ACL: uprawnienia ponad model właściciel-grupa-inni

Poziom: średni

Dotyczy: RHEL/Rocky 9–10 (ext4/xfs), różnice Debian/Ubuntu i SLES zaznaczone

Czas: ~17 min

Klasyczny model uprawnień zna tylko trzech odbiorców: właściciela, jedną grupę i resztę świata. ACL (Access Control List) dokłada dowolną liczbę dodatkowych wpisów dla konkretnych użytkowników i grup, bez ruszania właściciela ani grupy pliku. Ta strona pokazuje, jak z tego korzystać i gdzie leżą jego dwie najczęstsze pułapki: maska, która po cichu przycina nadane prawa, i `cp`, które gubi ACL bez ostrzeżenia.

Skróty użyte na tej stronie (5)

Skrót	Co znaczy
ACL	Access Control List: lista, kto ma jakie prawa do pliku lub zasobu
RHEL	Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa
SLES	SUSE Linux Enterprise Server: płatna dystrybucja Linuksa firmy SUSE, konkurent Rocky/RHEL
UID	User ID: numer użytkownika w Linuksie
XFS	system plików stworzony przez Silicon Graphics, domyślny w Rocky Linux i RHEL

Fiszka

TYLKO ODCZYT

Odczyt pełnej listy kontroli dostępu: getfacl

```
getfacl NazwaPliku
```

Pokazujemy wszystkie wpisy ACL pliku, łącznie z tymi, które odpowiadają klasycznym `rwx` właściciela, grupy i pozostałych — one też są reprezentowane jako wpisy ACL, tylko nazwane

inaczej (`user::`, `group::`, `other::`).

Przykład:

```
# file: raport.txt
# owner: adm_user
# group: deweloperzy
user::rw-
user:jkowalski:rwx
group::r--
mask::rwx
other::---
```

ACL = Access Control List: lista, kto ma jakie prawa do pliku lub zasobu · **getfacl** = get Access Control List: pokaż dodatkowe uprawnienia ACL nadane na pliku albo katalogu

TYLKO ODCZYT

Dodanie wpisu dla konkretnego użytkownika: setfacl -m u

```
setfacl -m u:nazwa_usera:rw NazwaPliku
```

`-m` (modify) dodaje albo nadpisuje wpis. Ten konkretny wpis daje `nazwa_usera` odczyt i zapis do pliku, niezależnie od tego, jaka jest jego grupa główna i niezależnie od uprawnień `other`.

setfacl = set Access Control List: nadaj dodatkowe uprawnienia ACL ponad zwykłe prawa Unixa

TYLKO ODCZYT

Dodanie wpisu dla grupy: setfacl -m g

```
setfacl -m g:NazwaGrupy:rx /srv/NazwaKatalogu
```

Analogicznie dla grupy — `NazwaGrupy` dostają odczyt i wejście do katalogu, choć katalog formalnie może należeć do zupełnie innej grupy.

setfacl = set Access Control List: nadaj dodatkowe uprawnienia ACL ponad zwykłe prawa Unixa

TYLKO ODCZYT

Usunięcie jednego wpisu: setfacl -x

```
setfacl -x u:nazwa_usera NazwaPliku
```

Kasujemy wyłącznie wskazany wpis, zostawiając resztę listy ACL bez zmian — w przeciwieństwie do `-b` poniżej.

ACL = Access Control List: lista, kto ma jakie prawa do pliku lub zasobu · **setfacl** = set Access Control List: nadaj dodatkowe uprawnienia ACL ponad zwykłe prawa Unixa

TYLKO ODCZYT

Usunięcie całego rozszerzonego ACL: setfacl -b

```
setfacl -b NazwaPliku
```

Kasujemy wszystkie wpisy ACL wykraczające poza klasyczny model — plik wraca do zwykłego `rwx` właściciela/grupy/pozostałych, jakby ACL nigdy tam nie było.

ACL = Access Control List: lista, kto ma jakie prawa do pliku lub zasobu · **setfacl** = set Access Control List: nadaj dodatkowe uprawnienia ACL ponad zwykłe prawa Unixa

TYLKO ODCZYT

Rekurencyjne nadanie wpisu ACL: setfacl -R

```
setfacl -R -m g:NazwaGrupy:rx /srv/NazwaKatalogu
```

Nakładamy wpis na katalog i wszystko, co już w nim jest. Nie dotyczy plików, które powstaną w tym katalogu **później** — do tego służy ACL domyślne, następny wpis Fiszki.

