="514" …)
action(type="omrelp" RebindInterval="250" target="example.com" port="6514" …)
```

5.9. 안정적인 원격 로깅 구성

RELP(Reliable Event Logging Protocol)를 사용하면 메시지 손실 위험이 크게 감소하여 **TCP**를 통해 **syslog** 메시지를 보내고 받을 수 있습니다. **RELP**는 이벤트 메시지를 안정적으로 전달하므로 메시지 손실이 허용되지 않는 환경에서 유용합니다. **RELP**를 사용하려면 서버에서 실행되고 로그를 수신하는 **imrelp** 입력 모듈과 클라이언트에서 실행되고 로그를 로깅 서버로 보내는 **omrelp** 출력 모듈을 구성합니다.

사전 요구 사항

- **rsyslog,librelp** 및 **rsyslog-relp** 패키지를 서버와 클라이언트 시스템에 설치했습니다.
- 지정된 포트는 **SELinux**에서 허용되며 방화벽에서 열려 있습니다.

프로세스

1. 안정적인 원격 로깅을 위해 클라이언트 시스템을 구성합니다.
   a. 클라이언트 시스템에서 **/etc/rsyslog.d/** 디렉터리에 **(예: relpclient .conf)** 새 **.conf** 파일을 생성하고 다음 내용을 삽입합니다.

```
module(load="omrelp")
*.* action(type="omrelp" target="_target_IP_" port="_target_port_")
```

다음과 같습니다.

- *target_IP* 는 로깅 서버의 IP 주소입니다.
- *target_port* 는 로깅 서버의 포트입니다.

b. /etc/rsyslog.d/relpclient.conf 파일에 변경 사항을 저장합니다.

c. rsyslog 서비스를 다시 시작합니다.

```
# systemctl restart rsyslog
```

d. 선택 사항: rsyslog 가 활성화되지 않은 경우 재부팅 후 rsyslog 서비스가 자동으로 시작되는지 확인합니다.

```
# systemctl enable rsyslog
```

2. 안정적인 원격 로깅을 위해 서버 시스템을 구성합니다.

a. 서버 시스템에서 /etc/rsyslog.d/ 디렉터리에 (예: relpserv .conf) 새 .conf 파일을 생성하고 다음 내용을 삽입합니다.

```
ruleset(name="relp"){
*.* action(type="omfile" file="_log_path_")
}


module(load="imrelp")
input(type="imrelp" port="_target_port_" ruleset="relp")
```

다음과 같습니다.

- *log_path* 는 메시지를 저장하는 경로를 지정합니다.

- ***target_port*** 는 로깅 서버의 포트입니다. 클라이언트 구성 파일과 동일한 값을 사용합니다.

b. **/etc/rsyslog.d/relpserv.conf** 파일에 변경 사항을 저장합니다.

c. **rsyslog** 서비스를 다시 시작합니다.

```
# systemctl restart rsyslog
```

d. 선택 사항: **rsyslog** 가 활성화되지 않은 경우 재부팅 후 **rsyslog** 서비스가 자동으로 시작되는지 확인합니다.

```
# systemctl enable rsyslog
```

검증

클라이언트 시스템이 서버에 메시지를 전송하는지 확인하려면 다음을 수행합니다.

1. 클라이언트 시스템에서 테스트 메시지를 보냅니다.

```
# logger test
```

2. 서버 시스템에서 지정된 ***log_path*** 에서 로그를 확인합니다. 예를 들면 다음과 같습니다.

```
# cat /var/log/remote/msg/hostname/root.log
Feb 25 03:53:17 hostname root[6064]: test
```

여기서 **hostname** 은 클라이언트 시스템의 호스트 이름입니다. 로그에는 **logger** 명령을 입력한 사용자의 사용자 이름이 포함되어 있습니다(이 경우 **root**).

추가 리소스

- **rsyslogd(8)** 및 **rsyslog.conf(5)** 시스템의 도움말 페이지

- /usr/share/doc/rsyslog/html/index.html 파일에 rsyslog-doc 패키지로 설치된 문서

5.10. 지원되는 RSYSLOG 모듈

Rsyslog 애플리케이션의 기능을 확장하려면 특정 모듈을 사용할 수 있습니다. 모듈은 추가 입력(Input Modules), 출력(출력 모듈) 및 기타 기능을 제공합니다. 모듈은 모듈을 로드한 후 사용할 수 있는 추가 구성 지시문을 제공할 수도 있습니다.

다음 명령을 입력하여 시스템에 설치된 입력 및 출력 모듈을 나열할 수 있습니다.

```
# ls /usr/lib64/rsyslog/{i,o}m*
```

rsyslog -doc 패키지를 설치한 후 /usr/share/doc/rsyslog/html/configuration/modules/idx_output.html 파일에서 사용 가능한 모든 rsyslog 모듈 목록을 볼 수 있습니다.

5.11. 커널 메시지를 원격 호스트에 기록하도록 NETCONSOLE 서비스 구성

디스크에 로깅하거나 직렬 콘솔을 사용할 수 없는 경우 netconsole 커널 모듈과 동일한 이름을 사용하여 네트워크를 통해 커널 메시지를 원격 rsyslog 서비스에 기록할 수 있습니다.

사전 요구 사항

- rsyslog 와 같은 시스템 로그 서비스는 원격 호스트에 설치됩니다.
- 원격 시스템 로그 서비스는 이 호스트에서 들어오는 로그 항목을 수신하도록 구성됩니다.

프로세스

1. netconsole-service 패키지를 설치합니다.

```
# dnf install netconsole-service
```

2. /etc/sysconfig/netconsole 파일을 편집하고 SYSLOGADDR 매개변수를 원격 호스트의 IP

주소로 설정합니다.

```
# SYSLOGADDR=192.0.2.1
```

3. **netconsole** 서비스를 활성화하고 시작합니다.

```
# systemctl enable --now netconsole
```

검증

- 원격 시스템 로그 서버에 **/var/log/messages** 파일을 표시합니다.

5.12. 추가 리소스

- **/usr/share/doc/rsyslog/html/index.html** 파일에 **rsyslog-doc** 패키지로 설치된 문서
- **rsyslog.conf(5)** 및 **rsyslogd(8)** 도움말 페이지
- 저널링 없이 시스템 로깅 구성 또는 최소한의 **journald** 사용 **(Red Hat Knowledgebase)**
- **RHEL**의 기본 로깅 설정이 성능 및 완화에 미치는 영향 **(Red Hat Knowledgebase)**

6장. RHEL 시스템 역할을 사용하여 로깅 구성

로깅 RHEL 시스템 역할을 사용하여 로컬 및 원격 호스트를 자동화된 방식으로 로깅 서버로 구성하여 많은 클라이언트 시스템에서 로그를 수집할 수 있습니다.

로깅 솔루션은 여러 로그 및 여러 로깅 출력을 읽는 방법을 제공합니다.

예를 들어 로깅 시스템은 다음과 같은 입력을 수신할 수 있습니다.

- 로컬 파일
- systemd/journal
- 네트워크를 통한 다른 로깅 시스템

또한 로깅 시스템에는 다음과 같은 출력이 있을 수 있습니다.

- /var/log/ 디렉터리의 로컬 파일에 저장된 로그
- Elasticsearch 엔진에 전송된 로그
- 다른 로깅 시스템으로 전달된 로그

로깅 RHEL 시스템 역할을 사용하면 입력 및 출력을 결합하여 시나리오에 적합할 수 있습니다. 예를 들어 저널 의 입력을 로컬 파일에 저장하는 로깅 솔루션을 구성할 수 있지만 파일에서 읽은 입력은 다른 로깅 시스템으로 전달되어 로컬 로그 파일에 저장됩니다.

6.1. 로깅 RHEL 시스템 역할을 사용하여 로컬 로그 메시지 필터링

로깅 RHEL 시스템 역할의 속성 기반 필터를 사용하여 다양한 조건에 따라 로컬 로그 메시지를 필터링

할 수 있습니다. 예를 들어 다음을 수행할 수 있습니다.

- 로그 명확성: 트래픽이 많은 환경에서 로그가 빠르게 증가할 수 있습니다. 오류와 같은 특정 메시지에 중점을 두면 문제를 보다 신속하게 식별하는 데 도움이 될 수 있습니다.
- 최적화된 시스템 성능: 과도한 양의 로그는 일반적으로 시스템 성능 저하와 연결됩니다. 중요한 이벤트에 대해서만 선택적 로깅을 수행하면 리소스 소모를 방지할 수 있으므로 시스템이 더 효율적으로 실행될 수 있습니다.
- 향상된 보안: 시스템 오류 및 실패한 로그인과 같은 보안 메시지를 통한 필터링은 관련 로그만 캡처하는 데 도움이 됩니다. 이는 위반을 탐지하고 규정 준수 표준을 준수하는 데 중요합니다.

사전 요구 사항

- 컨트롤 노드 및 관리형 노드를 준비했습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리 노드에 연결하는 데 사용하는 계정에는 **sudo** 권한이 있습니다.

프로세스

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: ~/**playbook.yml**)을 생성합니다.

```
---
- name: Deploy the logging solution
  hosts: managed-node-01.example.com
  tasks:
    - name: Filter logs based on a specific value they contain
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.logging
      vars:
        logging_inputs:
          - name: files_input
            type: basics
        logging_outputs:
          - name: files_output0
            type: files
            property: msg
```

```
      property_op: contains
      property_value: error
      path: /var/log/errors.log
    - name: files_output1
      type: files
      property: msg
      property_op: "!contains"
      property_value: error
      path: /var/log/others.log
  logging_flows:
    - name: flow0
      inputs: [files_input]
      outputs: [files_output0, files_output1]
```

예제 플레이북에 지정된 설정은 다음과 같습니다.

logging_inputs

로깅 입력 사전 목록을 정의합니다. 유형: 기본 옵션은 **systemd** 저널 또는 **Unix** 소켓의 입력을 다룹니다.

logging_outputs

로깅 출력 사전 목록을 정의합니다. **type: files** 옵션은 일반적으로 **/var/log/** 디렉터리에 있는 로컬 파일에 로그 저장을 지원합니다. 속성: **msg; property: contains;** 및 **property_value:** 오류 옵션은 오류 문자열이 포함된 모든 로그가 **/var/log/errors.log** 파일에 저장되도록 지정합니다. 속성: **msg;** 속성: **!contains;** 및 **property_value:** 오류 옵션은 다른 모든 로그가 **/var/log/others.log** 파일에 저장되도록 지정합니다. 오류 값을 필터링할 문자열로 교체할 수 있습니다.

logging_flows

logging_inputs 와 **logging_outputs** 간의 관계를 지정하는 로깅 흐름 사전 목록을 정의합니다. **inputs: [files_input]** 옵션은 로그 처리가 시작되는 입력 목록을 지정합니다. **outputs: [files_output0, files_output1]** 옵션은 로그가 전송되는 출력 목록을 지정합니다.

플레이북에서 사용되는 모든 변수와 **rsyslog** 에 대한 자세한 내용은 제어 노드의 **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** 파일 및 **rsyslog.conf(5)** 및 **syslog Cryostat** 매뉴얼 페이지를 참조하십시오.

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. Playbook을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

검증

1. 관리 노드에서 /etc/rsyslog.conf 파일의 구문을 테스트합니다.

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run...
rsyslogd: End of config validation run. Bye.
```

2. 관리 노드에서 오류 문자열이 포함된 메시지를 로그에 전송하는지 확인합니다.

a. 테스트 메시지를 보냅니다.

```
# logger error
```

b. /var/log/errors.log 로그를 확인합니다. 예를 들면 다음과 같습니다.

```
# cat /var/log/errors.log
Aug  5 13:48:31 hostname root[6778]: error
```

여기서 *hostname* 은 클라이언트 시스템의 호스트 이름입니다. 로그에는 logger 명령을 입력한 사용자의 사용자 이름이 포함되어 있습니다(이 경우 root).

6.2. 로깅 RHEL 시스템 역할을 사용하여 원격 로깅 솔루션 적용

로깅 RHEL 시스템 역할을 사용하여 하나 이상의 클라이언트가 systemd-journal 서비스에서 로그를 가져와서 원격 서버로 전달하는 원격 로깅 솔루션을 구성할 수 있습니다. 서버는 remote_rsyslog 및 remote_files 구성에서 원격 입력을 수신하고 원격 호스트 이름으로 이름이 지정된 디렉터리의 로컬 파일에 로그를 출력합니다.

