Übung: finde die minimal nötigen Berechtigungen

```
mkdir -m 000 /tmp/dir1 /tmp/dir2
echo geheim > file1
chmod 000 file1
```

Welche (minimalen) Berechtigungen müssen jeweils bei `/tmp/dir1`, `/tmp/dir2` und `file1` hinzugefügt werden, damit `file1` mit

```
cp file1 /tmp/dir1/file2
```

ins Verzeichnis `/tmp/dir1` kopiert werden kann?

Welche (minimalen) Berechtigungen müssen jeweils bei `/tmp/dir1`, `/tmp/dir2` und `file2` hinzugefügt werden, damit `file2` mit

```
mv /tmp/dir1/file2 /tmp/dir2/
```

ins Verzeichnis `dir2` verschoben werden kann?

Übung: mit umask Berechtigung setzen

Welche `umask` muss man setzen, damit neu angelegte

- Dateien die Berechtigung `rw- r-- -w-` und
- Verzeichnisse die Berechtigung `rwx r-x -w-`

bekommen?

Befehle Dateirechte

Dateizugriffsrechte betrachten	`ls -l datei`
Dateizugriffsrechte von Dateien im Verzeichnis betrachten	`ls -l verzeichnis`
Verzeichnisrechte betrachten (nicht der Einträge im Verzeichnis)	`ls -ld verzeichnis`
Besitzer der Datei ändern	`chown benutzer datei`
Gruppe der Datei ändern	`chown :users datei` `chgrp users datei`

SUID Recht für die Datei `/bin/cat` setzen	Programm starten und real bzw. effective UID ansehen:
	`chmod u+s /bin/cat`
	Als Nutzer:
	`cat &`
	`ps -C cat -o cmd,ruser,euser`

Wo darf ich schreiben?	`find / /dev -xdev -writable -ls`
Was darf eine Gruppe?	`find / /dev -xdev -group users -ls`
Alle SUID-root Dateien finden	`find / -xdev -type f -user root -perm /4000 -ls 2>/dev/null` `find / -xdev -type f -perm -u=s -ls`
Allen Programmen das Ausführungsrecht nehmen	`find verzeichnis/ -type f -perm /0111 -exec chmod a-x {} \;` `find verzeichnis/ -type f -perm /0111 -print0 | xargs -0 chmod a-x`

Beispiel: Schreibrechte im Verzeichnis - mehr als man denkt

```
mkdir /test
id nutzer17
```

```
uid=1001(nutzer17) gid=100(users) groups=100(users),16(dialout),33(video)
```

```
id iw
```

```
uid=1000(iw) gid=100(users)
groups=100(users),16(dialout),17(audio),33(video)
```

```
chown iw:users /test/
chmod g+w /test/
ls -ld /test/
```

```
drwxrwxr-x  2 iw users 4096 Oct 10 17:30 /test/
```

```
su - iw
cat <<EOF > /test/unveraenderbar.txt
```

```
Dies ist ein unveraenderlicher Text
```

```
EOF
chmod u=rw,g=r,o=r /test/unveraenderbar.txt
logout
su - nutzer17
ls -l /test/unveraenderbar.txt
```

```
-rw-r--r--  1 iw users 36 2005-10-10 17:32 /test/unveraenderbar.txt
```

```
vi /test/unveraenderbar.txt
ls -la /test/
```

```
insgesamt 12
drwxrwxr-x   2 iw       users 4096 2005-10-10 17:35 .
drwxr-xr-x  22 root     root  4096 2005-10-10 17:30 ..
-rw-r--r--   1 nutzer17 users   34 2005-10-10 17:35 unveraenderbar.txt
```

Übung: Weniger Rechte für Eigentümer und Gruppe

Erzeuge ein Verzeichnis `/srv/open` mit großzügigen Berechtigungen:

```
mkdir -m 777 /srv/open
```

Erzeuge eine Datei `/srv/open/datei356` mit folgendem Inhalt:

/srv/open/datei356

```
#!/bin/bash
echo executable
```

Setze die Berechtigungen zu 356:

```
chmod 356 /srv/open/datei356
```

Wer darf was?

