리눅스 파일 시스템과 권한 관리

Linux File System and Permission Management

목차

1. 개요

리눅스 운영체제의 핵심 개념 중 하나는 '모든 것은 파일이다'라는 철학입니다. 이 철학에 따라 리눅스는 일반 파일뿐만 아니라 디렉토리, 하드웨어 장치, 프로세스 간 통신까지도 파일로 다룹니다. 이러한 통일된 접근 방식은 시스템 관리를 단순화하고 일관성을 제공합니다.

본 보고서에서는 리눅스 파일 시스템의 계층 구조, 다양한 파일 타입, 그리고 보안의 근간이 되는 권한 관리 시스템에 대해 다룹니다. 특히 권한 관리는 멀티유저 환경에서 시스템 보안을 유지하는 핵심 메커니즘으로, 실무에서 반드시 이해해야 할 개념입니다.

1.1 리눅스 파일 시스템의 중요성

파일 시스템은 운영체제가 저장 장치에 데이터를 조직화하고 관리하는 방식을 정의합니다. 리눅스는 계층적 디렉토리 구조를 통해 모든 파일을 체계적으로 관리하며, 권한 시스템을 통해 각 파일과 디렉토리에 대한 접근을 제어합니다.

2. 리눅스 파일 시스템 구조

2.1 루트 디렉토리와 계층 구조

리눅스 파일 시스템은 루트 디렉토리(/)를 최상위로 하는 단일 계층 트리 구조를 가집니다. Windows와 달리 C:, D: 같은 드라이브 개념이 없으며, 모든 파티션과 장치는 루트 아래의 특정 디렉토리에 마운트됩니다.

2.1.1 마운트(Mount) 개념

마운트는 파일 시스템을 특정 디렉토리에 연결하는 과정입니다. 예를 들어, USB 드라이브를 /mnt/usb에 마운트하면 해당 장치의 파일들이 이 경로를 통해 접근 가능해집니다. 이는 서로 다른 파티션이나 디스크라도 하나의 통합된 디렉토리 트리로 보이게 만듭니다.

2.2 주요 디렉토리와 역할

디렉토리	역할 및 설명
/	루트 디렉토리. 모든 파일과 디렉토리의 최상위 위치
/bin	기본적인 사용자 명령어들 (ls, cp, mv 등)
/boot	부팅에 필요한 커널 이미지 및 설정 파일
/etc	시스템 설정 파일과 관리자용 스크립트
/home	일반 사용자의 홈 디렉토리가 생성되는 위치 (/home/ubuntu 등)
/root	관리자(root) 계정의 홈 디렉토리
/dev	하드웨어 장치 파일 (/dev/sda, /dev/console 등)
/lib	공유 라이브러리 파일
/tmp	임시 파일 저장소 (재부팅 시 삭제될 수 있음)
/var	가변 데이터 (로그 파일, 메일 스풀 등). /var/log에 시스템 로그 저장
/proc	가상 파일 시스템. 커널과 프로세스 정보를 파일 형태로 제공

디렉토리	역할 및 설명
/usr	사용자 애플리케이션과 파일. /usr/bin, /usr/lib 등 포함

각 디렉토리는 특정한 목적을 가지고 있으며, 이러한 표준화된 구조는 리눅스 배포판 간 일관성을 유지하고 시스템 관리를 용이하게 합니다.

3. 파일 타입과 특성

리눅스는 다양한 타입의 파일을 지원하며, 각 파일 타입은 고유한 용도와 특성을 가집니다. ls -l 명령어 실행 시 첫 번째 문자로 파일 타입을 식별할 수 있습니다.

3.1 파일 타입 분류

기호	파일 타입	설명
-	일반 파일	텍스트 파일, 바이너리 파일, 이미지 등
d	디렉토리	파일을 담는 컨테이너
l	심볼릭 링크	다른 파일을 가리키는 포인터 (Windows의 바로가기와 유사)
b	블록 장치	블록 단위로 데이터를 읽고 쓰는 장치 (/dev/sda)
c	문자 장치	문자 단위로 데이터를 읽고 쓰는 장치 (/dev/console)
p	파이프	프로세스 간 단방향 데이터 통신을 위한 파일
s	소켓	네트워크 프로세스 간 양방향 통신을 위한 파일

3.2 특수 파일의 활용

파이프와 소켓

파이프는 한 프로세스의 출력을 다른 프로세스의 입력으로 연결하는 단방향 통신 채널입니다. 명령줄에서 | 기호를 사용할 때 내부적으로 파이프 파일이 생성됩니다.