따라서 다음과 같은 경우 필요한 사용 사례를 처리할 수 있습니다.

- 중앙 집중식 로그 관리: 단일 스토리지 지점에서 여러 시스템의 로그 메시지를 수집, 액세스 및 관리하면 일상적인 모니터링 및 문제 해결 작업을 단순화할 수 있습니다. 또한 이 사용 사례로 로그 메시지를 확인하기 위해 개별 시스템에 로그인할 필요가 줄어듭니다.
- 보안 강화: 한 중앙에 로그 메시지를 저장하면 안전하고 변조된 환경에 있을 가능성이 높아집니다. 이러한 환경을 통해 보안 문제를 보다 효과적으로 감지하고 대응하고 감사 요구 사항을 충족할 수 있습니다.
- 로그 분석의 효율성 개선: 여러 시스템의 로그 메시지를 교정하는 것이 여러 머신 또는 서비스에 걸쳐 있는 복잡한 문제를 신속하게 해결하는 데 중요합니다. 이렇게 하면 다양한 소스의 이벤트를 신속하게 분석하고 참조할 수 있습니다.

사전 요구 사항

- 컨트롤 노드 및 관리형 노드를 준비했습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리 노드에 연결하는 데 사용하는 계정에는 **sudo** 권한이 있습니다.
- 서버 또는 클라이언트 시스템의 **SELinux** 정책에 포트를 정의하고 해당 포트에 대한 방화벽을 엽니다. 기본 **SELinux** 정책에는 포트 **601, 514, 6514, 10514** 및 **20514**가 포함됩니다. 다른 포트를 사용하려면 클라이언트 및 서버 시스템에서 **SELinux** 정책 수정을 참조하십시오.

프로세스

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: **~/playbook.yml**)을 생성합니다.

```
---
- name: Deploy the logging solution
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure the server to receive remote input
      ansible.builtin.include_role:
```

```
      name: redhat.rhel_system_roles.logging
    vars:
      logging_inputs:
        - name: remote_udp_input
          type: remote
          udp_ports: [ 601 ]
        - name: remote_tcp_input
          type: remote
          tcp_ports: [ 601 ]
      logging_outputs:
        - name: remote_files_output
          type: remote_files
      logging_flows:
        - name: flow_0
          inputs: [remote_udp_input, remote_tcp_input]
          outputs: [remote_files_output]

- name: Deploy the logging solution
  hosts: managed-node-02.example.com
  tasks:
    - name: Configure the server to output the logs to local files in directories named by
remote host names
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.logging
      vars:
        logging_inputs:
          - name: basic_input
            type: basics
        logging_outputs:
          - name: forward_output0
            type: forwards
            severity: info
            target: <host1.example.com>
            udp_port: 601
          - name: forward_output1
            type: forwards
            facility: mail
            target: <host1.example.com>
            tcp_port: 601
        logging_flows:
          - name: flows0
            inputs: [basic_input]
            outputs: [forward_output0, forward_output1]
```

예제 플레이북의 첫 번째 플레이에 지정된 설정은 다음과 같습니다.

logging_inputs

로깅 입력 사전 목록을 정의합니다. **type: remote** 옵션은 네트워크를 통해 다른 로깅 시스템의 원격 입력을 다룹니다. **udp_ports: [601]** 옵션은 모니터링할 **UDP** 포트 번호 목록을 정의합니다. **tcp_ports: [601]** 옵션은 모니터링할 **TCP** 포트 번호 목록을 정의합니다.

udp_ports 및 **tcp_ports** 가 모두 설정된 경우 **udp_ports** 가 사용되고 **tcp_ports** 가 삭제됩니다.

logging_outputs

로깅 출력 사전 목록을 정의합니다. **type: remote_files** 옵션은 원격 호스트별로 로컬 파일에 출력 저장소 로그를 만들고 프로그램 이름이 로그를 생성했습니다.

logging_flows

logging_inputs 와 **logging_outputs** 간의 관계를 지정하는 로깅 흐름 사전 목록을 정의합니다. 입력: **[remote_udp_input, remote_tcp_input]** 옵션은 로그 처리가 시작되는 입력 목록을 지정합니다. **outputs: [remote_files_output]** 옵션은 로그를 전송할 출력 목록을 지정합니다.

예제 플레이북의 두 번째 플레이에 지정된 설정은 다음과 같습니다.

logging_inputs

로깅 입력 사전 목록을 정의합니다. 유형: 기본 옵션은 **systemd** 저널 또는 **Unix** 소켓의 입력을 다룹니다.

logging_outputs

로깅 출력 사전 목록을 정의합니다. **type: forward** 옵션은 네트워크를 통해 원격 로깅 서버로 로그 전송을 지원합니다. **severity: info** 옵션은 정보 중요도의 로그 메시지를 나타냅니다. **facility: mail** 옵션은 로그 메시지를 생성하는 시스템 프로그램의 유형을 나타냅니다. **target: <** ***host1.example.com*** **>** 옵션은 원격 로깅 서버의 호스트 이름을 지정합니다. **udp_port: 601/tcp_port: 601** 옵션은 원격 로깅 서버가 수신 대기하는 **UDP/TCP** 포트를 정의합니다.

logging_flows

logging_inputs 와 **logging_outputs** 간의 관계를 지정하는 로깅 흐름 사전 목록을 정의합니다. **inputs: [basic_input]** 옵션은 로그 처리가 시작되는 입력 목록을 지정합니다. **outputs: [forward_output0, forward_output1]** 옵션은 로그가 전송되는 출력 목록을 지정합니다.

역할 변수 및 **rsyslog** 에 대한 자세한 내용은 제어 노드의 **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** 파일 및 **rsyslog.conf(5)** 및 **syslog Cryostat** 매뉴얼 페이지를 참조하십시오.

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. Playbook을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

검증

1. 클라이언트 및 서버 시스템 모두에서 /etc/rsyslog.conf 파일의 구문을 테스트합니다.

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run (level 1), master config
/etc/rsyslog.conf
rsyslogd: End of config validation run. Bye.
```

2. 클라이언트 시스템이 서버에 메시지를 전송하는지 확인합니다.

a. 클라이언트 시스템에서 테스트 메시지를 보냅니다.

```
# logger test
```

b. 서버 시스템에서 /var/log/ *<host2.example.com>* /messages 로그를 확인합니다. 예를 들면 다음과 같습니다.

```
# cat /var/log/<host2.example.com>/messages
Aug  5 13:48:31 <host2.example.com> root[6778]: test
```

여기서 *<host2.example.com >*은 클라이언트 시스템의 호스트 이름입니다. 로그에는 logger 명령을 입력한 사용자의 사용자 이름이 포함되어 있습니다(이 경우 root).

6.3. TLS에서 로깅 RHEL 시스템 역할 사용

TLS(Transport Layer Security)는 컴퓨터 네트워크를 통해 보안 통신을 허용하도록 설계된 암호화 프로토콜입니다.

로깅 **RHEL** 시스템 역할을 사용하여 하나 이상의 클라이언트가 **systemd-journal** 서비스에서 로그를 가져와서 **TLS**를 사용하는 동안 원격 서버로 전송하는 로그 메시지의 보안 전송을 구성할 수 있습니다.

일반적으로 원격 로깅 솔루션에서 로그를 전송하기 위한 **TLS**는 신뢰할 수 없는 또는 인터넷과 같은 공용 네트워크를 통해 중요한 데이터를 전송할 때 사용됩니다. 또한 **TLS**에서 인증서를 사용하여 클라이언트가 로그를 정확하고 신뢰할 수 있는 서버로 전달하도록 할 수 있습니다. 이렇게 하면 **"man-in-the-middle"**와 같은 공격을 방지할 수 있습니다.

6.3.1. TLS를 사용하여 클라이언트 로깅 구성

로깅 **RHEL** 시스템 역할을 사용하여 **RHEL** 클라이언트에 대한 로깅을 구성하고 **TLS** 암호화를 사용하여 원격 로깅 시스템으로 로그를 전송할 수 있습니다.

이 절차에서는 개인 키와 인증서를 생성합니다. 다음으로 **Ansible** 인벤토리의 **clients** 그룹에 있는 모든 호스트에 **TLS**를 구성합니다. **TLS** 프로토콜은 네트워크를 통한 로그의 보안 전송을 위해 메시지 전송을 암호화합니다.

참고

인증서를 생성하기 위해 플레이북에서 인증서 **RHEL** 시스템 역할을 호출할 필요가 없습니다. 로깅 **RHEL** 시스템 역할은 **logging_certificates** 변수가 설정된 경우 자동으로 호출합니다.

CA에서 생성된 인증서에 서명하려면 관리형 노드를 **IdM** 도메인에 등록해야 합니다.

사전 요구 사항

- 컨트롤 노드 및 관리형 노드를 준비했습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.

- 관리 노드에 연결하는 데 사용하는 계정에는 **sudo** 권한이 있습니다.

- 관리형 노드는 **IdM** 도메인에 등록됩니다.

- 관리 노드에서 로깅 서버를 구성하려는 로깅 서버가 **RHEL 9.2** 이상을 실행하고 **FIPS** 모드가 활성화된 경우 클라이언트는 확장 마스터 시크릿(**Extended Master Secret**) 확장을 지원하거나 **TLS 1.3**을 사용해야 합니다. **TLS 1.2** 연결이 없는 경우 실패합니다. 자세한 내용은 **Red Hat Knowledgebase** 솔루션 **TLS** 확장 "확장 마스터 시크릿" 적용 을 참조하십시오.

프로세스

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: ~/**playbook.yml**)을 생성합니다.

