# Oracle® Solaris 11.2의 파일 보안 및 파일 무결성 확인

부품 번호: E53950
2014년 7월

ORACLE®

# 목차

# 이 설명서 사용

- **개요** - 적합한 파일을 보호하고 숨겨진 파일 권한을 보며 허위 파일을 찾아 그 실행을 방지하는 방법을 설명합니다. 또한 Oracle Solaris 시스템에서 시간에 따라 파일의 무결성을 확인하는 방법을 설명합니다.
- **대상** - 시스템 관리자
- **필요한 지식** - 사이트 보안 요구 사항

## 제품 설명서 라이브러리

이 제품에 대한 최신 정보 및 알려진 문제는 설명서 라이브러리(http://www.oracle.com/pls/topic/lookup?ctx=E56343)에서 확인할 수 있습니다.

## Oracle 지원 액세스

Oracle 고객은 My Oracle Support를 통해 온라인 지원에 액세스할 수 있습니다. 자세한 내용은 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info를 참조하거나, 청각 장애가 있는 경우 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs를 방문하십시오.

## 피드백

http://www.oracle.com/goto/docfeedback에서 이 설명서에 대한 피드백을 보낼 수 있습니다.

♦ ♦ ♦ 1 장

# 1 파일에 대한 액세스 제어

이 장에서는 Oracle Solaris에서 파일을 보호하는 방법에 대해 설명합니다. 또한 시스템을 손상시킬 수 있는 사용 권한을 가진 파일로부터 시스템을 보호하는 방법에 대해 설명합니다.

**참고 -** ZFS는 기본 OS 파일 시스템입니다. ACL(액세스 제어 목록)을 사용하여 ZFS 파일을 보호하려면 "Oracle Solaris 11.2의 ZFS 파일 시스템 관리 "의 7 장, "ACL 및 속성을 사용하여 Oracle Solaris ZFS 파일 보호"를 참조하십시오.

이 장에서는 다음 내용을 다룹니다.



## UNIX 사용 권한으로 파일 보호

UNIX 파일 권한 및 ACL을 통해 파일을 보호할 수 있습니다. 고정된 비트가 설정된 파일 및 실행 파일에는 특수한 보안 조치가 필요합니다.

### 파일 확인 및 보안 명령

이 표에서는 파일과 디렉토리를 모니터 및 보안하는 명령에 대해 설명합니다.

**표 1-1** 파일 및 디렉토리 보안 명령

| 명령 | 설명 | 매뉴얼 페이지 |
|---|---|---|
| `ls` | 디렉토리의 파일 및 파일 정보를 나열합니다. | `ls(1)` |
| `chown` | 파일 소유권을 변경합니다. | `chown(1)` |
| `chgrp` | 파일의 그룹 소유권을 변경합니다. | `chgrp(1)` |

| 명령 | 설명 | 매뉴얼 페이지 |
|---|---|---|
| chmod | 파일 사용 권한을 변경합니다. 문자 및 기호를 사용하는 심볼릭 모드 또는 8진수를 사용하는 절대 모드를 사용하여 파일 사용 권한을 변경할 수 있습니다. | chmod(1) |

# 파일 및 디렉토리 소유권

기존 UNIX 파일 사용 권한은 다음과 같은 세 가지 사용자 클래스에 소유권을 지정할 수 있습니다.

- **사용자** - 파일 또는 디렉토리 소유자(일반적으로 파일을 만든 사용자)입니다. 파일 소유자는 파일 읽기 권한, 파일 쓰기(파일 변경) 권한 또는 파일 실행 권한(파일이 명령인 경우)을 가지는 사용자를 결정할 수 있습니다.
- **그룹** - 사용자 그룹의 구성원입니다.
- **기타** - 파일 소유자가 아니며 그룹 구성원이 아닌 기타 모든 사용자입니다.

일반적으로 파일 소유자는 파일 사용 권한을 지정하거나 수정할 수 있습니다. 또한 root 계정은 파일 소유권을 변경할 수 있습니다. 시스템 정책을 대체하려면 예 1-2. “사용자가 고유 파일의 소유권을 변경할 수 있도록 설정”를 참조하십시오.

파일 유형은 일곱 가지 중 하나일 수 있습니다. 각 유형은 다음과 같은 기호로 표시됩니다.

| | |
|---|---|
| -(마이너스 기호) | 텍스트 또는 프로그램 |
| b | 블록 특정 파일 |
| c | 문자 특정 파일 |
| d | 디렉토리 |
| l | 심볼릭 링크 |
| s | 소켓 |
| D | 도어 |
| P | 명명된 파이프(FIFO) |

# UNIX 파일 사용 권한

다음 표에서는 파일 또는 디렉토리 사용자의 각 클래스에 부여할 수 있는 사용 권한을 나열하고 설명합니다.

**표 1-2** 파일 및 디렉토리 사용 권한

| 기호 | 사용 권한 | 객체 | 설명 |
|---|---|---|---|
| r | 읽기 | 파일 | 지정된 사용자가 파일을 열고 파일 내용을 읽을 수 있습니다. |
| | | 디렉토리 | 지정된 사용자가 디렉토리의 파일을 나열할 수 있습니다. |
| w | 쓰기 | 파일 | 지정된 사용자가 파일 내용을 수정하거나 파일을 삭제할 수 있습니다. |
| | | 디렉토리 | 지정된 사용자가 파일을 추가하거나 디렉토리에 링크를 추가할 수 있습니다. 파일을 제거하거나 디렉토리의 링크를 제거할 수도 있습니다. |
| x | 실행 | 파일 | 지정된 사용자가 파일을 실행할 수 있습니다(파일이 프로그램 또는 셸 스크립트인 경우). exec(2) 시스템 호출 중 하나로 프로그램을 실행할 수도 있습니다. |
| | | 디렉토리 | 지정된 사용자가 파일을 열거나 디렉토리의 파일을 실행할 수 있습니다. 디렉토리 및 하위 디렉토리를 만들 수도 있습니다. |
| - | 거부됨 | 파일 및 디렉토리 | 지정된 사용자가 파일을 읽거나 쓰거나 실행할 수 없습니다. |

해당 파일 사용 권한은 일반 파일을 비롯하여 장치, 소켓, 명명된 파이프(FIFO) 등의 특수 파일에 적용됩니다.

심볼릭 링크의 경우 링크가 가리키는 파일의 사용 권한이 적용됩니다.

해당 디렉토리에 대해 제한적인 파일 사용 권한을 설정하여 디렉토리 및 하위 디렉토리의 파일을 보호할 수 있습니다. 단, root 역할은 시스템의 모든 파일 및 디렉토리에 대한 액세스 권한을 가집니다.

## setuid, setgid 및 고정된 비트를 사용하는 특수 파일 권한

실행 파일 및 공용 디렉토리에 대해 세 가지 특수 유형의 사용 권한(setuid, setgid 및 고정된 비트)을 사용할 수 있습니다. 해당 사용 권한이 설정되면 실행 파일을 실행하는 모든 사용자가 실행 파일 소유자(또는 그룹)의 ID를 사용합니다.

특수 사용 권한에는 보안 위험이 따르므로 특수 사용 권한을 설정할 때는 각별히 신중해야 합니다. 예를 들어, 특정 사용자가 사용자 ID(UID)를 0(root의 UID)으로 설정하는 프로그램을 실행하여 root 기능을 얻을 수 있습니다. 또한 모든 사용자가 자신이 소유한 파일에 대해 특수 사용 권한을 설정할 수 있으므로 다른 보안 위험에 노출됩니다.

root 권한을 얻기 위해 setuid 사용 권한 및 setgid 사용 권한이 무단으로 사용되고 있지 않은지 시스템을 모니터해야 합니다. root 또는 bin 이외의 다른 사용자에게 관리 프로그램의 소유권을 부여하는 사용 권한은 의심스러운 것입니다. 이 특수 사용 권한을 사용하는 모든 파일을 검색하여 나열하려면 특수 파일 사용 권한이 있는 파일을 찾는 방법 [21]을 참조하십시오.

## setuid 사용 권한

실행 파일에 대해 setuid 사용 권한이 설정되면 파일 소유자를 기반으로 이 파일을 실행하는 프로세스에 액세스 권한이 부여됩니다. 액세스 권한은 실행 파일을 실행 중인 사용자를 기반으로 하는 것이 아닙니다. 이 특수 사용 권한에 따라 사용자는 일반적으로 소유자에게만 제공되는 파일 및 디렉토리 액세스 권한을 얻을 수 있습니다.

예를 들어, passwd 명령에 대한 setuid 사용 권한은 사용자가 암호를 변경할 수 있도록 합니다. setuid 사용 권한이 있는 passwd 명령은 다음과 유사합니다.

```
-r-sr-sr-x   1 root     sys        56808 Jun 17 12:02 /usr/bin/passwd
```

이 특수 사용 권한에는 보안 위험이 따릅니다. 정해진 일부 사용자는 프로세스 실행이 끝난 후에도 setuid 프로세스로 부여받은 사용 권한을 유지할 수 있습니다.

**참고** - 프로그램의 예약된 UID(0-100)로 setuid 사용 권한을 사용하면 유효 UID가 제대로 설정되지 않을 수 있습니다. 셸 스크립트를 사용하십시오. 또는 setuid 사용 권한에 예약된 UID를 사용하지 마십시오.

## setgid 사용 권한

setgid 사용 권한은 setuid 사용 권한과 유사합니다. 프로세스의 유효 그룹 ID(GID)가 파일을 소유한 그룹으로 변경되며 해당 그룹에게 부여된 사용 권한을 기반으로 사용자에게 액세스 권한이 부여됩니다. /usr/bin/mail 명령의 setgid 사용 권한은 다음과 같습니다.

```
-r-x--s--x   1 root    mail      71212 Jun 17 12:01 /usr/bin/mail
```

setgid 권한이 디렉토리에 적용되면 이 디렉토리에서 만든 파일이 이 디렉토리를 소유하는 그룹에 속하게 됩니다. 파일은 만들기 프로세스가 속한 그룹에 속하지 않습니다. 디렉토리에서 쓰기 및 실행 권한을 가지는 사용자는 해당 디렉토리에서 파일을 만들 수 있습니다. 단, 파일은 사용자가 속한 그룹이 아닌 디렉토리를 소유한 그룹에 속합니다.

root 권한을 얻기 위해 setgid 사용 권한이 무단으로 사용되고 있지 않은지 시스템을 모니터해야 합니다. root 또는 bin 이외의 다른 비정상적인 그룹에 프로그램에 대한 그룹 액세스 권한을 부여하는 사용 권한은 의심스러운 것입니다. 이 사용 권한을 사용하는 모든 파일을 검색하여 나열하려면 특수 파일 사용 권한이 있는 파일을 찾는 방법 [21]을 참조하십시오.

## 고정된 비트

고정된 비트는 디렉토리 내의 파일을 보호하는 사용 권한 비트입니다. 디렉토리에 고정된 비트가 설정된 경우 파일 소유자, 디렉토리 소유자 또는 권한 있는 사용자만 파일을 삭제할 수 있습니다. 권한 있는 사용자의 예로 root 사용자를 들 수 있습니다. 고정된 비트는 사용자가 공용 디렉토리(예: /tmp)에서 다른 사용자의 파일을 삭제하지 못하도록 합니다.

```
drwxrwxrwt 7  root  sys   400 Sep  3 13:37 tmp
```

TMPFS 파일 시스템에서 공용 디렉토리를 설정할 때 수동으로 고정된 비트를 설정해야 합니다. 지침은 예 1-5. “절대 모드로 특수 파일 사용 권한 설정”를 참조하십시오.

## 기본 umask 값

파일 또는 디렉토리를 만들 때 일련의 기본 사용 권한이 사용됩니다. 시스템 기본값은 공개됩니다. 텍스트 파일의 666 사용 권한은 모든 사용자에게 읽기 및 쓰기 권한을 부여합니다. 디렉토리 및 실행 파일의 777 사용 권한은 모든 사용자에게 읽기, 쓰기 및 실행 권한을 부여합니다. 일반적으로 사용자는 셸 초기화 파일(예: .bashrc 및 .kshrc.user)의 시스템 기본값을 대체합니다. 관리자는 /etc/profile 파일에서 기본값을 설정할 수도 있습니다.

umask 명령으로 지정된 값은 기본값에서 제외됩니다. 이 프로세스는 chmod 명령이 권한을 부여하는 것과 동일한 방법으로 사용 권한을 거부합니다. 예를 들어, chmod 022 명령은 그룹 및 기타에 쓰기 권한을 부여합니다. umask 022 명령은 그룹 및 기타에 대한 쓰기 권한을 거부합니다.

다음 표에서는 일반적인 umask 값과 해당 값이 실행 파일에 끼치는 영향을 보여 줍니다.

**표 1-3** 다양한 보안 레벨에 대한 umask 설정

| 보안 레벨 | umask 설정 | 허용되지 않는 사용 권한 |
|---|---|---|
| 허가(744) | 022 | 그룹 및 기타에 대한 w |
| 보통(751) | 026 | 그룹에 대한 w, 기타에 대한 rw |
| 엄격(740) | 027 | 그룹에 대한 w, 기타에 대한 rwx |
| 심각(700) | 077 | 그룹 및 기타에 대한 rwx |

umask 값 설정에 대한 자세한 내용은 umask(1) 매뉴얼 페이지를 참조하십시오.

## 파일 사용 권한 모드

chmod 명령을 통해 파일 사용 권한을 변경할 수 있습니다. 사용 권한을 변경하려면 root나 파일 또는 디렉토리의 소유자여야 합니다.

chmod 명령을 사용하여 다음 두 가지 모드 중 하나로 사용 권한을 설정할 수 있습니다.

- **절대 모드** - 숫자를 사용하여 파일 사용 권한을 나타냅니다. 절대 모드를 사용하여 사용 권한을 변경하는 경우 8진수 모드 숫자별로 각 세 문자의 사용 권한을 나타냅니다. 절대 모드는 사용 권한 설정에 사용되는 가장 일반적인 방법입니다.
- **심볼릭 모드** - 문자와 기호의 조합을 통해 사용 권한을 추가하거나 사용 권한을 제거합니다.

다음 표에서는 절대 모드로 파일 사용 권한을 설정하는 데 사용할 8진수 값을 나열합니다. 세 세트의 이러한 숫자를 사용하여 소유자, 그룹 및 기타에 대한 사용 권한을 이 순서대로 설정할 수 있습니다. 예를 들어, 값 644는 소유자에 대해 읽기 및 쓰기 권한을 설정하고 그룹 및 기타에 대해 읽기 전용 권한을 설정합니다.

**표 1-4** 절대 모드로 파일 사용 권한 설정

| 8진수 값 | 파일 사용 권한 세트 | 사용 권한 설명 |
|---|---|---|
| 0 | --- | 권한 없음 |
| 1 | --x | 실행 권한만 |
| 2 | -w- | 쓰기 권한만 |
| 3 | -wx | 쓰기 및 실행 권한 |
| 4 | r-- | 읽기 권한만 |
| 5 | r-x | 읽기 및 실행 권한 |
| 6 | rw- | 읽기 및 쓰기 권한 |
| 7 | rwx | 읽기, 쓰기 및 실행 권한 |

다음 표에서는 심볼릭 모드로 파일 사용 권한을 설정하는 데 사용할 기호를 나열합니다. 기호는 사용 권한을 설정 또는 변경할 사용자, 수행할 연산 및 지정 또는 변경하려는 사용 권한을 지정할 수 있습니다.