소켓 파일은 네트워크 통신을 위한 인터페이스로, 로컬 또는 원격 프로세스 간 양방향 통신을 지원합니다. 웹 서버, 데이터베이스 등의 네트워크 애플리케이션은 소켓을 통해 통신합니다.

4. 권한 관리 시스템

리눅스의 권한 관리 시스템은 멀티유저 환경에서 파일과 디렉토리에 대한 접근을 제어하는 핵심 보안 메커니즘입니다. 각 파일과 디렉토리는 소유자(Owner), 그룹(Group), 기타 사용자(Others)에 대한 읽기(r), 쓰기(w), 실행(x) 권한을 가집니다.

4.1 권한의 구조

ls -l 명령어의 출력 예시:

```
drwxr-xr-x 2 root root 4096 Mar 11 18:23 Desktop
```

위 예시를 분석하면 다음과 같습니다:

- **d**: 디렉토리
- **rwx**: 소유자(root)는 읽기, 쓰기, 실행 가능
- **r-x**: 그룹(root)은 읽기, 실행만 가능
- **r-x**: 기타 사용자는 읽기, 실행만 가능

4.2 권한의 의미

4.2.1 파일에 대한 권한

- **읽기(r)**: 파일 내용을 읽을 수 있는 권한. cat, vim, cp 등의 명령어 사용 가능
- **쓰기(w)**: 파일 내용을 수정, 변경할 수 있는 권한
- **실행(x)**: 파일을 실행할 수 있는 권한 (실행 파일, 스크립트 등)

4.2.2 디렉토리에 대한 권한

- **읽기(r)**: 디렉토리 내부의 파일 목록을 확인할 수 있는 권한 (ls 명령어)
- **쓰기(w)**: 디렉토리 내에 파일을 생성, 삭제할 수 있는 권한
- **실행(x)**: 디렉토리에 접근할 수 있는 권한. 이 권한이 없으면 cd 명령어 사용 불가

중요 사항: 디렉토리는 실행(x) 권한이 있어야 접근 가능합니다. 읽기(r) 권한만 있고 실행 권한이 없으면 디렉토리 내용을 볼 수는 있지만 들어갈 수 없습니다.

4.3 권한 변경 명령어

4.3.1 chmod - 권한 변경

chmod 명령어는 파일과 디렉토리의 권한을 변경합니다. 두 가지 모드가 있습니다:

심볼릭 모드:

```
chmod u+x script.sh
chmod g-w file.txt
```

```
chmod o+r document.pdf
```

숫자 모드:

권한을 숫자로 표현: r=4, w=2, x=1

`chmod 755 script.sh` (rwxr-xr-x)

`chmod 644 file.txt` (rw-r--r--)

`chmod 700 private.sh` (rwx------)

4.3.2 chown - 소유권 변경

파일이나 디렉토리의 소유자를 변경합니다:

```
chown ubuntu file.txt
chown ubuntu:staff file.txt
chown -R ubuntu /home/ubuntu/project
```

5. 보안 관점에서의 권한 관리

권한 관리는 리눅스 보안의 핵심입니다. 적절한 권한 설정은 시스템을 무단 접근과 악의적인 행위로부터 보호합니다.

5.1 최소 권한 원칙

최소 권한 원칙(Principle of Least Privilege)은 사용자나 프로세스에게 작업 수행에 필요한 최소한의 권한만 부여하는 보안 원칙입니다.