```
---
- name: Configure remote logging solution by using TLS for secure transfer of logs
  hosts: managed-node-01.example.com
  tasks:
    - name: Deploying files input and forwards output with certs
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.logging
      vars:
        logging_certificates:
          - name: logging_cert
            dns: ['www.example.com']
            ca: ipa
            principal: "logging/{{ inventory_hostname }}@IDM.EXAMPLE.COM"
        logging_pki_files:
          - ca_cert: /local/path/to/ca_cert.pem
            cert: /local/path/to/logging_cert.pem
            private_key: /local/path/to/logging_cert.pem
        logging_inputs:
          - name: input_name
            type: files
            input_log_path: /var/log/containers/*.log
        logging_outputs:
          - name: output_name
            type: forwards
            target: your_target_host
            tcp_port: 514
            tls: true
            pki_authmode: x509/name
            permitted_server: 'server.example.com'
        logging_flows:
          - name: flow_name
            inputs: [input_name]
            outputs: [output_name]
```

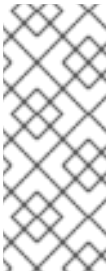
예제 플레이북에 지정된 설정은 다음과 같습니다.

logging_certificates

이 매개변수의 값은 인증서 **RHEL** 시스템 역할의 **certificate _requests** 에 전달되며 개인 키와 인증서를 생성하는 데 사용됩니다.

logging_pki_files

이 매개변수를 사용하여 로깅에서 사용하는 경로 및 기타 설정을 구성하여 로깅에서 사용하는 **CA,** 인증서 및 **TLS**에 사용되는 키 파일을 찾습니다.**ca_cert_src, ca_cert _src ,cert,cert_src, private_key ,private_key _src, tls.**



참고

logging_certificates 를 사용하여 관리 노드에서 파일을 생성하는 경우 **logging_certificates** 에서 생성되지 않은 파일을 복사하는 데 사용되는 **ca_cert_src,cert_src** 및 **private_key_src** 를 사용하지 마십시오.

ca_cert

관리 노드의 **CA** 인증서 파일의 경로를 나타냅니다. 기본 경로는 **/etc/pki/tls/certs/ca.pem** 이며 파일 이름은 사용자가 설정합니다.

인증서

관리 노드의 인증서 파일의 경로를 나타냅니다. 기본 경로는 **/etc/pki/tls/certs/server-cert.pem** 이며 파일 이름은 사용자가 설정합니다.

private_key

관리 노드의 개인 키 파일의 경로를 나타냅니다. 기본 경로는 **/etc/pki/tls/private/server-key.pem** 이며 파일 이름은 사용자가 설정합니다.

ca_cert_src

ca_cert 에서 지정한 위치로 대상 호스트에 복사되는 제어 노드의 **CA** 인증서 파일의 경로를 나타냅니다. **logging_certificates** 를 사용하는 경우 이 사용하지 마십시오.

cert_src

인증서에서 지정한 위치로 대상 호스트에 복사되는 제어 노드의 인증서 파일의 경로를 나타냅니다. **logging_certificates** 를 사용하는 경우 이 사용하지 마십시오.

private_key_src

private_key 에서 지정한 위치로 대상 호스트에 복사되는 제어 노드에서 개인 키 파일의 경로를 나타냅니다. **logging_certificates** 를 사용하는 경우 이 사용하지 마십시오.

tls

이 매개변수를 **true** 로 설정하면 네트워크를 통한 로그 전송이 안전합니다. 보안 래퍼가 필요하지 않은 경우 **tls: false** 를 설정할 수 있습니다.

역할 변수 및 **rsyslog** 에 대한 자세한 내용은 제어 노드의 **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** 파일 및 **rsyslog.conf(5)** 및 **syslog Cryostat** 매뉴얼 페이지를 참조하십시오.

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. **Playbook**을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

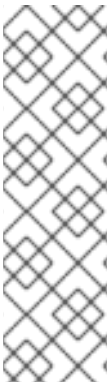
추가 리소스

- **RHEL** 시스템 역할을 사용하여 **CA**에서 인증서 요청 및 자체 서명된 인증서 생성

6.3.2. TLS를 사용하여 서버 로깅 구성

로깅 **RHEL** 시스템 역할을 사용하여 **RHEL** 서버에서 로깅을 구성하고 **TLS** 암호화를 사용하여 원격 로깅 시스템에서 로그를 수신하도록 설정할 수 있습니다.

이 절차에서는 개인 키와 인증서를 생성합니다. 다음으로 **Ansible** 인벤토리의 서버 그룹에 있는 모든 호스트에 **TLS**를 구성합니다.



참고

인증서를 생성하기 위해 플레이북에서 인증서 RHEL 시스템 역할을 호출할 필요가 없습니다. 로깅 RHEL 시스템 역할은 자동으로 호출합니다.

CA에서 생성된 인증서에 서명하려면 관리형 노드를 IdM 도메인에 등록해야 합니다.

사전 요구 사항

- 컨트롤 노드 및 관리형 노드를 준비했습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리 노드에 연결하는 데 사용하는 계정에는 sudo 권한이 있습니다.
- 관리형 노드는 IdM 도메인에 등록됩니다.
- 관리 노드에서 로깅 서버를 구성하려는 로깅 서버가 RHEL 9.2 이상을 실행하고 FIPS 모드가 활성화된 경우 클라이언트는 확장 마스터 시크릿(Extended Master Secret) 확장을 지원하거나 TLS 1.3을 사용해야 합니다. TLS 1.2 연결이 없는 경우 실패합니다. 자세한 내용은 Red Hat Knowledgebase 솔루션 TLS 확장 "확장 마스터 시크릿" 적용 을 참조하십시오.

프로세스

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: ~/playbook.yml)을 생성합니다.

```
---
- name: Configure remote logging solution by using TLS for secure transfer of logs
  hosts: managed-node-01.example.com
  tasks:
    - name: Deploying remote input and remote_files output with certs
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.logging
      vars:
        logging_certificates:
          - name: logging_cert
            dns: ['www.example.com']
            ca: ipa
```

```
      principal: "logging/{{ inventory_hostname }}@IDM.EXAMPLE.COM"
    logging_pki_files:
     - ca_cert: /local/path/to/ca_cert.pem
       cert: /local/path/to/logging_cert.pem
       private_key: /local/path/to/logging_cert.pem
    logging_inputs:
     - name: input_name
       type: remote
       tcp_ports: [514]
       tls: true
       permitted_clients: ['clients.example.com']
    logging_outputs:
     - name: output_name
       type: remote_files
       remote_log_path:
/var/log/remote/%FROMHOST%/%PROGRAMNAME:::secpath-replace%.log
       async_writing: true
       client_count: 20
       io_buffer_size: 8192
    logging_flows:
     - name: flow_name
       inputs: [input_name]
       outputs: [output_name]
```

예제 플레이북에 지정된 설정은 다음과 같습니다.

logging_certificates

이 매개변수의 값은 인증서 **RHEL** 시스템 역할의 **certificate _requests** 에 전달되며 개인 키와 인증서를 생성하는 데 사용됩니다.

logging_pki_files

이 매개변수를 사용하여 로깅에서 사용하는 경로 및 기타 설정을 구성하여 로깅에서 사용하는 **CA,** 인증서 및 **TLS**에 사용되는 키 파일을 찾습니다**.ca_cert_src, ca_cert _src ,cert,cert_src, private_key ,private_key _src, tls.**

참고

logging_certificates 를 사용하여 관리 노드에서 파일을 생성하는 경우 **logging_certificates** 에서 생성되지 않은 파일을 복사하는 데 사용되는 **ca_cert_src,cert_src** 및 **private_key_src** 를 사용하지 마십시오.

ca_cert

관리 노드의 **CA** 인증서 파일의 경로를 나타냅니다. 기본 경로는 **/etc/pki/tls/certs/ca.pem** 이며 파일 이름은 사용자가 설정합니다.

인증서

관리 노드의 인증서 파일의 경로를 나타냅니다. 기본 경로는 **/etc/pki/tls/certs/server-cert.pem** 이며 파일 이름은 사용자가 설정합니다.

private_key

관리 노드의 개인 키 파일의 경로를 나타냅니다. 기본 경로는 **/etc/pki/tls/private/server-key.pem** 이며 파일 이름은 사용자가 설정합니다.

ca_cert_src

ca_cert 에서 지정한 위치로 대상 호스트에 복사되는 제어 노드의 **CA** 인증서 파일의 경로를 나타냅니다. **logging_certificates** 를 사용하는 경우 이 사용하지 마십시오.

cert_src

인증서에서 지정한 위치로 대상 호스트에 복사되는 제어 노드의 인증서 파일의 경로를 나타냅니다. **logging_certificates** 를 사용하는 경우 이 사용하지 마십시오.

private_key_src

private_key 에서 지정한 위치로 대상 호스트에 복사되는 제어 노드에서 개인 키 파일의 경로를 나타냅니다. **logging_certificates** 를 사용하는 경우 이 사용하지 마십시오.

tls

이 매개변수를 **true** 로 설정하면 네트워크를 통한 로그 전송이 안전합니다. 보안 래퍼가 필요하지 않은 경우 **tls: false** 를 설정할 수 있습니다.

역할 변수 및 **rsyslog** 에 대한 자세한 내용은 제어 노드의 **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** 파일 및 **rsyslog.conf(5)** 및 **syslog Cryostat** 매뉴얼 페이지를 참조하십시오.

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. Playbook을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

추가 리소스

- RHEL 시스템 역할을 사용하여 CA에서 인증서 요청 및 자체 서명된 인증서 생성

6.4. RELP에서 로깅 RHEL 시스템 역할 사용

신뢰할 수 있는 이벤트 로깅 프로토콜(RELP)은 TCP 네트워크를 통한 데이터 및 메시지 로깅을 위한 네트워킹 프로토콜입니다. 이벤트 메시지의 안정적인 전달을 보장하고 메시지 손실을 허용하지 않는 환경에서 사용할 수 있습니다.

RELP 발신자는 명령 형태로 로그 항목을 전송하고 수신자는 해당 항목이 처리되면 이를 승인합니다. 일관성을 보장하기 위해 RELP는 모든 종류의 메시지 복구를 위해 전송된 각 명령에 트랜잭션 번호를 저장합니다.

RELP Client와 RELP Server 간의 원격 로깅 시스템을 고려할 수 있습니다. RELP 클라이언트는 로그를 원격 로깅 시스템으로 전송하고 RELP 서버는 원격 로깅 시스템에서 보낸 모든 로그를 수신합니다. 이러한 사용 사례를 달성하기 위해 로깅 RHEL 시스템 역할을 사용하여 로그 항목을 안정적으로 전송하고 수신하도록 로깅 시스템을 구성할 수 있습니다.

6.4.1. RELP를 사용하여 클라이언트 로깅 구성

로깅 RHEL 시스템 역할을 사용하여 로컬에 저장된 로그 메시지를 RELP를 사용하여 원격 로깅 시스템으로 전송할 수 있습니다.

이 절차에서는 Ansible 인벤토리의 clients 그룹에 있는 모든 호스트에 대해 RELP를 구성합니다. RELP 구성은 TLS(Transport Layer Security)를 사용하여 네트워크를 통한 로그의 보안 전송을 위해 메시지 전송을 암호화합니다.

사전 요구 사항

- 컨트롤 노드 및 관리형 노드를 준비했습니다.

- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리 노드에 연결하는 데 사용하는 계정에는 **sudo** 권한이 있습니다.

프로세스

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: **~/playbook.yml**)을 생성합니다.

```
---
- name: Configure client-side of the remote logging solution by using RELP
  hosts: managed-node-01.example.com
  tasks:
    - name: Deploy basic input and RELP output
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.logging
      vars:
        logging_inputs:
          - name: basic_input
            type: basics
        logging_outputs:
          - name: relp_client
            type: relp
            target: logging.server.com
            port: 20514
            tls: true
            ca_cert: /etc/pki/tls/certs/ca.pem
            cert: /etc/pki/tls/certs/client-cert.pem
            private_key: /etc/pki/tls/private/client-key.pem
            pki_authmode: name
            permitted_servers:
              - '*.server.example.com'
        logging_flows:
          - name: example_flow
            inputs: [basic_input]
            outputs: [relp_client]
```

예제 플레이북에 지정된 설정은 다음과 같습니다.

target

이는 원격 로깅 시스템이 실행 중인 호스트 이름을 지정하는 필수 매개 변수입니다.

port

원격 로깅 시스템이 수신 대기 중인 포트 번호입니다.

tls

네트워크를 통해 안전한 로그 전송을 보장합니다. 보안 래퍼가 필요하지 않은 경우 **tls** 변수를 **false** 로 설정할 수 있습니다. 기본적으로 **tls** 매개변수는 **RELP**로 작업하는 동안 **true** 로 설정되며 키/인증서 및 트립릿 **{ca_cert,cert,private_key}** 및/또는 **{ca_cert_src,cert_src,private_key_src}**가 필요합니다.

- **{ca_cert_src,cert_src,private_key_src} triplet**가 설정되면 기본 위치 **/etc/pki/tls/certs** 및 **/etc/pki/tls/private** 이 제어 노드에서 파일을 전송하는 대상으로 사용됩니다. 이 경우 파일 이름은 트리플릿의 원래 이름과 동일합니다.

- **{ca_cert,cert,private_key} triplet**가 설정되면 로깅 구성 전에 파일이 기본 경로에 있어야 합니다.

- 두 트래블릿이 설정되어 있으면 파일이 로컬 경로에서 제어 노드에서 관리 노드의 특정 경로로 전송됩니다.

ca_cert

CA 인증서의 경로를 나타냅니다. 기본 경로는 **/etc/pki/tls/certs/ca.pem** 이며 파일 이름은 사용자가 설정합니다.

인증서

인증서 경로를 나타냅니다. 기본 경로는 **/etc/pki/tls/certs/server-cert.pem** 이며 파일 이름은 사용자가 설정합니다.

private_key

개인 키의 경로를 나타냅니다. 기본 경로는 **/etc/pki/tls/private/server-key.pem** 이며 파일 이름은 사용자가 설정합니다.

ca_cert_src

관리 노드에 복사되는 로컬 **CA** 인증서 파일 경로를 나타냅니다. **ca_cert** 를 지정하면 위치에 복사됩니다.

cert_src

관리 노드에 복사되는 로컬 인증서 파일 경로를 나타냅니다. **cert** 가 지정되면 해당 인증서가 위치에 복사됩니다.

private_key_src

관리 노드에 복사되는 로컬 키 파일 경로를 나타냅니다. **private_key** 가 지정되면 해당 키가 위치에 복사됩니다.

pki_authmode

인증 모드를 이름 또는 지문으로 허용합니다.

permitted_servers

로깅 클라이언트가 **TLS**를 통해 로그를 연결하고 보낼 수 있는 서버 목록입니다.

입력

로깅 입력 사전 목록입니다.

출력

로깅 출력 사전 목록입니다.

역할 변수 및 **rsyslog** 에 대한 자세한 내용은 제어 노드의 **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** 파일 및 **rsyslog.conf(5)** 및 **syslog Cryostat** 매뉴얼 페이지를 참조하십시오.

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. **Playbook**을 실행합니다.