r w x	**Mitglied der Gruppe**	**nicht Mitglied der Gruppe**
Besitzer	_ _ _	_ _ _
nicht Besitzer	_ _ _	_ _ _

Tip: Das Lesen (`r`) von Dateien kann man mit

```
head -0 /srv/open/datei356
```

testen. [1)] Das Schreiben (`w`) von Dateien kann man zerstörungsfrei mit

```
>> /srv/open/datei356
```

testen. [2)] Das Ausführen (`x`) kann man testen, in dem man

```
/srv/open/datei356
```

ausführt.

Beispiel: Gruppenzugehörigkeiten kleben am Prozess länger als man denkt

```
groupadd projekt
useradd -m -G projekt iw
touch /tmp/datei
chown root:projekt /tmp/datei
chmod g+w /tmp/datei
su - iw
id iw
echo test1 >> /tmp/datei
su -
usermod -G users iw
id iw
logout
id iw
echo test1 >> /tmp/datei
```

Beispiel: Ungewollter Eigentümerwechsel nach dem Löschen eines Nutzers

```
useradd -m chef
userdel chef
useradd -m raumpflege
ls -l /home
```

Daher sollte man vor oder nach dem Löschen einer Benutzerkennung sämtliche Dateien, die diesem Benutzer gehörten, als `root` übernehmen oder einem anderen, für diesen Zweck erstellten technischen Benutzer übereignen.

Beispiel: SUID-Bit Dateien finden und SUID-Bit dauerhaft entfernen

SUID-Bit Dateien finden (als root):

```
find / /boot -xdev -perm /u+s -type f -user root -ls
```

SUID-Bit dauerhaft entfernen: Ubuntu / Debian:

```
dpkg-statoverride --update --add root root 0755 /bin/ping
```

Beispiel: Auswirkung von mount-Optionen

```
mkdir /mnt/sda2
mount -o ro /dev/sda2 /mnt/sda2
touch /mnt/sda2/test
```

Weitere Mount-Optionen mit Auswirkungen auf Dateirechte:

Option	Bedeutung
nodev	Gerätedateien sind nicht erlaubt
noexec	Ausführbare Dateien sind nicht erlaubt
nosuid	S-BIT wird ignoriert
ro	Dateien sind nicht veränderbar

Beispiel: Ungewollter Eigentümerwechsel bei Backup und Restore

Server A

```
useradd -u 2000 nutzer_a
useradd -u 2001 nutzer_b
useradd -u 2002 nutzer_c
useradd -u 2003 nutzer_d
```

Server B

```
useradd -u 2000 nutzer_a
useradd -u 2003 nutzer_b
useradd -u 2001 nutzer_e
```

```
mkdir /tmp/backup
```

Server A

```
mkdir /tmp/workdir
```

```
touch /tmp/workdir/file_{a,b,c,d}
chown nutzer_a /tmp/workdir/file_a
chown nutzer_b /tmp/workdir/file_b
chown nutzer_c /tmp/workdir/file_c
chown nutzer_d /tmp/workdir/file_d
```

```
rsync -a /tmp/workdir/file_* server_b:/tmp/backup
```

Server B

```
ls -l /tmp/backup
```

Server A

```
mkdir /tmp/restore
rsync -a server_b:/tmp/backup/file_* /tmp/restore
ls -l /tmp/restore
```

[1)]
head -0 zeigt die ersten 0 Zeilen, also nichts, aber öffnet die Datei. Im Fehlerfall erscheint eine Meldung.
[2)]

```
sysctl fs.protected_regular
```

```
fs.protected_regular = 2
```

https://www.kernel.org/doc/Documentation/sysctl/fs.txt: *This protection is similar to protected_fifos, but it avoids writes to an attacker-controlled regular file, where a program expected to create one. When set to „0“, writing to regular files is unrestricted. When set to „1“ don't allow O_CREAT open on regular files that we don't own in world writable sticky directories, unless they are owned by the owner of the directory. When set to „2“ it also applies to group writable sticky directories.*