**표 1-5** 심볼릭 모드로 파일 사용 권한 설정

| 기호 | 기능 | 설명 |
|---|---|---|
| u | *who* | 사용자(소유자) |
| g | *who* | 그룹 |
| o | *who* | 기타 |
| a | *who* | 모두 |
| = | *operator* | 지정 |
| + | *operator* | 추가 |
| - | *operator* | 제거 |
| r | *permissions* | 읽기 |
| w | *permissions* | 쓰기 |
| x | *permissions* | 실행 |
| l | *permissions* | 필수 잠금, setgid 비트 설정, 그룹 실행 비트 해제 |
| s | *permissions* | setuid 또는 setgid 비트 설정 |
| t | *permissions* | 고정된 비트 설정, 기타에 대한 실행 비트 설정 |

기능 열에 지정된 *who operator permissions*는 파일 또는 디렉토리 사용 권한을 변경하는 기호를 지정합니다.

*who* 사용 권한을 변경할 사용자를 지정합니다.

*operator* 수행할 연산을 지정합니다.

*permissions* 변경할 사용 권한을 지정합니다.

절대 모드 또는 심볼릭 모드로 특수 파일 사용 권한을 설정할 수 있습니다. 하지만 디렉토리에 대한 `setuid` 사용 권한을 설정하거나 제거하려면 심볼릭 모드를 사용해야 합니다. 절대 모드에서는 사용 권한 세 문자의 왼쪽에 새 8진수 값을 추가하여 특수 사용 권한을 설정합니다. 예 1-5. “절대 모드로 특수 파일 사용 권한 설정”를 참조하십시오. 다음 표에서는 특수 파일 사용 권한을 설정하는 데 사용할 8진수 값을 나열합니다.

**표 1-6** 절대 모드로 특수 파일 사용 권한 설정

| 8진수 값 | 특수 파일 사용 권한 |
|---|---|
| 1 | 고정된 비트 |
| 2 | `setgid` |
| 4 | `setuid` |

# 액세스 제어 목록을 사용하여 UFS 파일 보호

기존 UNIX 파일 보호에서는 세 가지 사용자 클래스(파일 소유자, 파일 그룹 및 기타)에 대해 읽기, 쓰기 및 실행 권한을 제공합니다. UFS 파일 시스템에서는 액세스 제어 목록(ACL)이 다음 작업을 가능하게 하는 향상된 파일 보안을 제공합니다.

- 파일 소유자, 그룹, 기타, 특정 사용자 및 그룹에 대한 파일 사용 권한 정의
- 위 범주 각각에 대한 기본 사용 권한 정의

---

**참고** - ZFS 파일 시스템의 ACL과 NFSv4 파일의 ACL에 대한 자세한 내용은 “Oracle Solaris 11.2의 ZFS 파일 시스템 관리 ”의 7 장, “ACL 및 속성을 사용하여 Oracle Solaris ZFS 파일 보호”을 참조하십시오.

---

예를 들어, 그룹의 모든 사용자가 파일을 읽을 수 있도록 하려는 경우 해당 파일에 대해 그룹 읽기 권한을 부여하면 됩니다. 하지만 그룹에서 사용자 한 명만 해당 파일에 쓸 수 있도록 하려면 ACL을 사용하면 됩니다.

UFS 파일 시스템의 ACL에 대한 자세한 내용은 Oracle Solaris 10 릴리스용 시스템 관리 설명서*:* 보안 서비스를 참조하십시오.

# 보안 손상으로부터 실행 파일 보호

프로그램은 스택의 데이터를 읽고 씁니다. 일반적으로 프로그램은 코드용으로 특별히 지정된 메모리의 읽기 전용 부분에서 실행됩니다. 스택 버퍼 오버플로우를 야기하는 일부 공격은

스택에 새 코드를 삽입하여 프로그램이 해당 코드를 실행하도록 합니다. 스택 메모리에서 실행 권한을 제거하면 이러한 공격이 성공하지 못하도록 방지됩니다. 즉, 대부분의 프로그램은 실행 가능 스택을 사용하지 않고도 제대로 작동할 수 있습니다.

64비트 프로세스에는 항상 실행할 수 없는 스택이 사용됩니다. 기본적으로 32비트 SPARC 프로세스에는 실행 가능한 스택이 포함됩니다. noexec_user_stack 변수를 사용하면 32비트 프로세스의 스택이 실행 가능한지 여부를 지정할 수 있습니다.

이 변수를 설정하면 스택에서 코드를 실행하려고 시도하는 프로그램에 SIGSEGV 신호가 전송됩니다. 일반적으로 이 신호가 전송되면 프로그램이 코어 덤프와 함께 종료됩니다. 또한 해당 프로그램은 잘못된 프로그램의 이름, 프로세스 ID 및 프로그램을 실행한 사용자의 실제 UID를 포함하는 경고 메시지를 생성합니다. 예를 들면 다음과 같습니다.

```
a.out[347] attempt to execute code on stack by uid 555
```

메시지는 syslog kern 기능이 notice 레벨로 설정된 경우 syslog 데몬을 통해 기록됩니다. 기본적으로 이 로깅은 syslog.conf 파일에서 설정되므로 콘솔과 /var/adm/messages 파일에 메시지가 전송됩니다. 자세한 내용은 syslogd(1M) 및 syslog.conf(4) 매뉴얼 페이지를 참조하십시오.

syslog 메시지는 잠재적인 보안 문제를 관찰하는 데 유용합니다. 또한 메시지는 noexec_user_stack 변수를 설정하여 올바른 작업으로부터 금지된 실행 가능 스택에 종속되어 있는 유효한 프로그램을 식별합니다. 메시지가 기록되지 않도록 하려면 /etc/system 파일에서 로그 변수 noexec_user_stack_log를 0으로 설정하십시오. 메시지가 기록되고 있지 않더라도 계속해서 SIGSEGV 신호는 실행 중인 프로그램이 코어 덤프와 함께 종료되도록 할 수 있습니다.

프로그램은 스택 실행을 명시적으로 표시하거나 방지할 수 있습니다. 프로그램의 mprotect() 함수는 스택을 명시적으로 실행 가능한 스택으로 표시합니다. 자세한 내용은 mprotect(2) 매뉴얼 페이지를 참조하십시오. -M /usr/lib/ld/map.noexstk로 컴파일된 프로그램은 시스템 차원의 설정에 관계없이 스택을 실행할 수 없는 스택으로 설정합니다.

# 파일 보호

다음 절차에서는 UNIX 사용 권한으로 파일을 보호하고, 보안 위험이 있는 파일을 찾고, 해당 파일로 인한 손상으로부터 시스템을 보호합니다.

## UNIX 권한으로 파일 보호

다음 작업 맵에서는 파일 사용 권한을 나열하고, 파일 사용 권한을 변경하고, 특수 파일 사용 권한으로 파일을 보호하는 절차에 대해 설명합니다.

| 작업 | 지침 |
| --- | --- |
| 파일 정보를 표시합니다. | 파일 정보 표시 방법 [15] |
| 로컬 파일 소유권을 변경합니다. | 파일 소유자 변경 방법 [16]<br>파일의 그룹 소유권 변경 방법 [17] |
| 로컬 파일 사용 권한을 변경합니다. | 심볼릭 모드로 파일 사용 권한 변경 방법 [17]<br>절대 모드로 파일 사용 권한 변경 방법 [18]<br>절대 모드로 특수 파일 사용 권한 변경 방법 [19] |

## ▼ 파일 정보 표시 방법

`ls` 명령을 사용하여 디렉토리의 모든 파일에 대한 정보를 표시합니다.

- **다음 명령을 입력하여 현재 디렉토리에 있는 모든 파일의 긴 목록을 표시합니다.**

```
% ls -la
```

`-l` 사용자 소유권, 그룹 소유권 및 파일 사용 권한이 포함된 긴 형식을 표시합니다.

`-a` 점(.)으로 시작하는 숨겨진 파일을 비롯하여 모든 파일을 표시합니다.

`ls` 명령에 대한 모든 옵션은 `ls`(1) 매뉴얼 페이지를 참조하십시오.

**예 1-1** 파일 정보 표시

다음 예에서는 `/sbin` 디렉토리에 있는 파일의 부분 목록이 표시됩니다.

```
% cd /sbin
% ls -l
total 4960
-r-xr-xr-x   1 root   bin      12756 Dec 19  2013 6to4relay
lrwxrwxrwx   1 root   root        10 Dec 19  2013 accept -> cupsaccept
-r-xr-xr-x   1 root   bin      38420 Dec 19  2013 acctadm
-r-xr-xr-x   2 root   sys      70512 Dec 19  2013 add_drv
-r-xr-xr-x   1 root   bin       3126 Dec 19  2013 addgnupghome
drwxr-xr-x   2 root   bin         37 Dec 19  2013 amd64
-r-xr-xr-x   1 root   bin       2264 Dec 19  2013 applygnupgdefaults
-r-xr-xr-x   1 root   bin        153 Dec 19  2013 archiveadm
-r-xr-xr-x   1 root   bin      12644 Dec 19  2013 arp
.
.
.
```

각 행에는 파일 정보가 다음 순서대로 표시됩니다.

- 파일 유형 - 예: `d`. 파일 유형 목록은 “파일 및 디렉토리 소유권” [8]을 참조하십시오.
- 사용 권한 - 예: `r-xr-xr-x`. 설명은 “파일 및 디렉토리 소유권” [8]을 참조하십시오.
- 하드 링크 수 - 예: `2`
- 파일 소유자 - 예: `root`
- 파일 그룹 - 예: `bin`
- 파일 크기(바이트) - 예: `12644`
- 파일을 만든 날짜 또는 파일을 마지막으로 변경한 날짜 - 예: `Dec 19 2013`
- 파일 이름 - 예: `arp`

## ▼ 파일 소유자 변경 방법

**시작하기 전에** 파일 또는 디렉토리 소유자가 아닌 경우 객체 액세스 관리 권한 프로파일에 지정되어 있어야 합니다. 공용 객체인 파일을 변경하려면 `root` 역할을 맡아야 합니다.

자세한 내용은 “Oracle Solaris 11.2의 사용자 및 프로세스 보안 ”의 “지정된 관리 권한 사용”을 참조하십시오.

1. **로컬 파일에 대한 권한을 표시합니다.**

```
% ls -l example-file
-rw-r--r--   1 janedoe   staff   112640 May 24 10:49 example-file
```

2. **파일 소유자를 변경합니다.**

```
# chown stacey example-file
```

3. **파일 소유자가 변경되었는지 확인합니다.**

```
# ls -l example-file
-rw-r--r--   1 stacey   staff   112640 May 26 08:50 example-file
```

NFS 마운트된 파일에 대한 사용 권한을 변경하려면 “Oracle Solaris 11.2의 네트워크 파일 시스템 관리 ”의 5 장, “네트워크 파일 시스템 관리 명령”을 참조하십시오.

**예 1-2** 사용자가 고유 파일의 소유권을 변경할 수 있도록 설정

**보안 고려 사항** - 특별한 이유가 있는 경우에만 `rstchown` 변수 설정을 0으로 변경해야 합니다. 기본 설정은 공간 쿼터를 무시하기 위해 기타에 속한 것처럼 하여 사용자가 파일을 나열하지 못하도록 합니다.

이 예에서는 `/etc/system` 파일에서 `rstchown` 변수의 값을 0으로 설정합니다. 이와 같이 설정하면 파일 소유자가 `chown` 명령을 사용하여 파일 소유권을 다른 사용자로 변경할 수 있습니

다. 또한 소유자는 chgrp 명령을 사용하여 파일의 그룹 소유권을 소유자가 속하지 않은 그룹으로 설정할 수 있습니다. 시스템을 재부트하면 변경 사항이 적용됩니다.

```
set rstchown = 0
```

자세한 내용은 chown(1) 및 chgrp(1) 매뉴얼 페이지를 참조하십시오.

## ▼ 파일의 그룹 소유권 변경 방법

**시작하기 전에** 파일 또는 디렉토리 소유자가 아닌 경우 객체 액세스 관리 권한 프로파일에 지정되어 있어야 합니다. 공용 객체인 파일을 변경하려면 root 역할을 맡아야 합니다.

자세한 내용은 "Oracle Solaris 11.2의 사용자 및 프로세스 보안 "의 "지정된 관리 권한 사용"을 참조하십시오.

1. **파일의 그룹 소유권을 변경합니다.**

```
% chgrp scifi example-file
```

그룹 설정에 대한 자세한 내용은 "Oracle Solaris 11.2의 사용자 계정 및 사용자 환경 관리 "의 1 장, "사용자 계정 및 사용자 환경 정보"를 참조하십시오.

2. **파일의 그룹 소유권이 변경되었는지 확인합니다.**

```
% ls -l example-file
-rw-r--r--   1 stacey   scifi   112640 June 20 08:55  example-file
```

예 1-2. "사용자가 고유 파일의 소유권을 변경할 수 있도록 설정"를 참조하십시오.

## ▼ 심볼릭 모드로 파일 사용 권한 변경 방법

다음 절차에서는 사용자가 소유한 파일의 사용 권한을 변경합니다.

1. **심볼릭 모드로 사용 권한을 변경합니다.**

```
% chmod who operator permissions filename
```

*who* 사용 권한을 변경할 사용자를 지정합니다.

*operator* 수행할 연산을 지정합니다.

*permissions* 변경할 사용 권한을 지정합니다. 유효한 기호 목록은 표 1-5. "심볼릭 모드로 파일 사용 권한 설정"를 참조하십시오.

*filename* 파일 또는 디렉토리를 지정합니다.

2. **파일 사용 권한이 변경되었는지 확인합니다.**

```
% ls -l filename
```

---

**참고** - 파일 또는 디렉토리 소유자가 아닌 경우 객체 액세스 관리 권한 프로파일에 지정되어 있어야 합니다. 공용 객체인 파일을 변경하려면 `root` 역할을 맡아야 합니다.

---

**예 1-3** 심볼릭 모드로 사용 권한 변경

다음 예에서는 소유자가 기타에 대한 읽기 권한을 제거합니다.

```
% chmod o-r example-file1
```

다음 예에서는 소유자가 사용자, 그룹 및 기타에 대한 읽기 및 실행 권한을 추가합니다.

```
% chmod a+rx example-file2
```

다음 예에서는 소유자가 그룹 구성원에 대한 읽기, 쓰기 및 실행 권한을 추가합니다.

```
% chmod g=rwx example-file3
```

## ▼ 절대 모드로 파일 사용 권한 변경 방법

다음 절차에서는 사용자가 소유한 파일의 사용 권한을 변경합니다.

1. **절대 모드로 사용 권한을 변경합니다.**

```
% chmod nnn filename
```

*nnn* 파일 소유자, 파일 그룹 및 기타(해당 순서대로)에 대한 사용 권한을 나타내는 8진수 값을 지정합니다. 유효한 8진수 값 목록은 표 1-4. “절대 모드로 파일 사용 권한 설정”를 참조하십시오.