```
$ ansible-playbook ~/playbook.yml
```

6.4.2. RELP를 사용하여 서버 로깅 구성

로깅 RHEL 시스템 역할을 사용하여 RELP를 사용하여 원격 로깅 시스템에서 로그 메시지를 수신하는 서버를 구성할 수 있습니다.

이 절차에서는 Ansible 인벤토리의 서버 그룹에 있는 모든 호스트에 대해 RELP를 구성합니다. RELP 구성은 TLS를 사용하여 네트워크를 통한 로그의 보안 전송을 위해 메시지 전송을 암호화합니다.

사전 요구 사항

- 컨트롤 노드 및 관리형 노드를 준비했습니다.
- 관리 노드에서 플레이북을 실행할 수 있는 사용자로 제어 노드에 로그인되어 있습니다.
- 관리 노드에 연결하는 데 사용하는 계정에는 sudo 권한이 있습니다.

프로세스

1. 다음 콘텐츠를 사용하여 플레이북 파일(예: ~/playbook.yml)을 생성합니다.

```
---
- name: Configure server-side of the remote logging solution by using RELP
  hosts: managed-node-01.example.com
  tasks:
    - name: Deploying remote input and remote_files output
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.logging
      vars:
        logging_inputs:
          - name: relp_server
            type: relp
            port: 20514
            tls: true
            ca_cert: /etc/pki/tls/certs/ca.pem
            cert: /etc/pki/tls/certs/server-cert.pem
            private_key: /etc/pki/tls/private/server-key.pem
            pki_authmode: name
            permitted_clients:
              - '*client.example.com'
        logging_outputs:
          - name: remote_files_output
            type: remote_files
        logging_flows:
```

```
    - name: example_flow
      inputs: [relp_server]
      outputs: [remote_files_output]
```

예제 플레이북에 지정된 설정은 다음과 같습니다.

port

원격 로깅 시스템이 수신 대기 중인 포트 번호입니다.

tls

네트워크를 통해 안전한 로그 전송을 보장합니다. 보안 래퍼가 필요하지 않은 경우 **tls** 변수를 **false** 로 설정할 수 있습니다. 기본적으로 **tls** 매개변수는 **RELP**로 작업하는 동안 **true** 로 설정되며 키/인증서 및 트립릿 **{ca_cert,cert,private_key}** 및/또는 **{ca_cert_src,cert_src,private_key_src}**가 필요합니다.

- **{ca_cert_src,cert_src,private_key_src} triplet**가 설정되면 기본 위치 **/etc/pki/tls/certs** 및 **/etc/pki/tls/private** 이 제어 노드에서 파일을 전송하는 대상으로 사용됩니다. 이 경우 파일 이름은 트리플릿의 원래 이름과 동일합니다.

- **{ca_cert,cert,private_key} triplet**가 설정되면 로깅 구성 전에 파일이 기본 경로에 있어야 합니다.

- 두 트래블릿이 설정되어 있으면 파일이 로컬 경로에서 제어 노드에서 관리 노드의 특정 경로로 전송됩니다.

ca_cert

CA 인증서의 경로를 나타냅니다. 기본 경로는 **/etc/pki/tls/certs/ca.pem** 이며 파일 이름은 사용자가 설정합니다.

인증서

인증서의 경로를 나타냅니다. 기본 경로는 **/etc/pki/tls/certs/server-cert.pem** 이며 파일 이름은 사용자가 설정합니다.

private_key

개인 키의 경로를 나타냅니다. 기본 경로는 **/etc/pki/tls/private/server-key.pem** 이며 파일 이름은 사용자가 설정합니다.

ca_cert_src

관리 노드에 복사되는 로컬 **CA** 인증서 파일 경로를 나타냅니다. **ca_cert** 를 지정하면 위치에 복사됩니다.

cert_src

관리 노드에 복사되는 로컬 인증서 파일 경로를 나타냅니다. **cert** 가 지정되면 해당 인증서가 위치에 복사됩니다.

private_key_src

관리 노드에 복사되는 로컬 키 파일 경로를 나타냅니다. **private_key** 가 지정되면 해당 키가 위치에 복사됩니다.

pki_authmode

인증 모드를 이름 또는 지문으로 허용합니다.

permitted_clients

로깅 서버에서 **TLS**를 통해 로그를 연결하고 보낼 수 있는 클라이언트 목록입니다.

입력

로깅 입력 사전 목록입니다.

출력

로깅 출력 사전 목록입니다.

역할 변수 및 **rsyslog** 에 대한 자세한 내용은 제어 노드의 **/usr/share/ansible/roles/rhel-system-roles.logging/README.md** 파일 및 **rsyslog.conf(5)** 및 **syslog Cryostat** 매뉴얼 페이지를 참조하십시오.

2. 플레이북 구문을 확인합니다.

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

이 명령은 구문만 검증하고 잘못되었지만 유효한 구성으로부터 보호하지 않습니다.

3. **Playbook을 실행합니다.**

```
$ ansible-playbook ~/playbook.yml
```

7장. 시스템 감사

감사는 시스템에 추가 보안을 제공하지 않습니다. 대신 시스템에서 사용되는 보안 정책 위반을 검색하는 데 사용할 수 있습니다. 이러한 위반은 SELinux와 같은 추가 보안 조치로 인해 추가로 방지할 수 있습니다.

7.1. LINUX 감사

Linux 감사를 사용하면 시스템에 대한 보안 관련 정보를 추적할 수 있습니다. 사전 구성된 규칙에 따라 감사는 시스템에서 발생하는 이벤트에 대해 가능한 한 많은 정보를 기록하기 위해 로그 항목을 생성합니다. 이 정보는 미션 크리티컬한 환경과 보안 정책의 위반 및 수행 조치를 결정하는 데 중요합니다.

예를 들어 audit은 로그 파일에 다음 정보를 기록할 수 있습니다.

- 이벤트 날짜 및 시간, 유형 및 결과
- 주체 및 오브젝트의 민감도 레이블
- 이벤트를 트리거한 사용자의 ID와 이벤트 연결
- 감사 구성에 대한 모든 수정 사항 및 감사 로그 파일에 액세스 시도
- SSH, Kerberos 등과 같은 인증 메커니즘의 모든 사용
- 신뢰할 수 있는 데이터베이스(예: /etc/passwd)에 대한 변경 사항
- 시스템 내 또는 시스템으로 정보를 가져오거나 내보내려는 시도
- 사용자 ID, 주체 및 오브젝트 라벨 및 기타 특성을 기반으로 이벤트를 포함하거나 제외

감사 시스템 사용은 여러 보안 관련 인증을 위한 요구 사항이기도 합니다. 감사는 다음 인증 또는 규정 준수 가이드의 요구 사항을 충족하거나 초과하도록 설계되었습니다.

- CAPP(Controled Access Protection Profile)
- 보안 보호 프로필(LSPP)으로 레이블이 지정
- 규칙 세트 기본 액세스 제어(RSBAC)
- National Industrial Security Program Operating Manual (NISPOM)
- 연방 정보 보안 관리법 (FISMA)
- 결제 카드 산업 - 데이터 보안 표준 (PCI-DSS)
- STIG(Security Technical Implementation Guide)

감사는 또한 National Information Assurance Partnership (NIAP) 및 Best Security Industries (BSI)에 의해 평가되었습니다.

사용 사례

파일 액세스 모니터링

감사는 파일 또는 디렉터리가 액세스, 수정, 실행 또는 파일의 속성이 변경되었는지 여부를 추적할 수 있습니다. 예를 들어 중요한 파일에 대한 액세스를 감지하고 이러한 파일 중 하나가 손상된 경우 감사 추적을 사용할 수 있는 데 유용합니다.

시스템 호출 모니터링

특정 시스템 호출을 사용할 때마다 로그 항목을 생성하도록 감사를 구성할 수 있습니다. 예를 들어 set time ,clock_adjtime 및 기타 시간 관련 시스템 호출을 모니터링하여 시스템 시간 변경 사항을 추적하는 데 사용할 수 있습니다.

사용자가 실행하는 명령 기록

감사는 파일이 실행되었는지 여부를 추적할 수 있으므로 특정 명령의 모든 실행을 기록하도록 규칙을 정의할 수 있습니다. 예를 들어 **/bin** 디렉터리의 모든 실행 파일에 대해 규칙을 정의할 수 있습니다. 그런 다음 결과 로그 항목을 사용자 **ID**로 검색하여 사용자당 실행된 명령의 감사 추적을 생성할 수 있습니다.

시스템 경로 이름 실행 기록

규칙 호출 시 **inode**로 경로를 변환하는 파일 액세스를 감시하는 것 외에도, 이제 규칙 호출 시 존재하지 않거나 규칙이 호출된 후 파일이 교체된 경우에도 감사에서 경로 실행을 확인할 수 있습니다. 이를 통해 프로그램 실행 파일을 업그레이드하거나 설치하기 전에 규칙을 계속 작동할 수 있습니다.

보안 이벤트 기록

pam_faillock 인증 모듈은 실패한 로그인 시도를 기록할 수 있습니다. 실패한 로그인 시도를 기록하도록 감사를 설정하고 로그인을 시도한 사용자에 대한 추가 정보를 제공할 수 있습니다.

이벤트 검색

audit은 로그 항목을 필터링하고 여러 조건에 따라 전체 감사 추적을 제공하는 데 사용할 수 있는 **ausearch** 유틸리티를 제공합니다.

요약 보고서 실행

aureport 유틸리티는 기록 된 이벤트에 대한 일일 보고서를 생성하는 데 사용할 수 있습니다. 시스템 관리자는 이러한 보고서를 분석하고 의심 스러운 활동을 추가로 조사할 수 있습니다.

네트워크 액세스 모니터링

시스템 관리자가 네트워크 액세스를 모니터링할 수 있도록 **nftables**,**iptables**, **ebtables** 유틸리티를 구성하여 감사 이벤트를 트리거할 수 있습니다.

참고

감사는 수집된 정보의 양에 따라 시스템 성능에 영향을 미칠 수 있습니다.

7.2. 감사 시스템 아키텍처

감사 시스템은 사용자 공간 애플리케이션 및 유틸리티와 커널 측 시스템 호출 처리의 두 가지 주요 부분으로 구성됩니다. 커널 구성 요소는 사용자 공간 애플리케이션에서 시스템 호출을 수신하고 사용자,작업,**fstype** 또는 **exit** 중 하나를 통해 필터링합니다.

시스템 호출이 **exclude** 필터를 통과하면 감사 규칙 구성에 따라 앞서 언급한 필터 중 하나를 통해 전송되며, 추가 처리를 위해 감사 데몬으로 전송됩니다.

사용자 공간 감사 데몬은 커널에서 정보를 수집하고 로그 파일에 항목을 생성합니다. 기타 감사 사용자 공간 유틸리티는 감사 데몬, 커널 감사 구성 요소 또는 감사 로그 파일과 상호 작용합니다.

- **auditctl** 감사 제어 유틸리티는 커널 감사 구성 요소와 상호 작용하여 규칙을 관리하고 이벤트 생성 프로세스의 많은 설정 및 매개 변수를 제어합니다.

- 나머지 감사 유틸리티는 감사 로그 파일의 내용을 입력으로 사용하고 사용자 요구 사항에 따라 출력을 생성합니다. 예를 들어 **aureport** 유틸리티는 기록된 모든 이벤트에 대한 보고서를 생성합니다.

RHEL 10에서 감사 디스패치 데몬(**audisp**) 기능이 감사 데몬(**auditd**)에 통합되어 있습니다. 감사 이벤트와 실시간 분석 프로그램의 상호 작용을 위한 플러그인의 구성 파일은 기본적으로 **/etc/audit/plugins.d/** 디렉토리에 있습니다.

7.3. 보안 환경에 대한 감사 설정

기본 **auditd** 구성은 대부분의 환경에 적합해야 합니다. 그러나 환경이 엄격한 보안 정책을 충족해야 하는 경우 **/etc/audit/auditd.conf** 파일에서 감사 데몬 구성에 대해 다음 설정을 변경할 수 있습니다.

log_file

감사 로그 파일(일반적으로 **/var/log/audit/**)이 있는 디렉터리는 별도의 마운트 지점에 있어야 합니다. 이렇게 하면 다른 프로세스에서 이 디렉터리에서 공간을 소비하는 것을 방지하고 감사 데몬의 나머지 공간을 정확하게 감지할 수 있습니다.

max_log_file

감사 로그 파일이 포함된 파티션에서 사용 가능한 공간을 최대한 활용하도록 단일 감사 로그 파일의 최대 크기를 설정해야 합니다. **max_log_file** 매개변수는 최대 파일 크기(**MB**)를 지정합니다. 지정된 값은 숫자여야 합니다.

max_log_file_action

max_log_file 에 설정된 제한에 도달하면 감사 로그 파일을 덮어쓰지 않도록 **keep_logs** 로 설정해야 할 작업을 결정합니다.

space_left

space_left_action 매개변수에 설정된 작업이 트리거되는 디스크에 남아 있는 여유 공간의 양을 지정합니다. 관리자가 디스크 공간을 응답하고 확보할 수 있는 충분한 시간을 제공하는 숫자로 설정해야 합니다. **space_left** 값은 감사 로그 파일이 생성되는 비율에 따라 달라집니다. **space_left** 값이 정수로 지정되면 절대 크기(MB)(MiB)로 해석됩니다. 값이 **1**에서 **99** 사이의 숫자로 지정되고 백분율 기호(예: **5%**)가 지정된 경우 감사 데몬은 **log_file** 을 포함하는 파일 시스템의 크기에 따라 절대 크기(MB)를 계산합니다.

space_left_action

space_left_action 매개변수를 적절한 알림 방법을 사용하여 **email** 또는 **exec** 로 설정하는 것이 좋습니다.

admin_space_left

admin_space_left_action 매개변수에 설정된 작업이 트리거되는 절대 최소 여유 공간을 지정하며 관리자가 수행한 로그 작업에 충분한 공간을 남겨 두는 값으로 설정해야 합니다. 이 매개 변수의 숫자 값은 **space_left** 숫자보다 작아야 합니다. 감사 데몬이 디스크 파티션 크기에 따라 번호를 계산하도록 백분율 기호(예: **1%**)를 추가할 수도 있습니다.

admin_space_left_action

시스템을 단일 사용자 모드로 설정하고 관리자가 일부 디스크 공간을 확보할 수 있도록 하려면 **single**로 설정해야 합니다.

disk_full_action

감사 로그 파일이 있는 파티션에서 사용 가능한 공간이 없는 경우 트리거되는 작업을 **halt** 또는 **single** 로 설정해야 합니다. 이렇게 하면 감사에서 더 이상 이벤트를 기록할 수 없는 경우 시스템이 종료되거나 단일 사용자 모드에서 작동합니다.

disk_error_action

감사 로그 파일을 포함하는 파티션에서 오류가 감지되는 경우 트리거되는 작업을 지정합니다. 하드웨어 오작동 처리와 관련된 로컬 보안 정책에 따라 **syslog,single** 또는 **halt** 로 설정해야 합니다.

flush

incremental_async 로 설정해야 합니다. 이는 하드 드라이브와 하드 동기화를 강제 적용하기 전에 디스크에 전송할 수 있는 레코드 수를 결정하는 **freq** 매개변수와 함께 작동합니다. **freq** 매개 변수는 **100** 으로 설정해야 합니다. 이러한 매개변수를 사용하면 감사 이벤트 데이터가 활동 버스트에 적합한 성능을 유지하면서 디스크의 로그 파일과 동기화됩니다.

나머지 구성 옵션은 로컬 보안 정책에 따라 설정해야 합니다.

7.4. AUDITD 시작 및 제어

auditd 가 구성된 후 서비스를 시작하여 감사 정보를 수집하여 로그 파일에 저장합니다. **root** 사용자로 다음 명령을 사용하여 **auditd** 를 시작합니다.