실무 적용 예시:

- 웹 서버 프로세스는 root 가 아닌 www-data 같은 제한된 사용자로 실행
- 설정 파일은 644 (rw-r--r--)로 설정하여 일반 사용자는 읽기만 가능
- 실행 파일은 755 (rwxr-xr-x)로 설정
- 개인 정보를 담은 파일은 600 (rw-------)으로 설정하여 소유자만 접근 가능

5.2 특수 권한

5.2.1 SetUID (Set User ID)

SetUID가 설정된 실행 파일을 실행하면, 실행하는 사용자가 아닌 파일 소유자의 권한으로 프로세스가 실행됩니다. 숫자 모드로는 4000으로 표현됩니다.

```
chmod 4755 /usr/bin/passwd
```

보안 위험: SetUID는 강력한 기능이지만 잘못 사용하면 권한 상승(Privilege Escalation) 공격에 악용될 수 있습니다. 일반 사용자가 SetUID 비트가 설정된 root 소유 파일을 실행하면 root 권한을 획득할 수 있기 때문입니다.

5.2.2 SetGID (Set Group ID)

SetGID는 파일의 경우 파일 소유 그룹의 권한으로 실행되며, 디렉토리의 경우 해당 디렉토리에서 생성되는 모든 파일이 디렉토리의 그룹을 상속받습니다. 숫자 모드로는 2000으로 표현됩니다.

5.2.3 Sticky Bit

Sticky Bit는 주로 공유 디렉토리에 사용됩니다. 이 권한이 설정된 디렉토리에서는 파일 소유자나 디렉토리 소유자, 또는 root만 파일을 삭제할 수 있습니다. 숫자 모드로는 1000으로 표현됩니다.

```
chmod 1777 /tmp
```

이 설정으로 /tmp 디렉토리에 모든 사용자가 파일을 생성할 수 있지만, 다른 사용자의 파일은 삭제할 수 없어 안전합니다.

5.3 실제 보안 사례

사례 1: 잘못된 권한 설정으로 인한 정보 유출

데이터베이스 설정 파일이 644 (rw-r--r--)로 설정되어 있어 모든 사용자가 비밀번호를 읽을 수 있었던 경우. 올바른 설정은 600 (rw-------)입니다.

사례 2: SetUID 비트를 이용한 권한 상승

공격자가 SetUID가 설정된 취약한 프로그램을 이용해 root 권한을 획득. 정기적인 SetUID 파일 점검이 필요합니다:

```
find / -perm -4000 -type f 2>/dev/null
```

6. 실무 활용 및 결론

6.1 권한 설정 베스트 프랙티스

실무에서 권한을 설정할 때는 다음 지침을 따릅니다:

7. **일반 파일:** 644 (소유자는 읽기/쓰기, 그 외는 읽기만)
8. **실행 파일:** 755 (소유자는 모든 권한, 그 외는 읽기/실행)
9. **디렉토리:** 755 (접근과 목록 확인 가능)
10. **민감한 파일:** 600 (소유자만 읽기/쓰기)
11. **로그 디렉토리:** 750 (소유자는 모든 권한, 그룹은 읽기/실행)

6.2 시스템 감사

정기적으로 시스템의 권한 설정을 점검해야 합니다:

- 잘못된 권한을 가진 파일 찾기: `find / -perm -002`
- SetUID 파일 점검: `find / -perm -4000`
- 특정 사용자 소유 파일 찾기: `find / -user username`

6.3 결론

리눅스의 파일 시스템과 권한 관리는 단순히 파일을 조직화하는 것을 넘어, 시스템 보안의 핵심 요소입니다. 계층적 디렉토리 구조는 시스템을 논리적으로 구성하고, 권한 시스템은 멀티유저 환경에서 각 사용자의 접근을 제어합니다.

특히 현대의 클라우드 환경과 컨테이너 기술에서도 리눅스의 권한 관리 개념은 그대로 적용됩니다. Docker 컨테이너 내부도 리눅스 파일 시스템을 사용하며, Kubernetes 환경에서도 Pod의 보안 컨텍스트를 설정할 때 리눅스 권한을 활용합니다.

효과적인 권한 관리를 위해서는:

- 최소 권한 원칙을 항상 적용
- 특수 권한(SetUID, SetGID)은 필요한 경우에만 사용
- 정기적인 권한 감사 수행
- 시스템 로그(/var/log)를 모니터링하여 이상 접근 탐지

리눅스의 파일 시스템과 권한 관리를 깊이 이해하는 것은 시스템 관리자뿐만 아니라 모든 개발자와 보안 전문가에게 필수적입니다. 이는 안전하고 효율적인 시스템 운영의 기초가 됩니다.

참고문헌

리눅스 시스템 관리 실습 교안

리눅스 교안