*filename* 파일 또는 디렉토리를 지정합니다.

---

**참고** - `chmod` 명령을 사용하여 기존 ACL 항목이 포함된 객체에서 파일 또는 디렉토리 권한을 변경할 경우 ACL 항목도 함께 변경될 수 있습니다. 정확한 변경 사항은 `chmod` 사용 권한 작업 변경 사항과 파일 시스템의 `aclmode` 및 `aclinherit` 등록 정보 값에 따라 달라집니다.

자세한 내용은 “Oracle Solaris 11.2의 ZFS 파일 시스템 관리 ”의 7 장, “ACL 및 속성을 사용하여 Oracle Solaris ZFS 파일 보호”을 참조하십시오.

---

2. **파일 사용 권한이 변경되었는지 확인합니다.**

```
% ls -l filename
```

---

**참고** - 파일 또는 디렉토리 소유자가 아닌 경우 객체 액세스 관리 권한 프로파일에 지정되어 있어야 합니다. 공용 객체인 파일을 변경하려면 `root` 역할을 맡아야 합니다.

---

**예 1-4** 절대 모드로 사용 권한 변경

다음 예에서는 관리자가 공용 디렉토리의 사용 권한을 744(읽기, 쓰기, 실행/읽기 전용/읽기 전용)에서 755(읽기, 쓰기, 실행/읽기 및 실행/읽기 및 실행)로 변경합니다.

```
# ls -ld public_dir
drwxr--r--  1 jdoe   staff    6023 Aug  5 12:06 public_dir
# chmod 755 public_dir
# ls -ld public_dir
drwxr-xr-x  1 jdoe   staff    6023 Aug  5 12:06 public_dir
```

다음 예에서는 파일 소유자가 실행 가능 셸 스크립트의 사용 권한을 읽기 및 쓰기에서 읽기, 쓰기 및 실행으로 변경합니다.

```
% ls -l my_script
-rw------- 1 jdoe   staff    6023 Aug  5 12:06 my_script
% chmod 700 my_script
% ls -l my_script
-rwx------ 1 jdoe   staff    6023 Aug  5 12:06 my_script
```

## ▼ 절대 모드로 특수 파일 사용 권한 변경 방법

**시작하기 전에** 파일 또는 디렉토리 소유자가 아닌 경우 객체 액세스 관리 권한 프로파일에 지정되어 있어야 합니다. 공용 객체인 파일을 변경하려면 `root` 역할을 맡아야 합니다.

자세한 내용은 "Oracle Solaris 11.2의 사용자 및 프로세스 보안 "의 "지정된 관리 권한 사용"을 참조하십시오.

1. **절대 모드로 특수 사용 권한을 변경합니다.**

```
% chmod nnnn filename
```

*nnnn* 파일 또는 디렉토리 사용 권한을 변경하는 8진수 값을 지정합니다. 맨 왼쪽에 있는 8진수 값은 특수 파일 사용 권한을 설정합니다. 특수 사용 권한에 대한 유효한 8진수 값 목록은 표 1-6. "절대 모드로 특수 파일 사용 권한 설정"을 참조하십시오.

*filename* 파일 또는 디렉토리를 지정합니다.

**참고 -** chmod 명령을 사용하여 ACL 항목이 있는 파일의 파일 그룹 사용 권한을 변경하면 파일 그룹 사용 권한과 ACL 마스크가 모두 새 사용 권한으로 변경됩니다. 새 ACL 마스크 사용 권한은 파일에 ACL 항목이 있는 추가 사용자 및 그룹에 대한 사용 권한을 변경할 수 있습니다. 모든 ACL 항목에 대해 적절한 사용 권한이 설정되도록 하려면 getfacl 명령을 사용하십시오. 자세한 내용은 getfacl(1) 매뉴얼 페이지를 참조하십시오.

2. **파일 사용 권한이 변경되었는지 확인합니다.**

```
% ls -l filename
```

**예 1-5** 절대 모드로 특수 파일 사용 권한 설정

다음 예에서는 관리자가 dbprog 파일에 대해 setuid 권한을 설정합니다.

```
# chmod 4555 dbprog
# ls -l dbprog
-r-sr-xr-x   1 db     staff        12095 May  6 09:29 dbprog
```

다음 예에서는 관리자가 dbprog2 파일에 대해 setgid 권한을 설정합니다.

```
# chmod 2551 dbprog2
# ls -l dbprog2
-r-xr-s--x   1 db     staff       24576 May  6 09:30 dbprog2
```

다음 예에서는 관리자가 public_dir 디렉토리에 대해 고정된 비트를 설정합니다.

```
# chmod 1777 public_dir
# ls -ld public_dir
drwxrwxrwt   2 jdoe   staff          512 May 15 15:27 public_dir
```

# 보안 위험이 있는 프로그램 방지

다음 작업 맵에서는 시스템에서 위험성이 있는 실행 파일을 찾고 프로그램이 실행 가능 스택을 악용하지 못하도록 하는 절차에 대해 설명합니다.

| 작업 | 설명 | 지침 |
|---|---|---|
| 특수 사용 권한을 가진 파일을 찾습니다. | setuid 비트가 설정되었지만 root 사용자가 소유하지 않은 파일을 찾습니다. | 특수 파일 사용 권한이 있는 파일을 찾는 방법 [21] |
| 실행 가능 스택이 오버플로우되지 않도록 합니다. | 프로그램이 실행 가능 스택을 악용하지 않도록 합니다. | 프로그램이 실행 가능 스택을 사용 안함으로 설정하는 방법 [22] |
| 실행 가능 스택 메시지가 기록되지 않도록 합니다. | 실행 가능 스택 메시지의 기록을 해제합니다. | 예 1-7. “실행 가능 스택 메시지 로깅을 사용 안함으로 설정” |

# ▼ 특수 파일 사용 권한이 있는 파일을 찾는 방법

이 절차에서는 프로그램에서 setuid 및 setgid 사용 권한의 잠재적인 무단 사용을 찾습니다. root 또는 bin 이외의 다른 사용자에게 소유권을 부여하는 실행 파일은 의심스러운 것입니다.

**시작하기 전에** root 역할을 맡아야 합니다. 자세한 내용은 "Oracle Solaris 11.2의 사용자 및 프로세스 보안"의 "지정된 관리 권한 사용"을 참조하십시오.

1. **find 명령을 사용하여 setuid 사용 권한이 있는 파일을 찾습니다.**

```
# find directory -user root -perm -4000 -exec ls -ldb {} \; >/tmp/filename
```

| | |
|---|---|
| find *directory* | 지정된 *directory*에서 시작하여 마운트된 경로를 모두 검사합니다. 여기서 directory는 루트(/), /usr, /opt 등일 수 있습니다. |
| -user root | root가 소유한 파일만 표시합니다. |
| -perm -4000 | 사용 권한이 4000으로 설정된 파일만 표시합니다. |
| -exec ls -ldb | find 명령 출력을 ls -ldb 형식으로 표시합니다. ls(1) 매뉴얼 페이지를 참조하십시오. |
| /tmp/*filename* | find 명령 결과가 포함된 파일입니다. |

자세한 내용은 find(1) 항목을 참조하십시오.

2. **/tmp/*filename*에 결과를 표시합니다.**

```
# more /tmp/filename
```

배경 정보는 "setuid 사용 권한" [10]을 참조하십시오.

**예 1-6** setuid 사용 권한이 있는 파일 찾기

다음 예의 출력에서는 rar이라는 그룹의 사용자가 /usr/bin/rlogin의 개인용 복사본을 만들고 setuid 권한을 root로 설정했음을 보여줍니다. 따라서 /usr/rar/bin/rlogin 프로그램이 root 권한으로 실행됩니다.

/usr/rar 디렉토리를 조사하고 /usr/rar/bin/rlogin 명령을 제거한 후 관리자가 find 명령에서 출력을 아카이브합니다.

```
# find /usr -user root -perm -4000 -exec ls -ldb {} \; > /var/tmp/ckprm
# cat /var/tmp/ckprm
-rwsr-xr-x  1 root   sys   28000  Jul 14 14:14  /usr/bin/atq
-rwsr-xr-x  1 root   sys   32364  Jul 14 14:14  /usr/bin/atrm
-r-sr-xr-x  1 root   sys   41432  Jul 14 14:14  /usr/bin/chkey
-rwsr-xr-x  1 root   bin   82804  Jul 14 14:14  /usr/bin/cdrw
```

```
-r-sr-xr-x  1 root   bin    8008  Jul 14 14:14  /usr/bin/mailq
-r-sr-sr-x  1 root   sys   45348  Jul 14 14:14  /usr/bin/passwd
-rwsr-xr-x  1 root   bin   37724  Jul 14 14:14  /usr/bin/pfedit
-r-sr-xr-x  1 root   bin   51440  Jul 14 14:14  /usr/bin/rcp
---s--x---  1 root   rar   41592  Jul 24 16:14  /usr/rar/bin/rlogin
-r-s--x--x  1 root   bin  166908  Jul 14 14:14  /usr/bin/sudo
-r-sr-xr-x  4 root   bin   24024  Jul 14 14:14  /usr/bin/uptime
-r-sr-xr-x  1 root   bin   79488  Jul 14 14:14  /usr/bin/xlock
# mv /var/tmp/ckprm /var/share/sysreports/ckprm
```

## ▼ 프로그램이 실행 가능 스택을 사용 안함으로 설정하는 방법

32비트 실행 가능 스택의 보안 위험에 대한 설명은 "보안 손상으로부터 실행 파일 보호" [13]를 참조하십시오.

**시작하기 전에** `root` 역할을 맡아야 합니다. 자세한 내용은 "Oracle Solaris 11.2의 사용자 및 프로세스 보안"의 "지정된 관리 권한 사용"을 참조하십시오.

1. **`/etc/system` 파일을 편집하고 다음 행을 추가합니다.**

```
# pfedit /etc/system
...
set noexec_user_stack=1
set noexec_user_stack_log=1
```

2. **시스템을 재부트합니다.**

```
# reboot
```

**예 1-7** 실행 가능 스택 메시지 로깅을 사용 안함으로 설정

이 예에서 관리자는 실행 가능 스택 메시지 로깅을 사용 안함으로 설정한 후 시스템을 재부트합니다.

```
# cat /etc/system
set noexec_user_stack=1
set noexec_user_stack_log=0
# reboot
```

**참조** 자세한 내용은 다음을 참조하십시오.

- https://blogs.oracle.com/gbrunett/entry/solaris_non_executable_stack_overview
- https://blogs.oracle.com/gbrunett/entry/solaris_non_executable_stack_continued

- https://blogs.oracle.com/gbrunett/entry/solaris_non_executable_stack_concluded

♦ ♦ ♦ 2 장

# 2 BART를 사용하여 파일 무결성 확인

이 장에서는 파일 무결성 도구인 BART에 대해 설명합니다. BART는 시간별로 시스템에서 파일의 무결성을 확인할 수 있게 해주는 명령줄 도구입니다. 이 장에서는 다음 내용을 다룹니다.



## BART 정보

BART는 암호화 강도 체크섬과 파일 시스템 메타 데이터를 사용하여 변경 사항을 확인하는 파일 무결성 검사 및 보고 도구입니다. BART를 사용하면 손상되었거나 일반적이지 않은 파일을 식별하여 보안 위반을 검색하거나 시스템에서 성능 문제를 해결할 수 있습니다. BART를 사용하면 배치된 시스템에 설치된 파일의 불일치를 쉽고 안정적으로 보고하여 시스템 네트워크를 관리하는 비용을 줄일 수 있습니다.

BART를 사용하면 알려진 기준과 비교하여 시스템에서 발생한 파일 레벨 변경 사항을 확인할 수 있습니다. BART를 사용하여 완전하게 설치 및 구성된 시스템에서 기준 또는 제어 매니페스트를 만들 수 있습니다. 그러면 나중에 시스템의 스냅샷을 이 기준과 비교하여 설치된 이후 시스템에서 발생한 파일 레벨 변경 사항을 나열하는 보고서를 생성할 수 있습니다.

### BART 기능

BART는 강력하고 유연한 방식의 간단한 구문을 사용합니다. 이 도구를 사용하면 특정 시스템에서 시간별로 파일 변경 사항을 추적할 수 있습니다. 또한 비슷한 시스템 간의 파일 차이를 추적할 수도 있습니다. 이러한 비교를 통해 손상되었거나 일반적이지 않은 파일을 찾아내거나 소프트웨어가 오래된 시스템을 확인할 수 있습니다.

BART의 추가 이점 및 용도에는 다음이 포함됩니다.

- 모니터할 파일을 지정할 수 있습니다. 예를 들어, 소프트웨어를 쉽고 효율적으로 재구성하는 데 도움이 되는 로컬 사용자 정의를 모니터할 수 있습니다.

- 시스템 성능 문제를 해결할 수 있습니다.

# BART 구성 요소

BART는 두 가지 기본 파일인 매니페스트와 비교 파일 또는 보고서를 만듭니다. 선택적 규칙 파일을 사용하면 매니페스트와 보고서를 사용자 정의할 수 있습니다.

## BART 매니페스트

매니페스트는 특정 시점의 시스템에 대한 파일 레벨 스냅샷입니다. 매니페스트에는 체크섬과 같이 고유하게 식별하는 정보를 포함할 수 있는 파일의 속성에 대한 정보가 포함됩니다. `bart create` 명령에 대한 옵션은 특정 파일 및 디렉토리를 대상으로 지정할 수 있습니다. 규칙 파일은 "BART 규칙 파일" [27]에 설명된 대로 보다 세부적인 필터링을 제공할 수 있습니다.

---

**참고** - 기본적으로 BART는 루트(/) 디렉토리 아래에 모든 ZFS 파일 시스템을 카탈로그로 작성할 수 있습니다. 다른 파일 시스템 유형(예: NFS 또는 TMPFS 파일 시스템)에서는 마운트된 CD-ROM이 카탈로그에 저장됩니다.

---

초기 Oracle Solaris 설치 직후 시스템의 매니페스트를 만들 수 있습니다. 또한 사이트의 보안 정책을 충족하도록 시스템을 구성한 후 매니페스트를 만들 수도 있습니다. 이 유형의 제어 매니페스트는 이후 비교를 위한 기준을 제공합니다.

기준 매니페스트를 사용하면 시간별로 동일 시스템에서 파일 무결성을 추적할 수 있습니다. 또한 다른 시스템과의 비교를 위한 기본으로 사용될 수도 있습니다. 예를 들어, 네트워크에서 다른 시스템의 스냅샷을 작성한 후 이러한 매니페스트를 기준 매니페스트와 비교할 수 있습니다. 보고된 파일 불일치는 다른 시스템을 기준 시스템과 동기화해야 함을 나타냅니다.

매니페스트 형식은 "BART 매니페스트 파일 형식" [37]을 참조하십시오. 매니페스트를 만들려면 `How to Create a Control Manifest`에 설명된 대로 제어 매니페스트를 만드는 방법 [28] 명령을 사용합니다.