```
# service auditd start
```

부팅 시 시작되도록 **auditd** 를 구성하려면 다음을 수행합니다.

```
# systemctl enable auditd
```

 # auditctl -e 0 명령을 사용하여 **auditd** 를 일시적으로 비활성화하고 **# auditctl -e 1** 로 다시 활성화할 수 있습니다.

service auditd < *action*> 명령을 사용하여 **auditd** 에서 다른 작업을 수행할 수 있습니다. 여기서 **< *action* >**은 다음 중 하나일 수 있습니다.

중지

auditd 를 중지합니다.

재시작

auditd 를 다시 시작합니다.

reload 또는 **force-reload**

/etc/audit/auditd.conf 파일에서 **auditd** 구성을 다시 로드합니다.

rotate

/var/log/audit/ 디렉터리의 로그 파일을 순환합니다.

resume

예를 들어 감사 로그 파일이 있는 디스크 파티션에 사용 가능한 공간이 충분하지 않은 경우 이전에 일시 중지된 후 감사 이벤트 로깅을 재개합니다.

condrestart 또는 **try-restart**

이미 실행 중인 경우에만 **auditd** 를 다시 시작합니다.

status

auditd 의 실행 상태를 표시합니다.

참고

service 명령은 **auditd** 데몬과 올바르게 상호 작용할 수 있는 유일한 방법입니다. **auid** 값이 올바르게 기록되도록 **service** 명령을 사용해야 합니다. **systemctl** 명령은 **enable** 및 **status** 의 두 가지 작업에만 사용할 수 있습니다.

7.5. 감사 로그 항목

기본적으로 감사 시스템은 로그 항목을 **/var/log/audit/audit.log** 파일에 저장합니다. 로그 순환이 활성화된 경우 순환된 **audit.log** 파일이 동일한 디렉터리에 저장됩니다.

/etc/ssh/sshd_config 파일을 읽거나 수정하려는 모든 시도를 기록하려면 다음 감사 규칙을 추가합니다.

```
# auditctl -w /etc/ssh/sshd_config -p warx -k sshd_config
```

예를 들어 **auditd** 데몬이 실행 중인 경우 다음 명령을 사용하면 감사 로그 파일에 새 이벤트가 생성됩니다.

```
$ cat /etc/ssh/sshd_config
```

audit.log 파일의 이 이벤트는 다음과 같습니다.

```
type=SYSCALL msg=audit(1364481363.243:24287): arch=c000003e syscall=2 success=no exit=-13
a0=7fffd19c5592 a1=0 a2=7fffd19c4b50 a3=a items=1 ppid=2686 pid=3538 auid=1000 uid=1000
gid=1000 euid=1000 suid=1000 fsuid=1000 egid=1000 sgid=1000 fsgid=1000 tty=pts0 ses=1
comm="cat" exe="/bin/cat" subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
key="sshd_config"
type=CWD msg=audit(1364481363.243:24287):  cwd="/home/shadowman"
type=PATH msg=audit(1364481363.243:24287): item=0 name="/etc/ssh/sshd_config" inode=409248
dev=fd:00 mode=0100600 ouid=0 ogid=0 rdev=00:00 obj=system_u:object_r:etc_t:s0
```

```
nametype=NORMAL cap_fp=none cap_fi=none cap_fe=0 cap_fver=0
type=PROCTITLE msg=audit(1364481363.243:24287) :
proctitle=636174002F6574632F7373682F737368645F636F6E666967
```

위의 이벤트는 **4**개의 레코드로 구성되며 동일한 타임스탬프와 일련 번호를 공유합니다. 레코드는 항상 **type=** 키워드로 시작합니다. 각 레코드는 공백 또는 쉼표로 구분된 여러 ***name=값*** 쌍으로 구성됩니다. 위 이벤트에 대한 자세한 분석은 다음과 같습니다.

첫 번째 레코드

type=SYSCALL

type 필드에는 레코드 유형이 포함되어 있습니다. 이 예에서 **SYSCALL** 값은 이 레코드가 커널에 대한 시스템 호출에 의해 트리거되었음을 지정합니다.

msg=audit(1364481363.243:24287):

msg 필드 레코드:

- **audit(*time_stamp*: ID)** 형식의 타임스탬프와 고유한 레코드***ID*** 입니다. 동일한 감사 이벤트의 일부로 생성된 경우 여러 레코드가 동일한 타임스탬프와 **ID**를 공유할 수 있습니다. 타임스탬프는 **1970**년 **1**월 **1**일 **00:00:00 UTC** 이후의 **Unix** 시간 형식을 사용하고 있습니다.
- 커널 또는 사용자 공간 애플리케이션에서 제공하는 다양한 이벤트별 *이름=값* 쌍입니다.

arch=c000003e

arch 필드에는 시스템의 **CPU** 아키텍처에 대한 정보가 포함되어 있습니다. **c000003e** 값은 **16**진수 표기법으로 인코딩됩니다. **ausearch** 명령으로 감사 레코드를 검색할 때 **-i** 또는 **--interpret** 옵션을 사용하여 사람이 읽을 수 있는 **16**진수 값을 자동으로 변환합니다. **c000003e** 값은 **x86_64** 로 해석됩니다.

syscall=2

syscall 필드는 커널에 전송된 시스템 호출 유형을 기록합니다. 값 **2** 는 **/usr/include/asm/unistd_64.h** 파일에서 사람이 읽을 수 있는 값과 일치할 수 있습니다. 이 경우 **2** 는 개방형 시스템 호출입니다. **ausyscall** 유틸리티를 사용하면 시스템 호출 번호를 사람이 읽을 수 있는 동등한 값으로 변환할 수 있습니다. **ausyscall --dump** 명령을 사용하여 숫자와 함께 모든 시스템 호출 목록을 표시합니다. 자세한 내용은 **ausyscall(8)** 매뉴얼 페이지를 참조하십시오.

success=no

success 필드는 특정 이벤트에 기록된 시스템 호출이 성공 또는 실패했는지 여부를 기록합니다.

이 경우 호출이 성공하지 못했습니다.

exit=-13

exit 필드에는 시스템 호출에서 반환된 종료 코드를 지정하는 값이 포함되어 있습니다. 이 값은 다른 시스템 호출에 따라 다릅니다. 다음 명령을 사용하여 사람이 읽을 수 있는 동일한 값으로 값을 해석할 수 있습니다.