ACL = Access Control List: lista, kto ma jakie prawa do pliku lub zasobu · **setfacl** = set Access Control List: nadaj dodatkowe uprawnienia ACL ponad zwykłe prawa Unixa

TYLKO ODCZYT

Ustawienie ACL domyślnego dla nowych plików: setfacl -d -m

```
setfacl -d -m g:NazwaGrupy:rwx /srv/NazwaKatalogu
```

`-d` (default) ustawia ACL, które automatycznie dziedziczy każdy **nowo utworzony** plik i podkatalog wewnątrz `/srv/NazwaKatalogu` — odpowiednik `setgid` z poprzedniej strony, tylko dla pełnego zestawu uprawnień ACL, nie tylko dla grupy. Nie działa retroaktywnie na pliki, które już istniały w katalogu przed jego ustawieniem.

ACL = Access Control List: lista, kto ma jakie prawa do pliku lub zasobu · **setfacl** = set Access Control List: nadaj dodatkowe uprawnienia ACL ponad zwykłe prawa Unixa

ZMIENIA SYSTEM

Odtworzenie ACL z kopii zapasowej: setfacl --restore

```
setfacl --restore=acl-backup.txt
```

Odczytujemy plik w formacie wygenerowanym przez `getfacl -R` i nakładamy zapisane w nim wpisy z powrotem na dokładnie te same ścieżki — komenda musi być uruchomiona z tego samego katalogu roboczego, względem którego robiono kopię, bo ścieżki w pliku kopii są zapisane relatywnie.

ACL = Access Control List: lista, kto ma jakie prawa do pliku lub zasobu · **setfacl** = set Access Control List: nadaj dodatkowe uprawnienia ACL ponad zwykłe prawa Unixa

ZMIENIA SYSTEM

Utworzenie kopii zapasowej całego drzewa ACL: getfacl -R > backup

```
getfacl -R /srv/NazwaKatalogu > acl-backup.txt
```

Zrzucamy ACL każdego pliku i katalogu w drzewie do jednego pliku tekstowego — dokładnie w formacie, który rozumie `setfacl --restore`. Warto to robić przed każdą masową zmianą uprawnień na współdzielonym zasobie.

ACL = Access Control List: lista, kto ma jakie prawa do pliku lub zasobu · **getfacl** = get Access Control List: pokaż dodatkowe uprawnienia ACL nadane na pliku albo katalogu

ZMIENIA SYSTEM

Skopiowanie ACL między katalogami

```
getfacl /srv/projekt-szablon | setfacl --set-file=- /srv/projekt-nowy
```

Klasyczny trik na powielenie konfiguracji ACL z jednego katalogu (szablonu) na drugi: `getfacl` wypisuje ACL na standardowe wyjście, `setfacl --set-file=-` czyta je ze standardowego wejścia (myślnik) i nakłada w całości, nadpisując dotychczasowy ACL katalogu docelowego.

ACL = Access Control List: lista, kto ma jakie prawa do pliku lub zasobu · **getfacl** = get Access Control List: pokaż dodatkowe uprawnienia ACL nadane na pliku albo katalogu

◫◫ ZMIENIA SYSTEM

Sprawdzenie, czy punkt montowania ma włączone ACL: mount | grep acl

```
mount | grep acl
```

Sprawdzamy, czy dany system plików jest zamontowany z opcją `acl`. Pułapka: na współczesnych jądrach ACL bywa włączone **domyślnie** dla ext4 i zawsze aktywne dla xfs — w takim wypadku ta komenda może nie zwrócić nic, mimo że ACL faktycznie działa. Brak wyniku nie jest równoznaczny z brakiem obsługi ACL — patrz następny wpis i sekcję *Częste błędy*.

Przykład:

```
/dev/mapper/vgPostgres-lvPostgres on /srv type ext4 (rw,relatime,acl)
```

◫◫ **ACL** = Access Control List: lista, kto ma jakie prawa do pliku lub zasobu · **mount** = podepnij system plików z urządzenia albo obrazu pod wskazany katalog

◫◫ ZMIENIA SYSTEM

Sprawdzenie domyślnych opcji montowania zapisanych w systemie plików (ext4): tune2fs -l

```
tune2fs -l /dev/mapper/vgPostgres-lvPostgres | grep -i 'default mount options'
```

Pokazujemy opcje montowania zapisane w superbloku systemu plików ext4 w momencie jego utworzenia (`mkfs.ext4`) — jeśli `acl` jest tutaj, jądro włącza obsługę ACL automatycznie, nawet gdy `/etc/fstab` i `mount | grep acl` niczego nie pokazują wprost.

Przykład:

```
Default mount options:   user_xattr acl
```

Na XFS ACL jest wbudowane bezwarunkowo i nie ma odpowiednika tej flagi do sprawdzenia — najpewniejszym testem jest po prostu próba `setfacl` na realnym pliku.