## BART 보고서

BART 보고서에는 두 매니페스트 간의 파일별 불일치 항목이 나열됩니다. 불일치는 두 매니페스트에 대해 카탈로그화된 해당 파일의 속성 변경 사항입니다. 파일 항목 추가나 삭제도 불일치로 간주됩니다.

효과적인 비교를 위해서는 두 매니페스트의 대상 파일 시스템이 동일해야 합니다. 또한 동일 옵션 및 규칙 파일을 사용하여 매니페스트를 만들고 비교해야 합니다.

보고서 형식은 “BART 보고” [40]를 참조하십시오. 보고서를 만들려면 `How to Compare Manifests for the Same System Over Time`에 설명된 대로 시간에 따라 동일 시스템에 대한 매니페스트를 비교하는 방법 [31] 명령을 사용합니다.

### BART 규칙 파일

BART 규칙 파일은 특정 파일 및 파일 속성을 포함하거나 제외하도록 필터링하거나 대상으로 지정하기 위해 만드는 파일입니다. 그런 후 BART 매니페스트 및 보고서를 만들 때 이 파일을 사용할 수 있습니다. 매니페스트를 비교할 때 규칙 파일은 매니페스트 간의 불일치에 대해 플래그를 지정하는 데 도움이 됩니다.

---

**참고** - 규칙 파일을 사용하여 매니페스트를 만들 때는 동일한 규칙 파일을 사용하여 비교 매니페스트를 만들어야 합니다. 매니페스트를 비교할 때도 또한 규칙 파일을 사용해야 합니다. 그렇지 않으면 보고서에 잘못된 불일치가 많이 나열될 수 있습니다.

---

규칙 파일을 사용하여 시스템의 특정 파일 및 파일 속성을 모니터하려면 계획이 필요합니다. 규칙 파일을 만들기 전에 시스템에서 모니터할 파일 및 파일 속성을 결정하십시오.

사용자 오류로 인해 규칙 파일에도 구문 오류 및 기타 모호한 정보가 포함될 수 있습니다. 규칙 파일에 오류가 포함된 경우 해당 오류도 보고됩니다.

규칙 파일 형식은 “BART 규칙 파일 형식” [38] 및 `bart_rules(4)` 매뉴얼 페이지를 참조하십시오. 규칙 파일을 만들려면 규칙 파일을 사용하여 BART 보고서를 사용자 정의하는 방법 [36]을 참조하십시오.

## BART 사용 정보

`bart` 명령은 매니페스트를 만들고 비교하는 데 사용됩니다. 이 명령은 모든 사용자가 실행할 수 있습니다. 하지만 사용자는 자신에게 액세스 권한이 있는 파일만 카탈로그로 작성하고 모니터할 수 있습니다. 따라서 사용자 및 대부분의 역할은 자신의 홈 디렉토리에 있는 파일을 유용하게 카탈로그로 작성할 수 있지만 `root` 계정은 시스템 파일을 포함하여 모든 파일을 카탈로그로 작성할 수 있습니다.

## BART 보안 고려 사항

BART 매니페스트 및 보고서는 누구나 읽을 수 있습니다. BART 출력 결과에 민감한 정보가 포함될 수 있는 경우 출력 결과를 보호하기 위한 적합한 조치를 취하십시오. 예를 들어, 제한

적인 권한으로 출력 파일을 생성하거나 출력 결과 파일을 보호되는 디렉토리에 배치하는 옵션을 사용합니다.

## BART 사용

| 작업 | 설명 | 지침 |
|---|---|---|
| BART 매니페스트를 만듭니다. | 시스템에 설치된 모든 파일에 대한 정보 목록을 생성합니다. | 제어 매니페스트를 만드는 방법 [28] |
| 사용자 정의 BART 매니페스트를 만듭니다. | 시스템에 설치된 특정 파일에 대한 정보 목록을 생성합니다. | 매니페스트를 사용자 정의하는 방법 [30] |
| BART 매니페스트를 비교합니다. | 시간에 따른 시스템 변경 사항을 비교하는 보고서를 생성합니다.<br><br>또는 하나 이상의 시스템을 제어 시스템과 비교하는 보고서를 생성합니다. | 시간에 따라 동일 시스템에 대한 매니페스트를 비교하는 방법 [31]<br><br>여러 시스템의 매니페스트를 비교하는 방법 [33] |
| (옵션) BART 보고서를 사용자 정의합니다. | 다음 중 하나의 방법으로 사용자 정의 BART 보고서를 생성합니다.<br><br>■ 속성 지정<br>■ 규칙 파일 사용 | 파일 속성을 지정하여 BART 보고서를 사용자 정의하는 방법 [35]<br><br>규칙 파일을 사용하여 BART 보고서를 사용자 정의하는 방법 [36] |

## ▼ 제어 매니페스트를 만드는 방법

이 절차에서는 비교를 위해 기준 또는 제어 매니페스트를 만드는 방법을 설명합니다. 중앙 이미지에서 많은 시스템을 설치할 때 이 유형의 매니페스트를 사용합니다. 또는 설치가 동일한지 확인하려는 경우 이 유형의 매니페스트를 사용하여 비교를 실행합니다. 제어 매니페스트에 대한 자세한 내용은 “BART 매니페스트” [26]를 참조하십시오. 형식 규칙을 이해하려면 예 2-1. “BART 매니페스트 형식에 대한 설명”을 참조하십시오.

**참고** - 네트워크 파일 시스템을 카탈로그로 작성하도록 시도하지 마십시오. BART를 사용하여 네트워크 파일 시스템을 모니터하면 별로 가치가 없는 매니페스트를 생성하는 데 많은 리소스가 소모됩니다.

**시작하기 전에** `root` 역할을 맡아야 합니다. 자세한 내용은 “Oracle Solaris 11.2의 사용자 및 프로세스 보안”의 “지정된 관리 권한 사용”을 참조하십시오.

1. **사이트의 보안 요구 사항에 따라 Oracle Solaris 시스템을 사용자 정의한 후 제어 매니페스트를 만들고 출력을 파일로 재지정합니다.**

   ```
   # bart create options > control-manifest
   ```

`-R` 매니페스트에 대한 루트 디렉토리를 지정합니다. 규칙에서 지정한 모든 경로는 이 디렉토리에 대한 상대 경로로 해석됩니다. 매니페스트에서 보고하는 모든 경로는 이 디렉토리에 대한 상대 경로입니다.

`-I` 명령줄 또는 표준 입력에서 읽은 카탈로그화할 개별 파일 목록을 사용합니다.

`-r` 이 매니페스트에 대한 규칙 파일 이름입니다. `-` 인수는 표준 입력으로부터 규칙 파일을 읽습니다.

`-n` 파일 목록의 모든 일반 파일에 대한 내용 서명을 해제합니다. 이 옵션은 성능을 향상시키는 데 사용할 수 있습니다. 또는 시스템 로그 파일과 같이 파일 목록의 내용이 변경될 것으로 예상되는 경우 이 옵션을 사용할 수 있습니다.

2. **매니페스트의 내용을 검토합니다.**
   형식에 대한 설명은 예 2-1. “BART 매니페스트 형식에 대한 설명”을 참조하십시오.

3. **(옵션) 매니페스트를 보호합니다.**
   시스템 매니페스트를 보호하기 위한 한 가지 방법은 `root` 계정만 액세스할 수 있는 디렉토리에 매니페스트를 두는 것입니다.

```
# mkdir /var/adm/log/bartlogs
# chmod 700 /var/adm/log/bartlogs
# mv control-manifest /var/adm/log/bartlogs
```

   매니페스트에 대한 의미 있는 이름을 선택합니다. 예를 들어, `mach1-120313`에서와 같이 매니페스트가 생성된 시스템 이름과 날짜를 사용합니다.

**예 2-1** BART 매니페스트 형식에 대한 설명

이 예에서는 매니페스트 형식에 대한 설명이 샘플 출력 뒤에 나옵니다.

```
# bart create
! Version 1.1
! HASH SHA256
! Saturday, September 07, 2013 (22:22:27)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/ D 1024 40755 user::rwx,group::r-x,mask:r-x,other:r-x
3ebc418eb5be3729ffe7e54053be2d33ee884205502c81ae9689cd8cca5b0090 0 0
.
```

```
.
.
/zone D 512 40755 user::rwx group::r-x,mask:r-x,other:r-x 3f81e892
154de3e7bdfd0d57a074c9fae0896a9e2e04bebfe5e872d273b063319e57f334 0 0
.
.
.
```

각 매니페스트는 헤더와 파일 항목으로 구성됩니다. 각 파일 항목은 파일 유형에 따라 단일 행입니다. 예를 들어, 위의 출력에서 각 파일 항목에 대해 `F` 유형은 파일을 지정하고 `D` 유형은 디렉토리를 지정합니다. 또한 크기, 내용, 사용자 ID, 그룹 ID 및 권한에 대한 정보가 나열됩니다. 출력의 파일 항목은 특수 문자를 올바르게 처리하기 위해 파일 이름의 인코딩된 버전별로 정렬됩니다. 모든 항목은 파일 이름을 기준으로 오름차순으로 정렬됩니다. 내장된 개행 또는 탭 문자를 포함하는 파일 이름과 같은 모든 비표준 파일 이름은 정렬되기 전에 비표준 문자를 따옴표로 묶습니다.

`!`로 시작하는 행은 매니페스트에 대한 메타 데이터를 제공합니다. 매니페스트 버전 행은 매니페스트 사양 버전을 나타냅니다. 해시 행은 사용된 해시 방식을 나타냅니다. 체크섬으로 사용되는 `SHA256` 해시에 대한 자세한 내용은 `sha2`(3EXT) 매뉴얼 페이지를 참조하십시오.

날짜 행은 매니페스트가 만들어진 날짜를 날짜 형식으로 표시합니다. `date`(1) 매뉴얼 페이지를 참조하십시오. 일부 행은 매니페스트 비교 도구에서 무시됩니다. 무시되는 행에는 메타 데이터, 빈 행, 공백으로만 구성된 행 및 `#`으로 시작하는 주석이 포함됩니다.

## ▼ 매니페스트를 사용자 정의하는 방법

다음 중 하나의 방법으로 매니페스트를 사용자가 정의할 수 있습니다.

- 하위 트리 지정

  개별 하위 트리 지정은 `/etc` 디렉토리의 모든 파일과 같이 선택한 중요한 파일에 대한 변경 사항을 효율적으로 모니터할 수 있는 방법입니다.
- 파일 이름 지정

  파일 이름 지정은 데이터베이스 응용 프로그램을 구성하고 실행하는 파일과 같은 특히 민감한 파일을 효율적으로 모니터할 수 있는 방법입니다.
- 규칙 파일 사용

  규칙 파일을 사용하여 매니페스트를 만들고 비교하면 하나 이상의 파일이나 하위 트리에 대해 여러 속성을 지정할 수 있는 유연성이 제공됩니다. 명령줄에서 매니페스트 또는 보고서의 모든 파일에 적용되는 전역 속성 정의를 지정할 수 있습니다. 규칙 파일에서 전역으로 적용되지 않는 속성을 지정할 수 있습니다.

**시작하기 전에** `root` 역할을 맡아야 합니다. 자세한 내용은 "Oracle Solaris 11.2의 사용자 및 프로세스 보안"의 "지정된 관리 권한 사용"을 참조하십시오.

1. **카탈로그화 및 모니터할 파일을 확인합니다.**

2. **다음 옵션 중 하나를 사용하여 사용자 정의 매니페스트를 만듭니다.**

   - 하위 트리 지정:

     ```
     # bart create -R subtree
     ```

   - 파일 이름 지정:

     ```
     # bart create -I filename...
     ```

     예를 들면 다음과 같습니다.

     ```
     # bart create -I /etc/system /etc/passwd /etc/shadow
     ```

   - 규칙 파일 사용:

     ```
     # bart create -r rules-file
     ```

3. **매니페스트의 내용을 검토합니다.**

4. **(옵션) 이후에 사용할 수 있도록 매니페스트를 보호되는 디렉토리에 저장합니다.**
   예는 제어 매니페스트를 만드는 방법 [28]의 3단계을 참조하십시오.

---

**작은 정보** - 규칙 파일을 사용한 경우 매니페스트와 함께 규칙 파일을 저장합니다. 효율적인 비교를 위해서는 규칙 파일을 사용해서 비교를 실행해야 합니다.

---

## ▼ 시간에 따라 동일 시스템에 대한 매니페스트를 비교하는 방법

시간별로 매니페스트를 비교하면 손상되었거나 일반적이지 않은 파일을 찾아내고, 보안 위반 사항을 검색하고, 시스템의 성능 문제를 해결할 수 있습니다.

**시작하기 전에** `root` 역할을 맡아야 합니다. 자세한 내용은 "Oracle Solaris 11.2의 사용자 및 프로세스 보안"의 "지정된 관리 권한 사용"을 참조하십시오.

1. **시스템에서 모니터할 파일의 제어 매니페스트를 만듭니다.**

   ```
   # bart create -R /etc > control-manifest
   ```

2. **(옵션) 이후에 사용할 수 있도록 매니페스트를 보호되는 디렉토리에 저장합니다.**
   예는 제어 매니페스트를 만드는 방법 [28]의 3단계을 참조하십시오.

3. **나중에 제어 매니페스트에 대해 동일한 매니페스트를 준비합니다.**

   ```
   # bart create -R /etc > test-manifest
   ```

4. **두번째 매니페스트를 보호합니다.**

```
# mv test-manifest /var/adm/log/bartlogs
```

5. **두 매니페스트를 비교합니다.**
   매니페스트를 만드는 데 사용한 동일한 명령줄 옵션 및 규칙 파일을 사용하여 매니페스트를 비교합니다.

   ```
   # bart compare options control-manifest test-manifest > bart-report
   ```

6. **BART 보고서에서 이상한 점을 검토합니다.**

**예 2-2** 시간별로 동일 시스템의 파일 변경 사항 추적

이 예에서는 시간별로 /etc 디렉토리의 변경 사항을 추적하는 방법을 보여줍니다. 이 유형의 비교를 통해 손상된 시스템에서 중요한 파일을 찾을 수 있습니다.