```
# ausearch --interpret --exit -13
```

이전 예제에서는 감사 로그에 종료 코드 **-13** 으로 실패한 이벤트가 포함되어 있다고 가정합니다.

a0=7fffd19c5592, a1=0, a2=7fffd19c5592, a3=a

a0 ~ a3 필드는 이 이벤트에서 시스템 호출의 **16**진수 표기법으로 인코딩된 처음 **4**개의 인수를 기록합니다. 이러한 인수는 사용되는 시스템 호출에 따라 달라집니다. **ausearch** 유틸리티로 해석될 수 있습니다.

items=1

items 필드에는 **syscall** 레코드를 따르는 **PATH** 보조 레코드 수가 포함되어 있습니다.

ppid=2686

ppid 필드는 상위 프로세스 **ID(PPID)**를 기록합니다. 이 경우 **2686** 은 **bash** 와 같은 상위 프로세스의 **PPID**입니다.

pid=3538

pid 필드는 **PID(**프로세스 **ID)**를 기록합니다. 이 경우 **3538** 은 **cat** 프로세스의 **PID**입니다.

auid=1000

auid 필드는 **loginuid**인 감사 사용자 **ID**를 기록합니다. 이 **ID**는 로그인 시 사용자에게 할당되며 사용자의 **ID**가 변경되는 경우에도 **su - john** 명령으로 사용자 계정을 전환하여 모든 프로세스에 상속됩니다.

uid=1000

uid 필드는 분석 프로세스를 시작한 사용자의 사용자 **ID**를 기록합니다. 사용자 **ID**는 다음 명령을 사용하여 사용자 이름으로 해석될 수 있습니다. **ausearch -i --uid** ***UID***.

gid=1000

gid 필드는 분석 프로세스를 시작한 사용자의 그룹 ID를 기록합니다.

euid=1000

euid 필드는 분석 프로세스를 시작한 사용자의 유효한 사용자 ID를 기록합니다.

suid=1000

suid 필드는 분석 프로세스를 시작한 사용자의 설정된 사용자 ID를 기록합니다.

fsuid=1000

fsuid 필드는 분석 프로세스를 시작한 사용자의 파일 시스템 사용자 ID를 기록합니다.

egid=1000

egid 필드는 분석 프로세스를 시작한 사용자의 유효 그룹 ID를 기록합니다.

sgid=1000

sgid 필드는 분석 프로세스를 시작한 사용자의 세트 그룹 ID를 기록합니다.

fsgid=1000

fsgid 필드는 분석 프로세스를 시작한 사용자의 파일 시스템 그룹 ID를 기록합니다.

tty=pts0

tty 필드는 분석된 프로세스가 호출된 터미널을 기록합니다.

ses=1

ses 필드는 분석된 프로세스가 호출된 세션의 세션 ID를 기록합니다.

comm="cat"

comm 필드는 분석된 프로세스를 호출하는 데 사용된 명령의 명령줄 이름을 기록합니다. 이 경우 cat 명령을 사용하여 이 감사 이벤트를 트리거했습니다.

exe="/bin/cat"

exe 필드는 분석된 프로세스를 호출하는 데 사용된 실행 파일의 경로를 기록합니다.

subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

subj 필드는 분석된 프로세스가 실행 시 레이블이 지정된 **SELinux** 컨텍스트를 기록합니다.

key="sshd_config"

key 필드는 감사 로그에서 이 이벤트를 생성한 규칙과 연관된 관리자 정의 문자열을 기록합니다.

두 번째 레코드

type=CWD

두 번째 레코드에서 **type** 필드 값은 **CWD -** 현재 작업 디렉터리입니다. 이 유형은 첫 번째 레코드에 지정된 시스템 호출을 호출한 프로세스가 실행된 작업 디렉터리를 기록하는 데 사용됩니다.

이 레코드의 목적은 상대 경로가 연결된 **PATH** 레코드에서 캡처되는 경우 현재 프로세스의 위치를 기록하는 것입니다. 이렇게 하면 절대 경로가 재구성될 수 있습니다.

msg=audit(1364481363.243:24287)

msg 필드에는 첫 번째 레코드의 값과 동일한 타임스탬프 및 **ID** 값이 있습니다. 타임스탬프는 **1970**년 **1**월 **1**일 **00:00:00 UTC** 이후의 **Unix** 시간 형식을 사용하고 있습니다.

cwd="/home/*user_name*"

cwd 필드에는 시스템 호출이 호출된 디렉터리의 경로가 포함되어 있습니다.

세 번째 레코드

type=PATH

세 번째 레코드에서 **type** 필드 값은 **PATH** 입니다. 감사 이벤트에는 인수로 시스템 호출에 전달되는 모든 경로에 대한 **PATH-type** 레코드가 포함되어 있습니다. 이 감사 이벤트에서는 하나의 경로(**/etc/ssh/sshd_config**)만 인수로 사용되었습니다.

msg=audit(1364481363.243:24287):

msg 필드에는 첫 번째 및 두 번째 레코드의 값과 동일한 타임스탬프 및 **ID** 값이 있습니다.

item=0

item 필드는 **SYSCALL** 유형 레코드에서 참조하는 총 항목 수의 현재 레코드가 어떤 항목을 나타냅니다. 이 숫자는 **0** 기반이며 값 **0** 은 첫 번째 항목임을 의미합니다.

name="/etc/ssh/sshd_config"

name 필드는 시스템 호출에 전달된 파일 또는 디렉터리의 경로를 인수로 기록합니다. 이 경우 **/etc/ssh/sshd_config** 파일이었습니다.

inode=409248

inode 필드에는 이 이벤트에 기록된 파일 또는 디렉터리와 관련된 **inode** 번호가 포함되어 있습니다. 다음 명령은 **409248 inode** 번호와 연결된 파일 또는 디렉터리를 표시합니다.

```
# find / -inum 409248 -print
/etc/ssh/sshd_config
```

dev=fd:00

dev 필드는 이 이벤트에 기록된 파일 또는 디렉터리가 포함된 장치의 마이너 및 주요 **ID**를 지정합니다. 이 경우 값은 **/dev/fd/0** 장치를 나타냅니다.

mode=0100600

mode 필드는 **st_mode** 필드의 **stat** 명령에서 반환된 숫자 표기법으로 인코딩된 파일 또는 디렉터리 권한을 기록합니다. 자세한 내용은 **stat(2)** 도움말 페이지를 참조하십시오. 이 경우 **0100600** 은 **-rw--------**로 해석될 수 있습니다. 즉, **root** 사용자만 **/etc/ssh/sshd_config** 파일에 대한 읽기 및 쓰기 권한을 갖습니다.

ouid=0

ouid 필드는 오브젝트 소유자의 사용자 **ID**를 기록합니다.

ogid=0

ogid 필드는 오브젝트 소유자의 그룹 **ID**를 기록합니다.

rdev=00:00

rdev 필드에는 특수 파일에 대해서만 기록된 장치 식별자가 포함되어 있습니다. 이 경우 기록된 파일이 일반 파일이므로 사용되지 않습니다.

obj=system_u:object_r:etc_t:s0

obj 필드는 기록된 파일 또는 디렉터리가 실행 시 레이블이 지정된 **SELinux** 컨텍스트를 기록합니다.

nametype=NORMAL

nametype 필드는 지정된 **syscall**의 컨텍스트에서 각 경로 레코드 작업의 의도를 기록합니다.

cap_fp=none

cap_fp 필드는 파일 또는 디렉터리 오브젝트의 허용된 파일 시스템 기반 기능 설정과 관련된 데이터를 기록합니다.

cap_fi=none

cap_fi 필드는 파일 또는 디렉터리 오브젝트의 상속된 파일 시스템 기반 기능 설정과 관련된 데이터를 기록합니다.

cap_fe=0

cap_fe 필드는 파일 또는 디렉터리 오브젝트의 파일 시스템 기반 기능의 유효 비트 설정을 기록합니다.

cap_fver=0

cap_fver 필드는 파일 또는 디렉터리 오브젝트의 파일 시스템 기반 기능 버전을 기록합니다.

네 번째 레코드

type=PROCTITLE

type 필드에는 레코드 유형이 포함되어 있습니다. 이 예에서 **PROCTITLE** 값은 이 레코드가 커널에 대한 시스템 호출에 의해 트리거되는 이 감사 이벤트를 트리거한 전체 명령줄을 제공하도록 지정합니다.

proctitle=636174002F6574632F7373682F737368645F636F6E666967

proctitle 필드는 분석된 프로세스를 호출하는 데 사용된 명령의 전체 명령줄을 기록합니다. 이 필드는 사용자가 감사 로그 구문 분석기에 영향을 미치지 않도록 **16**진수 표기법으로 인코딩됩니다. 이 텍스트는 이 감사 이벤트를 트리거한 명령에 디코딩됩니다. **ausearch** 명령으로 감사 레코드를 검색할 때 **-i** 또는 **--interpret** 옵션을 사용하여 사람이 읽을 수 있는 **16**진수 값을 자동으로 변환합니다. **636174002F6574632F7373682F737368645F636F6E666967** 값은 **cat /etc/ssh/sshd_config** 로 해석됩니다.

7.6. AUDITCTL로 정의된 감사 규칙의 예

감사 시스템은 로그 파일에서 캡처되는 항목을 정의하는 일련의 규칙에서 작동합니다. 감사 규칙은 **auditctl** 유틸리티를 사용하거나 **/etc/audit/rules.d/** 디렉터리에서 명령줄에서 설정할 수 있습니다.

auditctl 명령을 사용하면 감사 시스템의 기본 기능을 제어하고 기록되는 감사 이벤트를 결정하는 규칙을 정의할 수 있습니다.

파일 시스템 규칙 예

1. **/etc/passwd** 파일의 모든 쓰기 액세스 권한 및 모든 속성 변경을 기록하는 규칙을 정의하려면 다음을 수행합니다.

```
# auditctl -w /etc/passwd -p wa -k passwd_changes
```

2. **/etc/selinux/** 디렉터리의 모든 쓰기 액세스 권한 및 모든 속성 변경을 기록하는 규칙을 정의하려면 다음을 수행합니다.

```
# auditctl -w /etc/selinux/ -p wa -k selinux_changes
```

시스템 호출 규칙 예

1. **adjtimex** 또는 **settimeofday** 시스템 호출이 프로그램에서 사용될 때마다 로그 항목을 생성하는 규칙을 정의하기 위해 시스템은 **64**비트 아키텍처를 사용합니다.

```
# auditctl -a always,exit -F arch=b64 -S adjtimex -S settimeofday -k time_change
```

2. 파일이 삭제되거나 **ID**가 **1000** 이상인 시스템 사용자에 의해 이름이 변경될 때마다 로그 항목을 생성하는 규칙을 정의하려면 다음을 수행합니다.

```
# auditctl -a always,exit -S unlink -S unlinkat -S rename -S renameat -F auid>=1000 -F
auid!=4294967295 -k delete
```

-F auid!=4294967295 옵션은 로그인 **UID**가 설정되지 않은 사용자를 제외하는 데 사용됩니다.

executable-file 규칙

/bin/id 프로그램의 모든 실행을 기록하는 규칙을 정의하려면 다음 명령을 실행합니다.

```
# auditctl -a always,exit -F exe=/bin/id -F arch=b64 -S execve -k execution_bin_id
```

추가 리소스

- 시스템의 **auditctl(8)** 도움말 페이지

7.7. 감사 영구 규칙

재부팅 시 지속되는 감사 규칙을 정의하려면 **/etc/audit/rules.d/audit.rules** 파일에 직접 포함하거나 **/etc/audit/rules.d/** 디렉터리에 있는 규칙을 읽는 **augenrules** 프로그램을 사용해야 합니다.

auditd 서비스가 시작될 때마다 **/etc/audit/audit.rules** 파일이 생성됩니다. **/etc/audit/rules.d/** 의 파일은 동일한 **auditctl** 명령줄 구문을 사용하여 규칙을 지정합니다. 해시 기호(**#**) 다음에 있는 행과 텍스트는 무시됩니다.

또한 **auditctl** 명령을 사용하여 **-R** 옵션을 사용하여 지정된 파일에서 규칙을 읽을 수 있습니다. 예를 들면 다음과 같습니다.

```
# auditctl -R /usr/share/audit/sample-rules/30-stig.rules
```

augenrules 스크립트는 **/etc/audit/rules.d/** 디렉터리에 있는 규칙을 읽고 **audit.rules** 파일로 컴파일합니다. 이 스크립트는 자연 정렬 순서에 따라 **.rules** 로 끝나는 모든 파일을 특정 순서로 처리합니다. 이 디렉터리의 파일은 다음과 같은 의미가 있는 그룹으로 구성됩니다.

10

kernel 및 **auditctl** 구성

20

일반적인 규칙과 일치할 수 있지만 다른 일치를 원하는 규칙

30

주요 규칙

40

선택적 규칙

50

서버별 규칙

70

시스템 로컬 규칙

90

종료 가능**(immutable)**

규칙은 한 번에 모두 사용할 수 없습니다. 이는 고려해야 하는 정책의 일부이며 **/etc/audit/rules.d/** 에 복사된 개별 파일입니다. 예를 들어 **STIG** 구성에서 시스템을 설정하려면 규칙 **10-base-config,30-stig,31-privileged, 99-finalize** 를 복사합니다.

/etc/audit/rules.d/ 디렉터리에 규칙이 있으면 **--load** 지시문을 사용하여 **augenrules** 스크립트를 실행하여 로드합니다.