- 제어 매니페스트를 만듭니다.

```
# cd /var/adm/logs/manifests
# bart create -R /etc > system1.control.090713
! Version 1.1
! HASH SHA256
! Saturday, September 07, 2013 (11:11:17)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/.cpr_config F 2236 100644 owner@:read_data/write_data/append_data/read_xattr/wr
ite_xattr/read_attributes/write_attributes/read_acl/write_acl/write_owner/synchr
onize:allow,group@:read_data/read_xattr/read_attributes/read_acl/synchronize:all
ow,everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
4e271c59 0 0 3ebc418eb5be3729ffe7e54053be2d33ee884205502c81ae9689cd8cca5b0090
/.login F 1429 100644 owner@:read_data/write_data/append_data/read_xattr/write_x
attr/read_attributes/write_attributes/read_acl/write_acl/write_owner/synchronize
:allow,group@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow,ev
eryone@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
4bf9d6d7 0 3 ff6251a473a53de68ce8b4036d0f569838cff107caf1dd9fd04701c48f09242e
.
.
.
```

- 나중에 동일한 명령줄 옵션을 사용하여 테스트 매니페스트를 만듭니다.

```
# bart create -R /etc > system1.test.101013
Version 1.1
```

```
! HASH SHA256
! Monday, October 10, 2013 (10:10:17)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/.cpr_config F 2236 100644 owner@:read_data/write_data/append_data/read_xattr/wr
ite_xattr/read_attributes/write_attributes/read_acl/write_acl/write_owner/synchr
onize:allow,group@:read_data/read_xattr/read_attributes/read_acl/synchronize:all
ow,everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
4e271c59 0 0 3ebc418eb5be3729ffe7e54053be2d33ee884205502c81ae9689cd8cca5b0090
.
.
.
```

- 매니페스트를 비교합니다.

```
# bart compare system1.control.090713 system1.test.101013
/security/audit_class
mtime  4f272f59
```

출력은 제어 매니페스트가 만들어진 이후 `audit_class` 파일에 대한 수정 시간이 변경되었음을 나타냅니다. 이 변경 사항이 예상치 않은 것이면 추가 조사를 진행할 수 있습니다.

## ▼ 여러 시스템의 매니페스트를 비교하는 방법

다른 시스템의 매니페스트를 비교하면 시스템이 동일하게 설치되었는지 또는 동기화된 상태로 업그레이드되었는지를 확인할 수 있습니다. 예를 들어, 특정 보안 대상에 대해 시스템을 사용자 정의한 경우, 이 비교를 통해 보안 대상을 나타내는 매니페스트와 다른 시스템의 매니페스트 간의 모든 불일치 항목을 찾을 수 있습니다.

**시작하기 전에** `root` 역할을 맡아야 합니다. 자세한 내용은 "Oracle Solaris 11.2의 사용자 및 프로세스 보안"의 "지정된 관리 권한 사용"을 참조하십시오.

1. **제어 매니페스트를 만듭니다.**

   ```
   # bart create options > control-manifest
   ```

   옵션은 `bart`(1M) 매뉴얼 페이지를 참조하십시오.

2. **(옵션) 이후에 사용할 수 있도록 매니페스트를 보호되는 디렉토리에 저장합니다.**

   예는 제어 매니페스트를 만드는 방법 [28]의 3단계을 참조하십시오.

3. **테스트 시스템에서 동일한 `bart` 옵션을 사용하여 매니페스트를 만듭니다.**

   ```
   # bart create options > test1-manifest
   ```

4. **(옵션) 이후에 사용할 수 있도록 매니페스트를 보호되는 디렉토리에 저장합니다.**

5. **비교를 수행하려면 매니페스트를 중앙 위치에 복사합니다.**
   예를 들면 다음과 같습니다.

   ```
   # cp control-manifest /net/test-server/var/adm/logs/bartlogs
   ```

   테스트 시스템이 NFS 마운트 시스템이 아닌 경우 `sftp` 또는 다른 신뢰할 수 있는 방법을 사용하여 매니페스트를 중앙 위치에 복사합니다.

6. **매니페스트를 비교하고 출력 결과를 파일로 재지정합니다.**

   ```
   # bart compare control-manifest test1-manifest > test1.report
   ```

7. **BART 보고서에서 이상한 점을 검토합니다.**

**예 2-3** `/usr/bin` 디렉토리에서 의심스러운 파일 식별

이 예에서는 두 시스템에서 `/usr/bin` 디렉토리의 내용을 비교합니다.

- 제어 매니페스트를 만듭니다.

```
# bart create -R /usr/bin > control-manifest.090713
! Version 1.1
! HASH SHA256
! Saturday, September 07, 2013 (11:11:17)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/2to3 F 105 100555 owner@:read_data/read_xattr/write_xattr/execute/read_attribut
es/write_attributes/read_acl/write_acl/write_owner/synchronize:allow,group@:read
_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow,everyone@:re
ad_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow 4bf9d261 0
2 154de3e7bdfd0d57a074c9fae0896a9e2e04bebfe5e872d273b063319e57f334
/7z F 509220 100555 owner@:read_data/read_xattr/write_xattr/execute/read_attribu
tes/write_attributes/read_acl/write_acl/write_owner/synchronize:allow,group@:rea
d_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow,everyone@:r
ead_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow 4dadc48a 0
2 3ecd418eb5be3729ffe7e54053be2d33ee884205502c81ae9689cd8cca5b0090
...
```

- 제어 시스템과 비교하고자 하는 각 시스템에 대해 동일한 매니페스트를 만듭니다.

```
# bart create -R /usr/bin > system2-manifest.101013
! Version 1.1
! HASH SHA256
! Monday, October 10, 2013 (10:10:22)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/2to3 F 105 100555 owner@:read_data/read_xattr/write_xattr/execute/read_attribut
es/write_attributes/read_acl/write_acl/write_owner/synchronize:allow,group@:read
_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow,everyone@:re
ad_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow 4bf9d261 0
2 154de3e7bdfd0d57a074c9fae0896a9e2e04bebfe5e872d273b063319e57f334
...
```

- 매니페스트를 동일한 위치에 복사합니다.

```
# cp control-manifest.090713 /net/system2.central/bart/manifests
```

- 매니페스트를 비교합니다.

```
# bart compare control-manifest.090713 system2.test.101013 > system2.report
/su:
gid  control:3  test:1
/ypcat:
mtime  control:3fd72511  test:3fd9eb23
```

출력 결과는 /usr/bin 디렉토리에 있는 su 파일의 그룹 ID가 제어 시스템의 그룹 ID와 동일하지 않음을 나타냅니다. 이 정보는 테스트 시스템에 다른 버전의 소프트웨어가 설치되었음을 나타낼 수 있습니다. GID가 변경되었으므로 누군가 파일을 손상했을 가능성이 더 큽니다.

## ▼ 파일 속성을 지정하여 BART 보고서를 사용자 정의하는 방법

이 절차는 기존 매니페스트의 출력 결과를 특정 파일 속성으로 필터링하는 데 유용합니다.

**시작하기 전에** root 역할을 맡아야 합니다. 자세한 내용은 “Oracle Solaris 11.2의 사용자 및 프로세스 보안”의 “지정된 관리 권한 사용”을 참조하십시오.

1. **검사할 파일 속성을 확인합니다.**

2. **확인할 파일 속성이 포함된 두 매니페스트를 비교합니다.**
   예를 들면 다음과 같습니다.

   ```
   # bart compare -i lnmtime,mtime control-manifest.121513 \
     test-manifest.010514 > bart.report.010514
   ```

   각 파일 속성을 구분하려면 명령줄 구문에 콤마를 사용합니다.

3. **BART 보고서에서 이상한 점을 검토합니다.**

## ▼ 규칙 파일을 사용하여 BART 보고서를 사용자 정의하는 방법

규칙 파일을 사용하여 원하는 특정 파일 및 파일 속성에 대해 BART 매니페스트를 사용자 정의할 수 있습니다. 기본 BART 매니페스트에서 다른 규칙 파일을 사용하여 동일 매니페스트에 대해 다른 비교를 실행할 수 있습니다.

**시작하기 전에** `root` 역할을 맡아야 합니다. 자세한 내용은 "Oracle Solaris 11.2의 사용자 및 프로세스 보안"의 "지정된 관리 권한 사용"을 참조하십시오.

1. **모니터하려고 하는 파일 및 파일 속성을 결정합니다.**
2. **적합한 지시어를 사용하여 규칙 파일을 만듭니다.**
3. **만든 규칙 파일로 제어 매니페스트를 만듭니다.**

   ```
   # bart create -r myrules1-file > control-manifest
   ```

4. **(옵션) 이후에 사용할 수 있도록 매니페스트를 보호되는 디렉토리에 저장합니다.**
   예는 제어 매니페스트를 만드는 방법 [28]의 3단계을 참조하십시오.
5. **나중에 다른 시스템에서 또는 둘 다에서 동일한 매니페스트를 만듭니다.**

   ```
   # bart create -r myrules1-file > test-manifest
   ```

6. **동일한 규칙 파일을 사용하여 매니페스트를 비교합니다.**

   ```
   # bart compare -r myrules1-file control-manifest test-manifest > bart.report
   ```

7. **BART 보고서에서 이상한 점을 검토합니다.**

**예 2-4** 규칙 파일을 사용하여 BART 매니페스트 및 비교 보고서 사용자 정의

다음 규칙 파일은 `/usr/bin` 디렉토리에서 파일의 모든 속성을 나열하도록 `bart create` 명령을 지정합니다. 또한 규칙 파일은 `bart compare` 명령이 동일 디렉토리에서 크기 및 내용 변경 사항만 보고하도록 지시합니다.

```
# Check size and content changes in the /usr/bin directory.
# This rules file only checks size and content changes.
# See rules file example.

IGNORE all
CHECK size contents
/usr/bin
```

- 만든 규칙 파일로 제어 매니페스트를 만듭니다.

  ```
  # bart create -r usrbinrules.txt > usr_bin.control-manifest.121013
  ```

- `/usr/bin` 디렉토리에 대한 변경 사항을 모니터하려고 할 때마다 동일한 매니페스트를 준비합니다.

  ```
  # bart create -r usrbinrules.txt > usr_bin.test-manifest.121113
  ```

- 동일한 규칙 파일을 사용하여 매니페스트를 비교합니다.

  ```
  # bart compare -r usrbinrules.txt usr_bin.control-manifest.121013 \
  usr_bin.test-manifest.121113
  ```

- `bart compare` 명령의 출력을 조사합니다.

  ```
   /usr/bin/gunzip:  add
  /usr/bin/ypcat:
  delete
  ```

앞의 출력 결과는 `/usr/bin/ypcat` 파일이 삭제되고 `/usr/bin/gunzip` 파일이 추가되었음을 나타냅니다.

# BART 매니페스트, 규칙 파일 및 보고서

이 절에서는 BART에서 사용하고 만드는 파일의 형식을 설명합니다.

## BART 매니페스트 파일 형식

각 매니페스트 파일 항목은 파일 유형에 따라 단일 행입니다. 각 항목은 파일의 이름인 *fname*으로 시작합니다. 파일 이름에 포함된 특수 문자로 인한 구문 분석 문제를 방지하기 위해 파일 이름이 인코딩됩니다. 자세한 내용은 “BART 규칙 파일 형식” [38]을 참조하십시오.

이후 필드는 다음 파일 속성을 나타냅니다.

*type* 다음 값을 가질 수 있는 파일 유형입니다.

- B - 블록 장치 노드의 경우

- C - 문자 장치 노드의 경우
- D - 디렉토리의 경우
- F - 파일의 경우
- L - 심볼릭 링크의 경우
- P - 파이프의 경우
- S - 소켓의 경우

*size* 바이트 단위의 파일 크기입니다.

*mode* 파일의 권한을 나타내는 8진수 숫자입니다.

*acl* 파일에 대한 ACL 속성입니다. ACL 속성이 있는 파일의 경우 여기에는 `acltotext()`의 출력이 포함됩니다.

*uid* 이 항목 소유자의 숫자 사용자 ID입니다.

*gid* 이 항목 소유자의 숫자 그룹 ID입니다.

*dirmtime* 1970년 1월 1일 00:00:00 UTC 이후 디렉토리에 대한 마지막 수정 시간(초 단위)입니다.

*lnmtime* 1970년 1월 1일 00:00:00 UTC 이후 링크에 대한 마지막 수정 시간(초 단위)입니다.

*mtime* 1970년 1월 1일 00:00:00 UTC 이후 파일에 대한 마지막 수정 시간(초 단위)입니다.

*contents* 파일의 체크섬 값입니다. 이 속성은 일반 파일에 대해서만 지정됩니다. 컨텍스트 검사를 해제하거나 체크섬을 계산할 수 없는 경우 이 필드의 값은 -입니다.

*dest* 심볼릭 링크의 대상입니다.

*devnode* 장치 노드의 값입니다. 이 속성은 문자 장치 파일 및 블록 장치 파일 전용입니다.

자세한 내용은 `bart_manifest(4)` 매뉴얼 페이지를 참조하십시오.

# BART 규칙 파일 형식

규칙 파일은 매니페스트에 포함할 파일을 지정하고 매니페스트 또는 보고서에 포함할 파일 속성을 지정하는 행으로 구성된 텍스트 파일입니다. #으로 시작하는 행, 빈 행 및 공백이 포함된 행은 도구에서 무시됩니다.

입력 파일에는 세 가지 유형의 지시어가 있습니다.

- 하위 트리 지시어(선택적 패턴 일치 수정자 포함)
- CHECK 지시어
- IGNORE 지시어

**예 2-5** 규칙 파일 형식

```
<Global CHECK/IGNORE Directives>
<subtree1> [pattern1..]
<IGNORE/CHECK Directives for subtree1>

<subtree2> [pattern2..]
<subtree3> [pattern3..]
<subtree4> [pattern4..]
<IGNORE/CHECK Directives for subtree2, subtree3, subtree4>
```

**참고** - 모든 지시문은 순서대로 읽혀집니다. 이후 지시문은 이전 지시문을 대체할 수 있습니다.

하위 트리 지시어는 절대 경로 이름 다음에 0개 이상의 패턴 일치 명령문으로 시작되어야 합니다.

## BART 규칙 파일 속성

CHECK 및 IGNORE 명령문은 추적하거나 무시할 파일 속성을 정의합니다. 각 매니페스트를 시작하는 메타 데이터에는 파일 유형별로 속성 키워드가 나열됩니다. 예 2-1. “BART 매니페스트 형식에 대한 설명”을 참조하십시오.

all 키워드는 모든 파일 속성을 나타냅니다.

## BART 인용 구문

BART에서 사용하는 규칙 파일 사양 언어는 비표준 파일 이름을 나타내기 위한 표준 UNIX 인용 구문입니다. 내장된 탭, 공백, 개행 또는 특수 문자는 도구에서 파일 이름을 읽을 수 있도록 8진수 형식으로 인코딩됩니다. 이 비표준 인용 구문은 특정 파일 이름(예: 내장된 캐리지 리턴이 포함된 파일 이름)이 명령 파이프라인에서 올바르게 처리되지 않도록 합니다. 규칙 사양 언어에서는 셸 구문만을 사용하여 설명하기 어렵거나 충분하지 않은 복잡한 파일 이름 필터링 조건의 표현을 허용합니다.

자세한 내용은 bart_rules(4) 매뉴얼 페이지를 참조하십시오.

# BART 보고

기본 모드에서 BART 보고서는 수정된 디렉토리 시간 기록(dirmtime)을 제외하고 시스템에 설치된 모든 파일을 확인합니다.

```
CHECK all
IGNORE dirmtime
```

규칙 파일을 제공할 경우 CHECK all 및 IGNORE dirmtime의 전역 지시어가 이 순서대로 자동으로 규칙 파일 앞에 붙습니다.

## BART 출력

다음 종료 값이 반환됩니다.

| | |
|---|---|
| 0 | 성공 |
| 1 | 파일을 처리할 때 치명적이지 않은 오류(예: 권한 문제) |
| >1 | 치명적인 오류(예: 잘못된 명령줄 옵션) |

보고 방식은 두 가지 유형의 출력(상세 정보 출력과 프로그래밍 출력)을 제공합니다.

- 상세 정보 출력은 기본 출력이며 현지화되고 여러 행에 표시됩니다. 상세 정보 출력은 지역화되며 사람이 읽을 수 있습니다. bart compare 명령이 두 시스템 매니페스트를 비교하는 경우 파일 차이 목록이 생성됩니다.

  출력 결과의 구조는 다음과 같습니다.

  *filename attribute control:control-val test:test-val*

  | | |
  |---|---|
  | *filename* | 제어 매니페스트와 테스트 매니페스트 간에 서로 다른 파일 이름입니다. |
  | *attribute* | 비교되는 매니페스트 간에 서로 다른 파일 속성의 이름입니다. *control-val*은 *test-val*보다 우선합니다. 여러 속성에 대한 불일치가 같은 파일에서 발생하는 경우 각 차이가 별도의 행에 나타납니다. |

  다음은 /etc/passwd 파일의 속성 차이에 대한 예입니다. 출력은 size, mtime 및 contents 속성이 변경되었음을 나타냅니다.

  ```
  /etc/passwd:
  size control:74 test:81
  mtime control:3c165879 test:3c165979
  contents control:daca28ae0de97afd7a6b91fde8d57afa
  test:84b2b32c4165887355317207b48a6ec7
  ```

- 프로그래밍 출력은 `bart compare` 명령에 `-p` 옵션을 사용하여 생성됩니다. 이 출력 결과는 프로그래밍 방식의 조작에 적합합니다.

  출력 결과의 구조는 다음과 같습니다.

  *filename attribute control-val test-val* [*attribute control-val test-val*]*

  | | |
  |---|---|
  | *filename* | 기본 형식의 *filename* 속성과 같습니다. |
  | *attribute control-val test-val* | 각 파일에 대한 제어 매니페스트와 테스트 매니페스트 간에 서로 다른 파일 속성 설명입니다. |

`bart` 명령에서 지원하는 속성 목록은 "BART 규칙 파일 속성" [39]을 참조하십시오.

자세한 내용은 `bart(1M)` 매뉴얼 페이지를 참조하십시오.

# 보안 용어

**감사 정책** 어떤 감사 이벤트를 기록할지 결정하는 전역 사용자별 설정입니다. 감사 서비스에 적용되는 전역 설정은 일반적으로 감사 추적에 포함할 선택적 정보 조각에 영향을 미칩니다. 두 설정 `cnt` 및 `ahlt`는 감사 대기열을 채울 때 시스템의 작업에 영향을 미칩니다. 예를 들어, 감사 정책에서 시퀀스 번호가 모든 감사 레코드에 속하도록 요구할 수 있습니다.

**감사 추적** 모든 호스트의 모든 감사 파일 모음입니다.

**감사 파일** 이진 감사 로그. 감사 파일은 감사 파일 시스템에 별도로 저장됩니다.

**강화** 호스트에 내재된 보안 취약성을 제거하도록 운영 체제의 기본 구성을 수정한 것입니다.

**개인 키** 각 사용자 주체에 제공되며 주체의 사용자와 KDC에만 알려진 키입니다. 사용자 주체의 경우 키는 사용자 암호를 기반으로 합니다. 키를 참조하십시오.

**개인 키 암호화** 개인 키 암호화에서 발신자와 수신자는 암호화에 동일한 키를 사용합니다. 공개 키 암호화도 참조하십시오.

**갱신 가능 티켓** 장시간 존재하는 티켓은 보안 위험이 있으므로 티켓을 `renewable`로 지정할 수 있습니다. 갱신 가능 티켓에는 두 개의 만료 시간 a) 티켓의 현재 인스턴스가 만료되는 시간 b) 티켓의 최대 수명이 있습니다. 클라이언트가 티켓을 계속 사용하려면 첫번째 만료가 발생하기 전에 티켓을 갱신합니다. 예를 들어, 1시간 동안 유효한 티켓이 있고 모든 티켓은 최대 10시간의 수명을 가질 수 있습니다. 티켓을 보유하는 클라이언트가 1시간보다 더 오래 티켓을 보관하려면 티켓을 갱신해야 합니다. 티켓이 최대 티켓 수명에 도달하면 자동으로 만료되어 갱신할 수 없습니다.

**검사 엔진** 파일에 알려진 바이러스가 있는지 조사하는, 외부 호스트에 상주하는 타사 응용 프로그램입니다.

**공개 키 기술에 대한 정책** 키 관리 프레임워크(KMF)에서 정책은 인증서 사용을 관리합니다. KMF 정책 데이터베이스는 KMF 라이브러리에서 관리되는 키 및 인증서 사용을 제약할 수 있습니다.

**공개 키 암호화** 각 사용자가 두 개의 키, 즉 하나의 공개 키와 하나의 개인 키를 사용하는 암호화 체계입니다. 공개 키 암호화에서 발신자는 수신자의 공개 키를 사용하여 메시지를 암호화하고, 수신자는 개인 키를 사용하여 암호를 해독합니다. Kerberos 서비스는 개인 키 시스템입니다. 개인 키 암호화도 참조하십시오.

**공급자** Oracle Solaris의 암호화 프레임워크 기능에서 소비자에게 제공된 암호화 서비스입니다. 공급자의 예로 PKCS #11 라이브러리, 커널 암호화 모듈, 하드웨어 가속기가 있습니다. 공급

자는 암호화 프레임워크에 플러그인되므로 플러그인이라고도 합니다. 소비자의 예는 소비자를 참조하십시오.

**공용 객체** `root` 사용자가 소유하고 어디서든 읽을 수 있는 파일입니다(예: `/etc` 디렉토리의 파일).

**관계** `kdc.conf` 또는 `krb5.conf` 파일에 정의된 구성 변수 또는 관계입니다.

**관리자 주체** *username*`/admin`(예: `jdoe/admin`) 형식의 이름을 가진 사용자 주체입니다. 관리자 주체는 일반 사용자 주체보다 더 많은 권한(예: 정책 변경)을 가질 수 있습니다. 주체 이름, 사용자 주체도 참조하십시오.

**권한** 1. 일반적으로 컴퓨터 시스템에서 일반 사용자의 권한을 벗어난 작업을 수행할 수 있는 권한 또는 기능입니다. 수퍼 유저 권한은 수퍼 유저에게 부여된 모든 rights(권한)입니다. 권한 있는 사용자 또는 권한 있는 응용 프로그램은 추가 권한이 부여된 사용자나 응용 프로그램입니다.

2. Oracle Solaris 시스템에서 프로세스에 대한 별개의 권한입니다. 권한은 `root`인 프로세스를 좀 더 세부적으로 제어합니다. 권한은 커널에서 정의되고 시행됩니다. 권한을 프로세스 권한 또는 커널 권한이라고도 합니다. 권한에 대한 자세한 설명은 `privileges`(5) 매뉴얼 페이지를 참조하십시오.

**권한 모델** 수퍼 유저 모델보다 더 엄격한 컴퓨터 시스템의 보안 모델입니다. 권한 모델에서 프로세스를 실행하려면 권한이 필요합니다. 시스템 운영은 관리자가 해당 프로세스에 보유한 권한으로 기반으로 별개의 부분으로 나눌 수 있습니다. 관리자의 로그인 프로세스에 권한을 지정할 수 있습니다. 또는 특정 명령에만 효력을 발휘하도록 권한을 지정할 수 있습니다.

**권한 부여** 1. Kerberos에서는 주체가 서비스를 사용할 수 있는지, 어떤 객체에 주체가 액세스할 수 있는지, 각 객체에 대해 허용된 액세스 유형 등을 결정하는 프로세스입니다.

2. 사용자 권한 관리에서는 다른 상황에서 보안 정책에 의해 금지되는 종류의 작업을 수행하기 위해 역할 또는 사용자에 지정하거나 권한 프로파일에 포함할 수 있는 권한입니다. 권한 부여는 커널이 아니라 사용자 응용 프로그램 레벨에서 적용됩니다.

**권한 세트** 권한 모음입니다. 각 프로세스에는 프로세스가 특정 권한을 사용할 수 있는지 여부를 결정하는 4개의 권한 세트가 있습니다. 제한 세트, 유효 세트, 허가된 세트, 상속 가능한 세트를 참조하십시오.

또한 권한의 기본 세트는 로그인 시 사용자의 프로세스에 지정된 권한 모음입니다.

**권한 에스컬레이션** 기본값을 대체하는 권한을 포함하여 지정된 권한이 허용하는 리소스 범위 밖에 있는 리소스에 대한 액세스를 얻는 것입니다. 그 결과, 프로세스에서 허용되지 않은 작업을 수행할 수 있습니다.

**권한 인식** 코드를 통해 권한 사용을 켜고 끄는 프로그램, 스크립트, 명령입니다. 운용 환경에서 켜져 있는 권한을 프로세스에 제공해야 합니다. 프로그램의 사용자가 권한을 프로그램에 추가한 권한 프로파일을 사용하도록 하면 됩니다. 권한에 대한 자세한 설명은 `privileges`(5) 매뉴얼 페이지를 참조하십시오.

**권한 있는 응용 프로그램** 시스템 컨트롤을 대체할 수 있는 응용 프로그램입니다. 응용 프로그램이 특정 UID, GID, 권한 부여 또는 권한과 같은 보안 속성을 검사합니다.

**권한 프로파일** 프로파일이라고도 합니다. 역할 또는 사용자에 지정할 수 있는 보안 대체 모음입니다. 권한(right) 프로파일에는 권한 부여, 권한(privilege), 보안 속성 포함 명령 및 보충 프로파일이라고 하는 기타 권한(right) 프로파일이 포함될 수 있습니다.

**기밀성** 프라이버시를 참조하십시오.

**기본** 주체 이름의 첫번째 부분입니다. 인스턴스, 주체 이름, 영역도 참조하십시오.

**기본 세트** 로그인 시 사용자 프로세스에 지정되는 권한 세트입니다. 수정되지 않은 시스템에서 각 사용자의 초기 상속 가능한 세트는 로그인 시 기본 세트와 같습니다.

**네트워크 애플리케이션 서버** `ftp`와 같은 네트워크 응용 프로그램을 제공하는 서버입니다. 영역에 여러 네트워크 애플리케이션 서버를 포함할 수 있습니다.

**네트워크 정책** 네트워크 트래픽을 보호하기 위해 네트워크 유틸리티가 구성하는 설정입니다. 네트워크 보안에 대한 자세한 내용은 "Oracle Solaris 11.2의 네트워크 보안 "을 참조하십시오.

**다이제스트** 메시지 다이제스트를 참조하십시오.

**단일 시스템 이미지** 단일 시스템 이미지는 Oracle Solaris 감사에 사용되어 동일한 이름 지정 서비스를 사용하는 감사 시스템 그룹을 설명합니다. 이러한 시스템은 해당 감사 레코드를 중앙 감사 서버로 보내고, 여기서 레코드가 한 시스템에서 나온 것처럼 레코드를 비교할 수 있습니다.

**동기 감사 이벤트** 감사 이벤트의 다수를 차지합니다. 이러한 이벤트는 시스템의 프로세스와 연관됩니다. 프로세스와 연관된 출처를 알 수 없는 이벤트는 실패한 로그인과 같은 동기 이벤트입니다.

**마스터 KDC** 각 영역의 주 KDC로, Kerberos 관리 서버인 `kadmind`와 인증 및 티켓 부여 데몬인 `krb5kdc`를 포함합니다. 각 영역에는 적어도 하나의 마스터 KDC가 있어야 하고, 클라이언트에 인증 서비스를 제공하는 많은 중복된 슬레이브 KDC를 포함할 수 있습니다.

**메시지 다이제스트** 메시지 다이제스트는 메시지에서 계산된 해시 값입니다. 해시 값은 메시지를 거의 고유하게 식별합니다. 다이제스트는 파일 무결성 확인에 유용합니다.

**무결성** 사용자 인증과 더불어, 암호화 체크섬을 통해 전송된 데이터의 유효성을 제공하는 보안 서비스입니다. 인증, 프라이버시도 참조하십시오.

**문장암호** 개인 키를 문장암호 사용자가 만들었는지 확인하는 데 사용되는 문구입니다. 좋은 문장암호는 10-30자 길이로, 영문자와 숫자를 섞어서 만들고 단순한 문구와 단순한 이름을 피합니다. 통신을 암호화 및 해독하기 위해 개인 키 사용을 인증하려면 문장암호를 묻는 메시지가 나타납니다.

**방식** 1. 데이터 인증 또는 기밀성을 이루기 위한 암호화 기법을 지정하는 소프트웨어 패키지입니다. 예: Kerberos V5, Diffie-Hellman 공개 키.

2. Oracle Solaris의 암호화 프레임워크 기능에서 특정 목적을 위한 알고리즘의 구현입니다. 예를 들어, 인증에 적용된 DES 방식(예: CKM_DES_MAC)은 암호화에 적용된 DES 방식(예: CKM_DES_CBC_PAD)과 별도의 방식입니다.

**보안 방식** 방식을 참조하십시오.

**보안 서비스** 서비스를 참조하십시오.

**보안 속성** 수퍼 유저가 아닌 사용자가 관리 명령을 실행할 때 이 명령이 성공하게 해주는 보안 정책의 대체입니다. 수퍼 유저 모델에서는 `setuid root` 및 `setgid` 프로그램이 보안 속성입니다. 이러한 속성을 명령에 적용할 때 누가 명령을 실행하든지 관계없이 명령을 성공합니다. 권한 모델에서는 커널 권한 및 기타 rights(권한)이 `setuid root` 프로그램을 보안 속성으로 대체합니다. 권한 모델은 `setuid` 및 `setgid` 프로그램을 보안 속성으로 인식하므로 수퍼 유저 모델과 호환됩니다.

**보안 정책** 정책을 참조하십시오.

**보안 종류** 종류를 참조하십시오.

**보안 키** 개인 키를 참조하십시오.

**비동기 감사 이벤트** 비동기 이벤트는 시스템 이벤트의 소수를 차지합니다. 이러한 이벤트는 어떤 프로세스와 연관되지 않으므로 프로세스를 차단했다가 나중에 깨울 수 없습니다. 비동기 이벤트의 예로, 초기 시스템 부트 및 PROM 진입/종료 이벤트가 있습니다.

**사용자 주체** 특정 사용자에 기인한 주체입니다. 사용자 주체의 기본 이름은 사용자 이름이고, 선택적 인스턴스는 의도한 해당 자격 증명 용도를 설명하는 데 사용되는 이름입니다(예: `jdoe` 또는 `jdoe/admin`). 사용자 인스턴스라고도 합니다. 서비스 주체도 참조하십시오.

**상속 가능한 세트** `exec`의 호출에서 프로세스가 상속할 수 있는 권한 세트입니다.

**서버** 네트워크 클라이언트에 리소스를 제공하는 주체입니다. 예를 들어, `central.example.com` 시스템에 `ssh`를 제공하면 해당 시스템은 `ssh` 서비스를 제공하는 서버입니다. 서비스 주체도 참조하십시오.