```
# augenrules --load
/sbin/augenrules: No change
No rules
enabled 1
failure 1
pid 742
rate_limit 0
...
```

추가 리소스

- 시스템의 **audit.rules(8)** 및 **augenrules(8)** 도움말 페이지

7.8. 표준 준수를 위해 사전 구성된 감사 규칙 파일

OSPP, PCI DSS 또는 **STIG**와 같은 특정 인증 표준 준수에 대한 감사를 구성하려면 감사 패키지와 함께 설치된 사전 구성된 규칙 파일 집합을 시작점으로 사용할 수 있습니다. 샘플 규칙은 **/usr/share/audit/sample-rules** 디렉터리에 있습니다.

주의

보안 표준이 동적이고 변경될 수 있으므로 **sample-rules** 디렉터리의 감사 샘플 규칙은 완전히 또는 최신 상태가 아닙니다. 이러한 규칙은 감사 규칙을 구성하고 작성할 수 있는 방법을 증명하기 위해서만 제공됩니다. 최신 보안 표준을 즉시 준수하지는 않습니다. 특정 보안 지침에 따라 시스템이 최신 보안 표준을 준수하도록 하려면 **SCAP 기반 보안 규정 준수 툴** 을 사용합니다.

30-NISPOM.rules

국가 산업 보안 프로그램 운영 설명서의 정보 시스템 보안 장에 지정된 요구 사항을 충족하는 감사 규칙 구성입니다.

30-ospp-v42*.rules

OSPP(Protection Profile for General Purpose Operating Systems) 프로파일 **4.2**에 정의된 요구 사항을 충족하는 감사 규칙 구성

30-pci-dss-v31.rules

PCI DSS(Payment Card Industry Data Security Standard) v3.1에서 설정한 요구 사항을 충족하는 감사 규칙 구성

30-STIG.rules

STIG(Security Technical Implementation Guides)에서 설정한 요구 사항을 충족하는 감사 규칙 구성

이러한 구성 파일을 사용하려면 해당 파일을 **/etc/audit/rules.d/** 디렉터리에 복사하고 **augenrules --load** 명령을 사용합니다. 예를 들면 다음과 같습니다.

```
# cd /usr/share/audit/sample-rules/
# cp 10-base-config.rules 30-stig.rules 31-privileged.rules 99-finalize.rules /etc/audit/rules.d/
# augenrules --load
```

번호 지정 체계를 사용하여 감사 규칙을 주문할 수 있습니다. 자세한 내용은 **/usr/share/audit/sample-rules/README-rules** 파일을 참조하십시오.

추가 리소스

- 시스템의 **audit.rules(7)** 도움말 페이지

7.9. AUGENRULES 비활성화

/etc/audit/audit.rules 파일에 정의된 규칙을 사용하도록 감사를 전환하는 **augenrules** 유틸리티를 비활성화할 수 있습니다.

프로세스

1. **/usr/lib/systemd/system/auditd.service** 파일을 **/etc/systemd/system/** 디렉터리에 복사합니다.

```
# cp -f /usr/lib/systemd/system/auditd.service /etc/systemd/system/
```

2. 선택한 텍스트 편집기에서 **/etc/systemd/system/auditd.service** 파일을 편집합니다. 예를 들면 다음과 같습니다.

```
# vi /etc/systemd/system/auditd.service
```

3. **augenrules** 가 포함된 행을 주석 처리하고 **auditctl -R** 명령이 포함된 행의 주석을 제거합니다.

```
#ExecStartPost=-/sbin/augenrules --load
ExecStartPost=-/sbin/auditctl -R /etc/audit/audit.rules
```

4. **systemd** 데몬을 다시 로드하여 **auditd.service** 파일의 변경 사항을 가져옵니다.

```
# systemctl daemon-reload
```

5. **auditd** 서비스를 다시 시작하십시오.

```
# service auditd restart
```

추가 리소스

- 시스템의 **augenrules(8)** 및 **audit.rules(8)** 도움말 페이지

- **auditd** 서비스 다시 시작은 **/etc/audit/audit.rules (Red Hat Knowledgebase)**에 대한 변경 사항을 덮어씁니다.

7.10. 소프트웨어 업데이트를 모니터링하도록 감사 설정

사전 구성된 규칙 **44-installers.rules** 를 사용하여 소프트웨어를 설치하는 다음 유틸리티를 모니터링하도록 감사를 구성할 수 있습니다.

- **dnf** [1]
- **yum**
- **pip**
- **npm**
- **cpan**
- **gem**
- **luarocks**

rpm 유틸리티를 모니터링하려면 **rpm-plugin-audit** 패키지를 설치합니다. 그러면 패키지를 설치하거나 업데이트할 때 감사에서 **SOFTWARE_UPDATE** 이벤트가 생성됩니다. 명령줄에서 **ausearch -m SOFTWARE_UPDATE** 를 입력하여 이러한 이벤트를 나열할 수 있습니다.

참고

사전 구성된 규칙 파일은 **ppc64le** 및 **aarch64** 아키텍처가 있는 시스템에서 사용할 수 없습니다.

사전 요구 사항

- **auditd** 는 7.3절. “보안 환경에 대한 감사 설정” 에 따라 구성됩니다.

프로세스

1. **/usr/share/audit/sample-rules/** 디렉터리에서 **/etc/audit/rules.d/** 디렉터리에 사전 구성된 규칙 파일 **44-installers.rules** 를 복사합니다.

```
# cp /usr/share/audit/sample-rules/44-installers.rules /etc/audit/rules.d/
```

2. 감사 규칙을 로드합니다.

```
# augenrules --load
```

검증

1. 로드된 규칙을 나열합니다.

```
# auditctl -l
-p x-w /usr/bin/dnf-3 -k software-installer
-p x-w /usr/bin/yum -k software-installer
-p x-w /usr/bin/pip -k software-installer
-p x-w /usr/bin/npm -k software-installer
-p x-w /usr/bin/cpan -k software-installer
-p x-w /usr/bin/gem -k software-installer
-p x-w /usr/bin/luarocks -k software-installer
```

2. 설치를 수행합니다. 예를 들면 다음과 같습니다.

```
# dnf reinstall -y vim-enhanced
```

3. 최근 설치 이벤트가 있는지 감사 로그를 검색합니다. 예를 들면 다음과 같습니다.

```
# ausearch -ts recent -k software-installer
&amp;#8211;&amp;#8211;&amp;#8211;&amp;#8211;
time->Thu Dec 16 10:33:46 2021
type=PROCTITLE msg=audit(1639668826.074:298):
proctitle=2F7573722F6C6962657865632F706C6174666F726D2D707974686F6E002F75737
22F62696E2F646E66007265696E7374616C6C002D790076696D2D656E68616E636564
type=PATH msg=audit(1639668826.074:298): item=2 name="/lib64/ld-linux-x86-64.so.2"
inode=10092 dev=fd:01 mode=0100755 ouid=0 ogid=0 rdev=00:00
obj=system_u:object_r:ld_so_t:s0 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0
cap_fver=0 cap_frootid=0
type=PATH msg=audit(1639668826.074:298): item=1 name="/usr/libexec/platform-python"
inode=4618433 dev=fd:01 mode=0100755 ouid=0 ogid=0 rdev=00:00
obj=system_u:object_r:bin_t:s0 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0
cap_fver=0 cap_frootid=0
type=PATH msg=audit(1639668826.074:298): item=0 name="/usr/bin/dnf" inode=6886099
dev=fd:01 mode=0100755 ouid=0 ogid=0 rdev=00:00 obj=system_u:object_r:rpm_exec_t:s0
nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=0
type=CWD msg=audit(1639668826.074:298): cwd="/root"
type=EXECVE msg=audit(1639668826.074:298): argc=5 a0="/usr/libexec/platform-python"
a1="/usr/bin/dnf" a2="reinstall" a3="-y" a4="vim-enhanced"
type=SYSCALL msg=audit(1639668826.074:298): arch=c000003e syscall=59 success=yes
exit=0 a0=55c437f22b20 a1=55c437f2c9d0 a2=55c437f2aeb0 a3=8 items=3 ppid=5256
pid=5375 auid=0 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=3
comm="dnf" exe="/usr/libexec/platform-python3.6"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="software-installer"
```

7.11. 감사를 사용하여 사용자 로그인 시간 모니터링

특정 시간에 로그인한 사용자를 모니터링하려면 동일한 정보를 제공하는 다양한 방법을 제공하는 **ausearch** 또는 **aureport** 도구를 사용할 수 있습니다. 특정 방식으로 감사를 구성할 필요가 없습니다.

사전 요구 사항

- **auditd** 는 7.3절. “보안 환경에 대한 감사 설정” 에 따라 구성됩니다.

프로세스

사용자 로그인 시간을 표시하려면 다음 명령 중 하나를 사용합니다.

- **USER_LOGIN** 메시지 유형의 감사 로그를 검색합니다.

```
# ausearch -m USER_LOGIN -ts '12/02/2020' '18:00:00' -sv no
time->Mon Nov 22 07:33:22 2021
type=USER_LOGIN msg=audit(1637584402.416:92): pid=1939 uid=0 auid=4294967295
ses=4294967295 subj=system_u:system_r:sshd_t:s0-s0:c0.c1023 msg='op=login acct="
(unknown)" exe="/usr/sbin/sshd" hostname=? addr=10.37.128.108 terminal=ssh res=failed'
```

-

  - **-ts** 옵션을 사용하여 날짜와 시간을 지정할 수 있습니다. 이 옵션을 사용하지 않는 경우 **ausearch** 는 오늘 부터 결과를 제공하고 시간을 생략하면 **ausearch** 는 자정의 결과를 제공합니다.

  - **-sv yes** 옵션을 사용하여 성공적인 로그인 시도를 필터링하고 로그인 시도에 실패한 경우 **-sv no** 를 사용할 수 있습니다.

- **ausearch** 명령의 원시 출력을 **aulast** 유틸리티로 파이프하여 마지막 명령의 출력과 유사한 형식으로 출력을 표시합니다. 예를 들면 다음과 같습니다.

```
# ausearch --raw | aulast --stdin
root     ssh          10.37.128.108    Mon Nov 22 07:33 - 07:33  (00:00)
root     ssh          10.37.128.108    Mon Nov 22 07:33 - 07:33  (00:00)
root     ssh          10.22.16.106     Mon Nov 22 07:40 - 07:40  (00:00)
reboot   system boot  4.18.0-348.6.el8 Mon Nov 22 07:33
```

- **aureport** 명령을 **--login -i** 옵션과 함께 사용하여 로그인 이벤트 목록을 표시합니다.

```
# aureport --login -i

Login Report
============================================
# date time auid host term exe success event
============================================
1. 11/16/2021 13:11:30 root 10.40.192.190 ssh /usr/sbin/sshd yes 6920
2. 11/16/2021 13:11:31 root 10.40.192.190 ssh /usr/sbin/sshd yes 6925
3. 11/16/2021 13:11:31 root 10.40.192.190 ssh /usr/sbin/sshd yes 6930
4. 11/16/2021 13:11:31 root 10.40.192.190 ssh /usr/sbin/sshd yes 6935
5. 11/16/2021 13:11:33 root 10.40.192.190 ssh /usr/sbin/sshd yes 6940
6. 11/16/2021 13:11:33 root 10.40.192.190 /dev/pts/0 /usr/sbin/sshd yes 6945
```

추가 리소스

- **ausearch(8), aulast(8)** 및 **aureport(8)** 도움말 페이지

7.12. 추가 리소스

- RHEL 감사 시스템 참조 **(Red Hat Knowledgebase)**

- 컨테이너의 auditd 실행 옵션 (Red Hat Knowledgebase)
- Linux 감사 문서 프로젝트 페이지 (Github.com)
- 시스템의 /usr/share/doc/audit/ 디렉터리
- auditd(8), auditctl(8), ausearch(8), audit.rules(7), audispd.conf(5), audispd(8), auditd.conf(5), ausearch-expression(5), aulast(8), aulastlog(8), aureport(8), ausyscall(8), autrace(8), auvirt(8) 도움말 페이지

---

[1] dnf 는 RHEL의 심볼릭 링크이므로 dnf 감사 규칙의 경로에 symlink의 대상이 포함되어야 합니다. 올바른 감사 이벤트를 수신하려면 path=/usr/bin/dnf 경로를 /usr/bin/dnf-3 로 변경하여 44-installers.rules 파일을 수정합니다.

8장. 보안 업데이트 관리 및 모니터링

보안 업데이트를 설치하고 업데이트에 대한 추가 세부 정보를 표시하여 새로 발견된 위협 및 취약성으로부터 **Red Hat Enterprise Linux** 시스템을 안전하게 보호하는 방법을 알아보십시오.

8.1. 보안 업데이트 식별

현재 및 향후 위협으로부터 엔터프라이즈 시스템을 안전하게 보호하려면 정기적인 보안 업데이트가 필요합니다. **Red Hat** 제품 보안팀은 엔터프라이즈 솔루션을 안정적으로 배포하고 유지 관리하는 데 필요한 지침을 제공합니다.

8.1.1. 보안 권고란 무엇입니까?

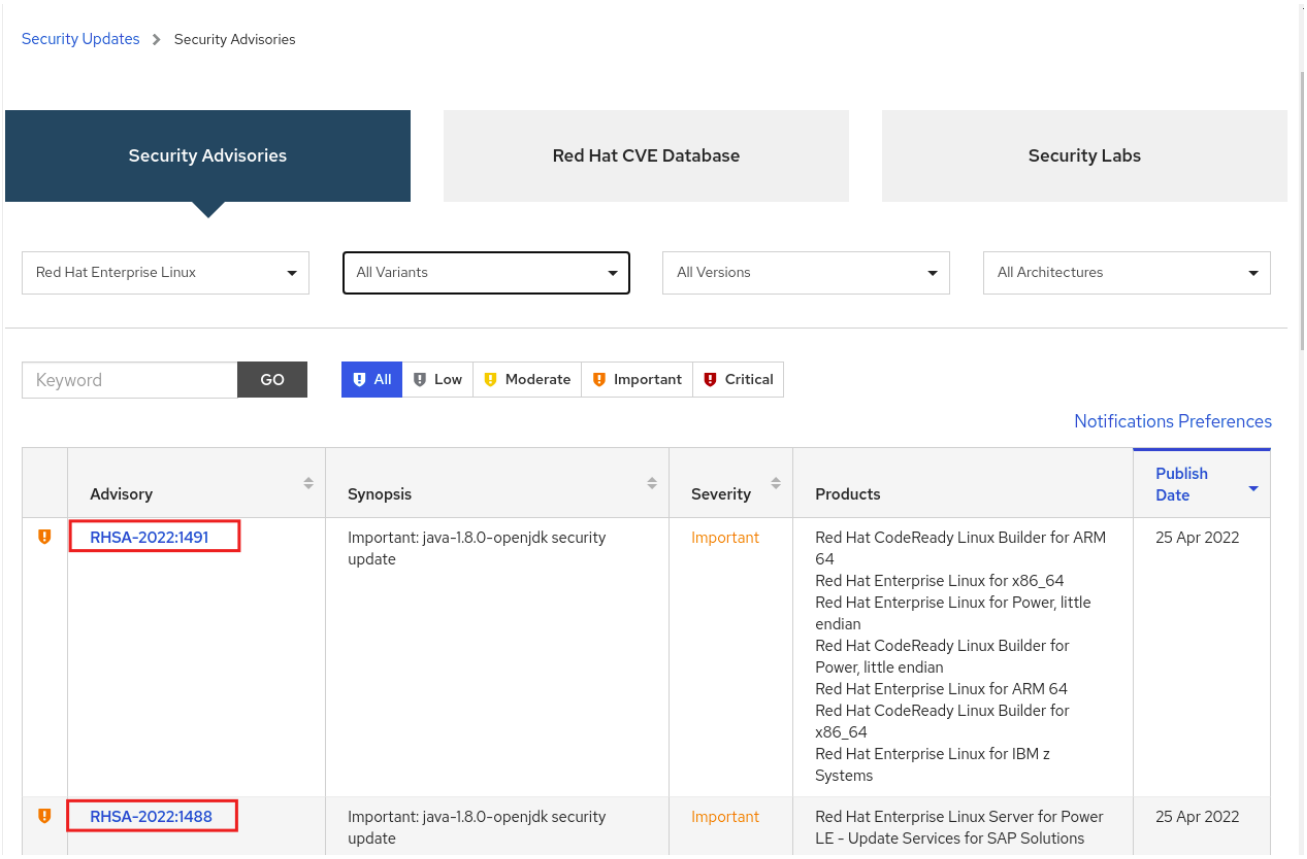
RHSA(Red Hat Security Advisories)에서는 **Red Hat** 제품 및 서비스에서 수정되는 보안 결함에 대한 정보를 설명합니다.

각 **RHSA**에는 다음 정보가 포함됩니다.

- 심각도
- 유형 및 상태
- 영향을 받는 제품
- 수정된 문제 요약
- 문제에 대한 티켓으로 연결됩니다. 모든 티켓이 공개되는 것은 아닙니다.
- **CVE(Common Vulnerabilities and Exposures)** 번호 및 공격 복잡성과 같은 추가 세부 정보가 포함된 링크입니다.

Red Hat 고객 포털은 **Red Hat**에서 게시한 **Red Hat** 보안 권고 목록을 제공합니다. **Red Hat Security Advisories** 목록에서 권고 **ID**로 이동하여 특정 권고의 세부 정보를 표시할 수 있습니다.

그림 **8.1.** 보안 권고 목록



선택적으로 특정 제품, 변형, 버전 및 아키텍처로 결과를 필터링할 수도 있습니다. 예를 들어 **Red Hat Enterprise Linux 9**에 대한 권고만 표시하려면 다음 필터를 설정할 수 있습니다.

- 제품: **Red Hat Enterprise Linux**
- 변형: 모든 변형
- 버전: **9**
- 선택적으로 마이너 버전을 선택합니다.

추가 리소스

- Red Hat Security Advisories 목록
- Anatomy of a Red Hat Security Advisory

8.1.2. 호스트에 설치되지 않은 보안 업데이트 표시

dnf 유틸리티를 사용하여 시스템에 사용 가능한 모든 보안 업데이트를 나열할 수 있습니다.

사전 요구 사항

- Red Hat 서브스크립션이 호스트에 연결되어 있습니다.

프로세스

- 호스트에 설치되지 않은 사용 가능한 모든 보안 업데이트를 나열합니다.

```
# dnf updateinfo list updates security
…
RHSA-2019:0997 Important/Sec. platform-python-3.6.8-2.el8_0.x86_64
RHSA-2019:0997 Important/Sec. python3-libs-3.6.8-2.el8_0.x86_64
RHSA-2019:0990 Moderate/Sec.  systemd-239-13.el8_0.3.x86_64
…
```

8.1.3. 호스트에 설치된 보안 업데이트 표시

dnf 유틸리티를 사용하여 시스템에 설치된 보안 업데이트를 나열할 수 있습니다.

프로세스

- 호스트에 설치된 모든 보안 업데이트를 나열합니다.

```
# dnf updateinfo list security --installed
…
RHSA-2019:1234 Important/Sec. libssh2-1.8.0-7.module+el8+2833+c7d6d092
RHSA-2019:4567 Important/Sec. python3-libs-3.6.7.1.el8.x86_64
RHSA-2019:8901 Important/Sec. python3-libs-3.6.8-1.el8.x86_64
…
```

단일 패키지의 여러 업데이트가 설치된 경우 **dnf** 는 패키지에 대한 모든 권고를 나열합니다. 이전 예에서는 시스템 설치 이후 **python3-libs** 패키지에 대한 두 가지 보안 업데이트가 설치되었습니다.

8.1.4. DNF를 사용하여 특정 권고 표시

dnf 유틸리티를 사용하여 업데이트에 사용할 수 있는 특정 권고 정보를 표시할 수 있습니다.

사전 요구 사항

- **Red Hat** 서브스크립션이 호스트에 연결되어 있습니다.
- 보안 권고의 **ID**를 알고 있습니다.
- 권고에서 제공하는 업데이트는 설치되지 않습니다.

프로세스

- 특정 권고를 표시합니다. 예를 들면 다음과 같습니다.

```
# dnf updateinfo info RHSA-2019:0997
==============================================================================
  Important: python3 security update
==============================================================================
  Update ID: RHSA-2019:0997
      Type: security
   Updated: 2019-05-07 05:41:52
      Bugs: 1688543 - CVE-2019-9636 python: Information Disclosure due to urlsplit improper
NFKC normalization
      CVEs: CVE-2019-9636
Description: …
```

8.2. 보안 업데이트 설치

Red Hat Enterprise Linux에서는 특정 보안 권고와 사용 가능한 모든 보안 업데이트를 설치할 수 있습니다. 보안 업데이트를 자동으로 다운로드하고 설치하도록 시스템을 구성할 수도 있습니다.

8.2.1. 사용 가능한 모든 보안 업데이트 설치

시스템의 보안을 최신 상태로 유지하려면 **dnf** 유틸리티를 사용하여 현재 사용 가능한 모든 보안 업데이트를 설치할 수 있습니다.

사전 요구 사항

- **Red Hat** 서브스크립션이 호스트에 연결되어 있습니다.

프로세스

1. **dnf** 유틸리티를 사용하여 보안 업데이트를 설치합니다.

```
# dnf update --security
```

--security 매개변수가 없으면 **dnf update** 는 버그 수정 및 개선 사항을 포함하여 모든 업데이트를 설치합니다.

2. **y** 를 눌러 설치를 확인하고 시작합니다.

```
…
Transaction Summary
==========================================
Upgrade  … Packages

Total download size: … M
Is this ok [y/d/N]: y
```

3. 선택 사항: 업데이트된 패키지를 설치한 후 시스템을 수동으로 다시 시작해야 하는 프로세스를 나열합니다.

```
# dnf needs-restarting
1107 : /usr/sbin/rsyslogd -n
1199 : -bash
```

이전 명령은 서비스가 아닌 재시작이 필요한 프로세스만 나열합니다. 즉, **systemctl** 유틸리티를 사용하여 나열된 프로세스를 다시 시작할 수 없습니다. 예를 들어 이 프로세스를 소유한 사용자가 로그아웃하면 출력의 **bash** 프로세스가 종료됩니다.

8.2.2. 특정 권고에서 제공하는 보안 업데이트 설치

특정 상황에서는 특정 업데이트만 설치해야 할 수 있습니다. 예를 들어 다운타임을 예약하지 않고 특정 서비스를 업데이트할 수 있는 경우 이 서비스에 대한 보안 업데이트를 설치하고 나중에 나머지 보안 업데이트를 설치할 수 있습니다.

사전 요구 사항

- **Red Hat** 서브스크립션이 호스트에 연결되어 있습니다.
- 업데이트하려는 보안 권고의 **ID**를 알고 있습니다.

  자세한 내용은 보안 권고 업데이트 식별 섹션을 참조하십시오.

프로세스

1. 특정 권고를 설치합니다. 예를 들면 다음과 같습니다.

```
# dnf update --advisory=RHSA-2019:0997
```

2. 또는 **dnf upgrade-minimal** 명령을 사용하여 최소 버전 변경으로 특정 권고를 적용하도록 업데이트합니다. 예를 들면 다음과 같습니다.

```
# dnf upgrade-minimal --advisory=RHSA-2019:0997
```

3. **y** 를 눌러 설치를 확인하고 시작합니다.

```
…
Transaction Summary
==========================================
Upgrade  … Packages

Total download size: … M
Is this ok [y/d/N]: y
```

4. 선택 사항: 업데이트된 패키지를 설치한 후 시스템을 수동으로 다시 시작해야 하는 프로세스를 나열합니다.

```
# dnf needs-restarting
1107 : /usr/sbin/rsyslogd -n
1199 : -bash
```

이전 명령은 서비스가 아닌 재시작이 필요한 프로세스만 나열합니다. 즉, **systemctl** 유틸리티를 사용하여 나열된 모든 프로세스를 다시 시작할 수 없습니다. 예를 들어 이 프로세스를 소유한 사용자가 로그아웃하면 출력의 **bash** 프로세스가 종료됩니다.

8.2.3. 자동으로 보안 업데이트 설치

모든 보안 업데이트를 자동으로 다운로드하고 설치하도록 시스템을 구성할 수 있습니다.

사전 요구 사항

- **Red Hat** 서브스크립션이 호스트에 연결되어 있습니다.
- **dnf-automatic** 패키지가 설치됩니다.

프로세스

1. **[commands]** 섹션의 **/etc/dnf/automatic.conf** 파일에서 **upgrade_type** 옵션이 기본값 또는 **security** 로 설정되어 있는지 확인합니다.

```
[commands]
#  What kind of upgrade to perform:
# default                            = all available upgrades
# security                           = only the security upgrades
upgrade_type = security
```

2. **systemd** 타이머 장치를 활성화하고 시작합니다.

```
# systemctl enable --now dnf-automatic-install.timer
```

검증

1. 타이머가 활성화되어 있는지 확인합니다.

```
# systemctl status dnf-automatic-install.timer
```

추가 리소스

- 시스템의 **dnf-automatic(8)** 도움말 페이지