**서버 주체** (RPCSEC_GSS API) 서비스를 제공하는 주체입니다. 서버 주체는 *service@host* 형식으로 ASCII 문자열로 저장됩니다. 클라이언트 주체도 참조하십시오.

**서비스** 1. 종종 여러 대의 서버에 의해 네트워크 클라이언트에 제공된 리소스입니다. 예를 들어, `central.example.com` 시스템에 `rlogin`을 제공하는 경우 해당 시스템은 `rlogin` 서비스를 제공하는 서버입니다.

2. 인증 외의 보호 레벨을 제공하는 보안 서비스(무결성 또는 프라이버시)입니다. 무결성 및 프라이버시를 참조하십시오.

**서비스 주체** 서비스에 Kerberos 인증을 제공하는 주체입니다. 서비스 주체의 경우 기본 이름은 `ftp`와 같은 서비스 이름이고, 해당 인스턴스는 서비스를 제공하는 시스템의 정규화된 호스트 이름입니다. 호스트 주체, 사용자 주체도 참조하십시오.

**서비스 키** 서비스 주체 및 KDC에서 공유되고 시스템 한도 밖에서 배포되는 암호화 키입니다. 키를 참조하십시오.

**세션 키** 인증 서비스 또는 TGS(티켓 부여 서비스)에서 생성된 키입니다. 세션 키는 클라이언트와 서비스 간에 보안 트랜잭션을 제공하기 위해 생성됩니다. 세션 키의 수명은 단일 로그인 세션으로 제한됩니다. 키를 참조하십시오.

**소비자** Oracle Solaris의 암호화 프레임워크 기능에서 소비자는 공급자로부터 전달된 암호화 서비스의 사용자입니다. 소비자는 응용 프로그램, 최종 사용자 또는 커널 작업일 수 있습니다. 소비자의 예로 Kerberos, IKE, IPsec 등이 있습니다. 공급자의 예는 공급자를 참조하십시오.

**소프트웨어 공급자** Oracle Solaris의 암호화 프레임워크 기능에서 암호화 서비스를 제공하는 커널 소프트웨어 모듈 또는 PKCS #11 라이브러리입니다. 공급자를 참조하십시오.

**수퍼 유저 모델** 컴퓨터 시스템의 전형적인 UNIX 보안 모델입니다. 수퍼 유저 모델에서 관리자는 all-or-nothing 방식으로 시스템을 제어합니다. 일반적으로 시스템을 관리하려는 경우 사용자는 수퍼 유저(`root`)로 로그인하여 모든 관리 작업을 수행할 수 있습니다.

**슬레이브 KDC** 마스터 KDC의 복사본으로, 대부분의 마스터 기능을 수행할 수 있습니다. 각 영역에는 대개 여러 개의 슬레이브 KDC(마스터 KDC는 하나만)가 있습니다. KDC, 마스터 KDC도 참조하십시오.

**시드** 무작위 수를 생성하기 위한 숫자 스타터입니다. 스타터가 무작위 소스에서 기원할 때 시드를 무작위 시드라고 합니다.

**알고리즘** 암호화 알고리즘입니다. 입력을 암호화하거나 해시하는 확립된 순환적 계산 프로시저입니다.

**암호 정책** 암호를 생성하는 데 사용할 수 있는 암호화 알고리즘입니다. 암호 변경 주기, 암호 시도 허용 회수, 기타 보안 고려 사항 등 암호와 관련한 일반적인 사안이라고 할 수 있습니다. 보안 정책에 암호가 필요합니다. 암호 정책에서는 암호를 AES 알고리즘으로 암호화해야 하고, 추가로 암호 강도와 관련된 요구 사항이 필요할 수 있습니다.

**암호화 알고리즘** 알고리즘을 참조하십시오.

**암호화 프레임워크의 정책** Oracle Solaris의 암호화 프레임워크 기능에서 정책은 기존 암호화 방식을 사용 안함으로 설정합니다. 그러면 방식을 사용할 수 없습니다. 암호화 프레임워크의 정책은 DES와 같은 공급자가 `CKM_DES_CBC`와 같은 특정 방식을 사용하는 것을 금지할 수 있습니다.

**애플리케이션 서버** 네트워크 애플리케이션 서버를 참조하십시오.

**액세스 제어 목록(ACL)** 액세스 제어 목록(ACL)은 전통적인 UNIX 파일 보호보다 좀 더 세부적인 파일 보안을 제공합니다. 예를 들어, ACL을 사용하여 파일에 그룹 읽기 액세스를 허용하면서 해당 그룹의 한 멤버만 파일 쓰기를 허용할 수 있습니다.

**역할** 지정된 사용자만 맡을 수 있는 권한 있는 응용 프로그램을 실행하기 위한 특수한 신원입니다.

**영역** 1. 단일 Kerberos 데이터베이스와 일련의 KDC(키 배포 센터)에 의해 제공된 논리적 네트워크입니다.

2. 주체 이름의 세번째 부분입니다. 주체 이름 jdoe/admin@CORP.EXAMPLE.COM의 경우 영역은 CORP.EXAMPLE.COM입니다. 주체 이름도 참조하십시오.

**유효 세트** 현재 프로세스에 발효 중인 권한 세트입니다.

**이름 서비스 범위** 역할이 작동하도록 허가된 범위입니다. 즉, NIS LDAP와 같은 지정된 이름 지정 서비스에서 제공하는 개별 호스트 또는 모든 호스트를 말합니다.

**인스턴스** 주체 이름의 두번째 부분으로, 인스턴스는 주체의 기본 부분을 한정합니다. 서비스 주체의 경우 인스턴스는 필수 사항입니다. 인스턴스는 host/central.example.com과 같이 호스트의 정규화된 도메인 이름입니다. 사용자 주체의 경우 인스턴스는 선택 사항입니다. 그러나 jdoe 및 jdoe/admin은 고유한 주체입니다. 기본, 주체 이름, 서비스 주체, 사용자 주체도 참조하십시오.

**인증** 주체의 제시된 신원을 확인하는 프로세스입니다.

**인증자** 인증자는 티켓(KDC에서) 및 서비스(서버에서)를 요청할 때 클라이언트에 의해 전달됩니다. 최근 시점에서 확인할 수 있는 클라이언트 및 서버에만 알려진 세션 키를 사용하여 생성된 정보를 포함하므로 트랜잭션이 안전한 것으로 나타납니다. 티켓과 함께 사용할 경우 인증자는 사용자 주체를 인증할 수 있습니다. 인증자에는 사용자의 주체 이름, 사용자 호스트의 IP 주소, 시간 기록이 포함됩니다. 티켓과 달리, 인증자는 대개 서비스 액세스를 요청할 때 한번만 사용할 수 있습니다. 인증자는 클라이언트 및 서버에 대한 세션 키를 사용하여 암호화됩니다.

**자격 증명** 티켓 및 일치하는 세션 키를 포함하는 정보 패키지입니다. 주체의 신원을 인증하는 데 사용됩니다. 티켓, 세션 키도 참조하십시오.

**자격 증명 캐시** KDC로부터 받은 자격 증명을 포함하는 저장 공간(대개 파일)입니다.

**잘못된 티켓** 아직 사용할 수 없는 후일자 티켓입니다. 잘못된 티켓은 유효해질 때까지 애플리케이션 서버에서 거부합니다. 유효화하려면 시작 시간이 지난 후에 VALIDATE 플래그 세트를 사용하여 TGS 요청의 클라이언트가 KDC에 잘못된 티켓을 제시해야 합니다. 후일자 티켓도 참조하십시오.

**장치 정책** 커널 레벨의 장치 보호입니다. 장치 정책은 장치에 두 개의 권한 세트로 구현됩니다. 첫번째 권한 세트는 장치의 읽기 액세스를 제어합니다. 두번째 권한 세트는 장치의 쓰기 액세스를 제어합니다. 정책을 참조하십시오.

**장치 할당** 사용자 레벨의 장치 보호입니다. 장치 할당은 하나의 장치를 한번에 한 사용자만 배타적으로 사용하도록 합니다. 장치 재사용 전에 장치 데이터를 비웁니다. 권한 부여를 사용하여 장치 할당이 허가된 사용자를 제한할 수 있습니다.

**전달 가능 티켓** 클라이언트가 원격 호스트에서 티켓을 요청하기 위해 전체 인증 프로세스를 거치지 않고도 사용할 수 있는 티켓입니다. 예를 들어, 사용자 jennifer의 시스템에 있는 동안 사용자 david가 전달 가능 티켓을 얻은 경우 david는 새 티켓을 얻지 않고도(다시 인증받을 필요 없이) 자신의 시스템에 로그인할 수 있습니다. 프록시 가능 티켓도 참조하십시오.

**정책** 일반적으로, 의사결정 및 조치를 반영하거나 결정하는 계획이나 행동 방침입니다. 컴퓨터 시스템의 경우 정책은 대개 보안 정책을 의미합니다. 사이트의 보안 정책은 처리 중인 정보의

민감도를 정의하는 규칙 세트이자, 허용되지 않은 액세스로부터 정보를 보호하는 데 사용되는 측정치입니다. 예를 들어, 시스템을 감사하고 장치를 사용할 수 있도록 할당하고 암호를 6주마다 변경하도록 보안 정책을 수립할 수 있습니다.

Oracle Solaris OS의 특정 영역에서 정책을 구현하는 방법은 감사 정책, 암호화 프레임워크의 정책, 장치 정책, Kerberos 정책, 암호 정책 및 rights policy(권한 정책)을 참조하십시오.

**제한 세트** 프로세스와 그 자식에 사용 가능한 권한에 대한 외부 제한입니다.

**종류** 전통적으로 보안 종류와 인증 종류는 인증 유형(AUTH_UNIX, AUTH_DES, AUTH_KERB)을 지칭한 종류로서, 동일한 의미입니다. RPCSEC_GSS는 인증과 더불어 무결성과 프라이버시 서비스를 제공하지만 역시 보안 종류입니다.

**주체** 1. 네트워크 통신에 참여하는 고유한 이름이 지정된 클라이언트/사용자 또는 서버/서비스입니다. Kerberos 트랜잭션에는 주체들(서비스 주체 및 사용자 주체) 간의 상호 작용 또는 주체와 KDC 간의 상호 작용이 관여합니다. 다시 말해서, 주체는 Kerberos가 티켓을 지정할 수 있는 고유한 개체입니다. 주체 이름, 서비스 주체, 사용자 주체도 참조하십시오.

2. (RPCSEC_GSS API) 클라이언트 주체, 서버 주체를 참조하십시오.

**주체 이름** 1. *primary/instance@REALM* 형식의 주체 이름입니다. 인스턴스, 기본, 영역도 참조하십시오.

2. (RPCSEC_GSS API) 클라이언트 주체, 서버 주체를 참조하십시오.

**책임 구분** 최소한의 특권 개념의 일부입니다. 책임을 구분하면 한 사용자가 트랜잭션을 완성하는 모든 작업을 수행하거나 승인할 수 없게 됩니다. 예를 들어, RBAC에서 보안 대체 지정으로부터 로그인 사용자 생성을 분리할 수 있습니다. 한 역할이 사용자를 만듭니다. 별도의 역할이 권한 프로파일, 역할, 기존 사용자의 권한과 같은 보안 속성을 지정할 수 있습니다.

**초기 티켓** (기존 TGT(티켓 부여 티켓)에 기반하지 않고) 직접 발행된 티켓입니다. 암호를 변경하는 응용 프로그램과 같은 일부 서비스는 `initial`로 표시된 티켓이 필요할 수 있으므로 클라이언트가 보안 키를 알고 있다는 것을 스스로 보증해야 합니다. 초기 티켓은 클라이언트가 (오랫동안 존재해 왔던 TGT(티켓 부여 티켓)에 의존하는 대신) 최근에 자체 인증되었음을 나타내므로 이 보증은 매우 중요합니다.

**최소 권한의 원칙** 최소한의 특권을 참조하십시오.

**최소한의 특권** 지정된 프로세스를 일부 수퍼 유저 권한에만 제공하는 보안 모델입니다. 최소 권한 모델은 일반 사용자가 파일 시스템 마운트 및 파일 소유권 변경과 같은 개인적인 관리 작업을 수행할 수 있도록 충분한 권한을 지정합니다. 반면에 프로세스는 완전한 수퍼 유저 권한(즉 모든 권한)이 아닌, 작업 완료에 필요한 권한으로만 실행됩니다. 버퍼 오버플로우 같은 프로그래밍 오류로 인한 손해는, 보호된 시스템 파일 읽기/쓰기 또는 시스템 정지 같은 중요한 능력에 액세스할 수 없는 비루트 사용자에 국한될 수 있습니다.

**최소화** 서버 실행에 필요한 최소 운영 체제를 설치합니다. 서버 작동에 직접적인 관련이 없는 소프트웨어는 설치되지 않거나 설치 후 삭제됩니다.

**출처를 알 수 없는 감사 이벤트** AUE_BOOT 이벤트와 같이 개시자가 결정할 수 없는 감사 이벤트입니다.

**클라이언트** 좁은 의미로, 사용자 대신 네트워크 서비스를 이용하는 프로세스입니다. 예를 들어, `rlogin`을 사용하는 응용 프로그램이 있습니다. 어떤 경우 서버 자체가 다른 서버나 서비스의 클라이언트가 될 수 있습니다.

더 넓은 의미로, a) Kerberos 자격 증명을 수신하고 b) 서버에서 제공한 서비스를 이용하는 호스트입니다.

간단히 말하면, 서비스를 이용하는 주체입니다.

**클라이언트 주체** (RPCSEC_GSS API) RPCSEC_GSS로 보안된 네트워크 서비스를 사용하는 클라이언트(사용자 또는 응용 프로그램)입니다. 클라이언트 주체 이름은 `rpc_gss_principal_t` 구조 형태로 저장됩니다.

**클럭 불균형** Kerberos 인증 시스템에 참여하는 모든 호스트의 내부 시스템 클럭에 차이가 날 수 있는 최대 시간입니다. 참여하는 호스트 사이에 클럭 불균형을 초과할 경우 요청이 거부됩니다. 클럭 불균형은 `krb5.conf` 파일에 지정할 수 있습니다.

**키** 1. 일반적으로, 두 가지의 주요 키 유형 중 하나입니다.

- 대칭 키 - 암호 해독 키와 똑같은 암호화 키입니다. 대칭 키는 파일을 암호화하는 데 사용됩니다.
- 비대칭 키 또는 공개 키 - Diffie-Hellman 또는 RSA와 같은 공개 키 알고리즘에 사용되는 키입니다. 공개 키에는 한 사용자에만 알려진 개인 키, 서버나 일반 리소스에서 사용되는 공개 키, 그리고 둘을 조합한 개인-공개 키 쌍이 포함됩니다. 개인 키는 보안 키라고도 합니다. 공개 키는 공유 키 또는 공통 키라고도 합니다.

2. keytab 파일의 항목(주체 이름)입니다. keytab 파일도 참조하십시오.

3. Kerberos에서 암호화 키로 사용되며 다음 세 가지 유형이 있습니다.

- 개인 키 - 주체 및 KDC에서 공유되고 시스템 한도 밖에서 배포되는 암호화 키입니다. 개인 키도 참조하십시오.
- 서비스 키 - 이 키는 개인 키와 동일한 목적을 제공하지만, 서버 및 서비스에서 사용됩니다. 서비스 키도 참조하십시오.
- 세션 키 - 단일 로그인 세션 기간으로 제한된 수명 동안 두 주체 간에 사용되는 임시 암호화 키입니다. 세션 키도 참조하십시오.

**키 저장소** 키 저장소는 응용 프로그램별로 검색하기 위한 암호, 문장암호, 인증서 및 기타 인증 객체를 저장합니다. 키 저장소는 일부 응용 프로그램이 사용하는 기술 또는 위치에 따라 달라질 수 있습니다.

**티켓** 사용자의 신원을 서버나 서비스로 안전하게 전달하는 데 사용되는 정보 패킷입니다. 티켓은 단일 클라이언트에만, 그리고 특정 서버의 특정 서비스에만 유효합니다. 티켓에는 서비스의 주체 이름, 사용자의 주체 이름, 사용자 호스트의 IP 주소, 시간 기록, 티켓의 수명을 정의하는 값이 포함됩니다. 클라이언트 및 서비스에서 사용할 무작위 세션 키로 티켓이 생성됩니

다. 일단 티켓이 만들어지면 만료될 때까지 재사용할 수 있습니다. 티켓은 새로운 인증자와 함께 제시될 때 클라이언트 인증에만 사용됩니다. 인증자, 자격 증명, 서비스, 세션 키를 참조하십시오.

**티켓 파일** 자격 증명 캐시를 참조하십시오.

**프라이버시** 전송된 데이터를 보내기 전에 암호화하는 보안 서비스입니다. 프라이버시에는 데이터 무결성과 사용자 인증도 포함됩니다. 인증, 무결성, 서비스를 참조하십시오.

**프로파일 셸** 권한 관리에서는 역할 또는 사용자가 역할의 권한 프로파일에 지정된 권한 있는 응용 프로그램을 명령줄에서 실행할 수 있게 하는 셸입니다. 프로파일 셸 버전은 시스템에서 사용 가능한 셸에 해당합니다(예: `bash`의 `pfbash` 버전).

**프록시 가능 티켓** 클라이언트에 작업을 수행하는 대신, 서비스에서 사용할 수 있는 티켓입니다. 따라서 서비스가 클라이언트의 프록시 역할을 한다고 말할 수 있습니다. 티켓을 사용하여 서비스는 클라이언트의 신원을 차용할 수 있습니다. 프록시 가능 티켓을 사용하여 다른 서비스에 대한 서비스 티켓을 얻을 수 있지만, TGT(티켓 부여 티켓)는 얻을 수 없습니다. 프록시 가능 티켓과 전달 가능 티켓의 차이점은, 프록시 가능 티켓은 단일 작업에만 유효하다는 것입니다. 전달 가능 티켓도 참조하십시오.

**하드웨어 공급자** Oracle Solaris의 암호화 프레임워크 기능에서 장치 드라이버 및 해당 하드웨어 가속기입니다. 하드웨어 공급자는 컴퓨터 시스템에서 값비싼 암호화 작업 부담을 덜어주므로 CPU 리소스를 확보하여 다른 용도로 사용할 수 있습니다. 공급자를 참조하십시오.

**허가된 세트** 프로세스에서 사용할 수 있는 권한 세트입니다.

**호스트** 네트워크를 통해 액세스 가능한 시스템입니다.

**호스트 주체** `ftp`, `rcp` 또는 `rlogin`과 같은 다양한 네트워크 서비스를 제공하기 위해 주체(기본 이름 `host`로 서명됨)가 설정되는 특정 인스턴스의 서비스 주체입니다. 호스트 주체의 예는 `host/central.example.com@EXAMPLE.COM`입니다. 서버 주체도 참조하십시오.

**후일자 티켓** 후일자 티켓은 생성 후 지정된 시간까지 유효해지지 않습니다. 예를 들어, 이러한 티켓은 밤늦게 실행하려는 일괄 처리 작업에 유용합니다. 티켓을 훔친 경우 일괄 처리 작업이 실행될 때까지 티켓을 사용할 수 없기 때문입니다. 후일자 티켓을 발행할 때 `invalid`로 발행되고 a) 시작 시간이 지날 때까지 b) 클라이언트가 KDC에서 검증으로 요청할 때까지 해당 방법을 유지합니다. 후일자 티켓은 보통 TGT(티켓 부여 티켓)의 만료 시간까지 유효합니다. 그러나 후일자 티켓이 `renewable`로 표시된 경우 티켓의 수명이 보통 TGT(티켓 부여 티켓)의 전체 수명 기간과 똑같이 설정됩니다. 잘못된 티켓, 갱신 가능 티켓도 참조하십시오.

**AES** Advanced Encryption Standard의 머리글자어로, 고급 암호화 표준입니다. 대칭 128비트 블록 데이터 암호화 기술입니다. 미국 정부는 2000년 10월 알고리즘의 Rijndael 변형을 암호화 표준으로 채택했습니다. AES가 정부 표준으로 사용자 주체 암호화를 대체합니다.

**authenticated rights profile(인증된 권한 프로파일)** 지정된 사용자나 역할이 프로파일에서 작업을 실행하기 전에 암호를 입력하도록 요구하는 권한 프로파일입니다. 이 동작은 `sudo` 동작과 비슷합니다. 암호가 유효한 기간은 구성 가능합니다.

**Blowfish** 32-448비트의 가변 길이 키를 사용하는 대칭 블록 암호화 알고리즘입니다. 저작자인 Bruce Schneier에 따르면, Blowfish는 키를 자주 바꾸지 않는 응용 프로그램에 최적화되어 있습니다.

**DES** Data Encryption Standard의 머리글자어로, 데이터 암호화 표준입니다. 1975년에 개발되고 1981년에 ANSI에 의해 ANSI X.3.92로 표준화된 대칭 키 암호화 방법입니다. DES에서는 56비트 키를 사용합니다.

**Diffie-Hellman 프로토콜** 공개 키 암호화라고도 합니다. 1976년 Diffie와 Hellman이 개발한 비대칭 암호화 키 계약 프로토콜입니다. 이 프로토콜을 사용하면 두 사용자가 사전 보안 없이 비보안 매체를 통해 보안 키를 교환할 수 있습니다. Diffie-Hellman은 Kerberos에서 사용됩니다.

**DSA** Digital Signature Algorithm의 머리글자어로, 디지털 서명 알고리즘입니다. 512-4096비트의 가변 키 크기를 사용하는 공개 키 알고리즘입니다. 미국 정부 표준인 DSS는 1024비트까지 지원합니다. DSA는 입력에 SHA1을 사용합니다.

**ECDSA** Elliptic Curve Digital Signature Algorithm의 머리글자어로, 타원 곡선 디지털 서명 알고리즘입니다. 타원 곡선 수학을 기반으로 하는 공개 키 알고리즘입니다. ECDSA 키 크기는 동일한 길이의 서명을 생성하는 데 필요한 DSA 공개 키의 크기보다 많이 작습니다.

**FQDN** 정규화된 도메인 이름입니다. 간단한 `denver`와 대조되는 `central.example.com`을 예로 들 수 있습니다.

**GSS-API** Generic Security Service Application Programming Interface의 약자. Kerberos 서비스를 포함하여 다양한 모듈형 보안 서비스를 지원하는 네트워크 계층입니다. GSS-API는 보안 인증, 무결성, 프라이버시 서비스를 제공합니다. 인증, 무결성, 프라이버시를 참조하십시오.

**KDC** Key Distribution Center의 머리글자어로, 키 배포 센터입니다. 세 가지 Kerberos V5 구성 요소가 있는 시스템입니다.

- 주체 및 키 데이터베이스
- 인증 서비스
- TGS(티켓 부여 서비스)

각 영역에는 마스터 KDC가 있고 하나 이상의 슬레이브 KDC가 있어야 합니다.

**Kerberos** 인증 서비스, 서비스에서 사용되는 프로토콜 또는 서비스 구현에 사용되는 코드입니다.

Kerberos V5 구현에 가장 근접한 Oracle Solaris의 Kerberos 구현입니다.

"Kerberos"와 "Kerberos V5"는 기술적으로 서로 다르지만 Kerberos 문서에서 종종 바꿔서 사용하기도 합니다.

Kerberos(Cerberus라고도 씀)는 그리스 신화에서 지옥의 문을 지키는 머리가 셋 달린 사나운 개입니다.

**Kerberos 정책** Kerberos 서비스에서 암호 사용을 통제하는 규칙 세트입니다. 정책을 통해 주체의 액세스나 티켓 수명 매개변수를 규제할 수 있습니다.

**keytab 파일** 하나 이상의 키(주체)를 포함하는 키 테이블 파일입니다. 사용자가 암호를 사용하는 것처럼 호스트나 서비스는 keytab 파일을 사용합니다.

**kvno** 키 버전 번호. 생성 순서대로 특정 키를 추적하는 시퀀스 번호입니다. 가장 높은 kvno가 최신의 가장 현재 키입니다.

**MAC** 1. MAC(메시지 인증 코드)를 참조하십시오.

2. 레이블 지정이라고도 합니다. 정부 보안 기술에서 MAC은 필수 액세스 제어입니다. MAC의 예로 Top Secret 및 Confidential과 같은 레이블이 있습니다. MAC은 DAC(모든 액세스 제어)과 대조를 이룹니다. DAC의 예로 UNIX 사용 권한이 있습니다.

3. 하드웨어에서 LAN의 고유한 시스템 주소입니다. 시스템이 이더넷에 있는 경우 MAC은 이더넷 주소입니다.

**MAC(메시지 인증 코드)** MAC는 데이터 무결성을 보증하고 데이터 발신을 인증합니다. MAC는 도청에 대해 보호되지 않습니다.

**MD5** 디지털 서명을 포함하여 메시지 인증용으로 사용되는 반복적인 암호화 해시 함수입니다. 이 기능은 1991년 Rivest가 개발했습니다. 이 기술은 더 이상 사용되지 않습니다.

**NTP** Network Time Protocol의 약자. 네트워크 환경에서 정밀한 시간이나 네트워크 클럭 동기화(또는 둘 다)를 관리할 수 있는 델라웨어 대학교에서 설계한 소프트웨어입니다. NTP를 사용하여 Kerberos 환경에서 클럭 불균형을 유지 관리할 수 있습니다. 클럭 불균형도 참조하십시오.

**PAM** Pluggable Authentication Module의 약자. 여러 인증 방식에서 서비스를 재컴파일할 필요 없이 사용할 수 있는 프레임워크입니다. PAM은 로그인 시 Kerberos 세션 초기화를 사용으로 설정합니다.

**privileged user(권한 있는 사용자)** 컴퓨터 시스템에서 일반 사용자의 권한을 벗어난 권한이 지정된 사용자입니다. trusted users(신뢰할 수 있는 사용자)도 참조하십시오.

**QOP** Quality of Protection의 머리글자어로, 보호 품질입니다. 무결성 서비스나 프라이버시 서비스와 함께 사용되는 암호화 알고리즘을 선택할 수 있는 매개변수입니다.

**RBAC** 역할 기반 액세스 제어의 머리글자어로, Oracle Solaris의 사용자 권한 관리 기능입니다. rights(권한)을 참조하십시오.

**RBAC 정책** rights policy(권한 정책)을 참조하십시오.

**reauthentication(재인증)** 컴퓨터 작업을 수행하기 위해 암호를 제공하도록 요구하는 것입니다. 일반적으로 `sudo` 작업에는 재인증이 필요합니다. 인증된 권한 프로파일에는 재인증이 필요한 명령이 포함될 수 있습니다. authenticated rights profile(인증된 권한 프로파일)을 참조하십시오.

**rights policy(권한 정책)** 명령과 연관된 보안 정책입니다. 현재 Oracle Solaris에 유효한 정책은 `solaris`입니다. `solaris` 정책은 권한과 확장 권한 정책, 권한 부여 및 `setuid` 보안 속성을 인식합니다.

**rights(권한)** all-or-nothing 수퍼 유저 모델의 대안입니다. 사용자 권한(right) 관리 및 프로세스 권한(right) 관리를 통해 조직은 수퍼 유저의 권한(privilege)을 분담하고 이를 사용자 또는 역할에 지정할 수 있습니다. Oracle Solaris의 권한(right)은 커널 권한(privilege), 권한 부여 및 프로세스를 특정 UID 또는 GID로 실행하는 기능으로 구현됩니다. 권한 프로파일 및 역할에서 권한을 수집할 수 있습니다.

**RSA** 디지털 서명 및 공개 키 암호화 체계를 얻기 위한 방법입니다. 1978년에 개발자 Rivest, Shamir, Adleman이 처음 기술했습니다.

**SEAM** Solaris 시스템의 Kerberos 초기 버전에 대한 제품 이름입니다. 이 제품은 MIT(Massachusetts Institute of Technology)에서 개발된 Kerberos V5 기술을 기반으로 합니다. 이제 SEAM을 Kerberos 서비스라고 합니다. SEAM은 계속해서 MIT 버전과 약간 다릅니다.

**Secure Shell** 비보안 네트워크를 통해 보안 원격 로그인 및 기타 보안 네트워크 서비스를 제공하기 위한 특수한 프로토콜입니다.

**SHA1** Secure Hashing Algorithm의 머리글자어입니다. 이 알고리즘은 $2^{64}$ 미만의 입력 길이에서 작동하여 메시지 다이제스트를 생성합니다. SHA1 알고리즘은 DSA에 입력됩니다.

**stash 파일** stash 파일은 KDC용 마스터 키의 암호화된 복사본을 포함합니다. `kadmind` 및 `krb5kdc` 프로세스를 시작하기 전에 KDC를 자동으로 인증하도록 서버를 재부트할 때 이 마스터 키가 사용됩니다. stash 파일에 마스터 키가 포함되므로 stash 파일과 그 백업을 안전하게 보관해야 합니다. 암호화가 훼손되면 키를 사용하여 KDC 데이터베이스를 액세스하거나 수정해야 합니다.

**TGS** Ticket-Granting Service의 머리글자어로, 티켓 부여 서비스입니다. 티켓 발행을 담당하는 KDC의 부분입니다.

**TGT** Ticket-Granting Ticket의 머리글자어로, 티켓 부여 티켓입니다. 클라이언트가 다른 서비스에 대한 티켓을 요청할 수 있는 KDC에서 발행한 티켓입니다.

**trusted users(신뢰할 수 있는 사용자)** 결정된 사용자는 일부 신뢰 레벨에서 관리 작업을 수행할 수 있습니다. 일반적으로 관리자는 먼저 신뢰할 수 있는 사용자에 대한 로그인을 만들고 사용자의 신뢰 레벨 및 기능과 일치하는 관리 권한을 지정합니다. 이러한 사용자는 시스템을 구성 및 유지 관리하는 데 도움이 됩니다. 권한 있는 사용자라고도 합니다.

**VPN(가상 사설망)** 암호화를 사용하고 공용 네트워크를 통한 사용자 연결을 터널링하여 보안 통신을 제공하는 네트워크입니다.

# 색인

**ㅎ**