◫◫ **ACL** = Access Control List: lista, kto ma jakie prawa do pliku lub zasobu · **XFS** = system plików stworzony przez Silicon Graphics, domyślny w Rocky Linux i RHEL

◫◫ TYLKO ODCZYT

Wszystkie komendy

Komenda	Co robi
`getfacl`	odczyt pełnej listy ACL pliku/katalogu
`setfacl -m u:nazwa_usera:prawa`	dodanie/zmiana wpisu dla użytkownika
`setfacl -m g:NazwaGrupy:prawa`	dodanie/zmiana wpisu dla grupy
`setfacl -x wpis`	usunięcie jednego wpisu
`setfacl -b`	usunięcie całego rozszerzonego ACL
`setfacl -R`	rekurencja na istniejące pliki
`setfacl -d -m`	ACL domyślne dla przyszłych, nowych plików
`setfacl --restore=plik`	odtworzenie z kopii zapasowej
`getfacl -R > plik`	kopia zapasowa całego drzewa
`getfacl src \ | setfacl --set-file=- dst`	powielenie ACL między katalogami

Komenda	Co robi

`šmount \`
`p`
`r`
`a`
`w`
`d`
`z`
`e`
`n`
`i`
`e`
`w`
`s`
`p`
`a`
`r`
`c`
`i`
`a`
`A`
`C`
`L`
`n`
`a`
`s`
`y`
`s`
`t`
`e`
`m`
`i`
`e`
`p`
`l`
`i`
`k`
`ó`
`w`
	`grep acl , tune2fs -l \`
	`g`
	`r`
	`e`
	`p`
	`a`
	`c`
	`l`
	`` ` ``

Uważaj

- Znak `+` na końcu uprawnień w `ls -l` to jedyny sygnał w standardowym widoku, że plik ma rozszerzone ACL — bez `getfacl` tego nie zobaczymy.
- `cp` bez `-a` (i bez `--preserve=all` albo `-p` w połączeniu z odpowiednim systemem plików) **gubi ACL** pliku źródłowego — kopia dostaje ACL wynikające z domyślnego ACL katalogu docelowego, nie z pliku źródłowego.
- Maska ACL (`mask::`) potrafi po cichu obciąć każde nadane prawo — `setfacl` pokaże sukces, a efektywne uprawnienia mogą być niższe niż to, co właśnie

ustawiliśmy.

ACL = Access Control List: lista, kto ma jakie prawa do pliku lub zasobu

Z życia wzięte

Z ŻYCIA WZIĘTE

O 11:00 przyjeżdża pan Wojtek na coroczny audyt i potrzebuje dostępu tylko do odczytu w katalogu księgowości na fs01, bez wchodzenia do grupy księgowych. Tomek dopisuje mu jeden wpis `setfacl -m u:audytor:r-x` zamiast tworzyć nową grupę na potrzeby jednej wizyty. Po zakończeniu audytu wystarczy `setfacl -x u:audytor`, żeby dostęp zniknął bez śladu w reszcie uprawnień katalogu.

Po co to istnieje

Pojęcie: lista kontroli dostępu (ACL). ACL to rozszerzenie klasycznego modelu uprawnień, które pozwala przypisać prawa dostępu dowolnej liczbie konkretnych użytkowników i grup na jednym pliku czy katalogu — nie tylko jednemu właścicielowi, jednej grupie i „reszcie świata”. Wpisy ACL są przechowywane w rozszerzonych atrybutach systemu plików (extended attributes, `xattr`), obok zwykłych metadanych uniksowych, i odczytywane przez jądro przy każdej próbie dostępu.

Klasyczny model z poprzednich stron tego rozdziału ma twardy limit: jedna grupa na plik. Gdy trzeba dać dostęp jednej dodatkowej osobie spoza tej grupy — zewnętrznemu konsultantowi, audytorowi, drugiemu zespołowi — bez zmiany właściciela ani grupy pliku (bo to zepsułoby dostęp reszcie), klasyczny model nie ma na to sposobu innego niż stworzenie nowej grupy obejmującej wszystkich zainteresowanych, co szybko robi się nieporządne przy wielu takich wyjątkach naraz. ACL rozwiązuje to wprost: dopisujesz jeden wpis dla tej jednej osoby, nie ruszając niczego innego.

Analogia

Klasyczny model uprawnień to zamek w drzwiach z trzema rodzajami kluczy: głównym (właściciel), do klatki (grupa) i klamką bez zamka (reszta świata). ACL to dodatkowa **lista gości na recepcji**, obok tych trzech kluczy.

Recepcjonista ma podstawową zasadę (kto ma klucz, wchodzi), ale trzyma też osobną listę: „Pan Kowalski, mimo że nie ma klucza do klatki, ma wstęp do sali 214 do końca miesiąca”. Ta lista działa niezależnie od tego, kto formalnie jest najemcą piętra. Jest jednak jeden szczegół, o którym łatwo zapomnieć: zarządca budynku (maska ACL) może mieć ogólną politykę „nikt z listy gości nie wchodzi wyżej niż na pierwsze piętro”, niezależnie od tego, co obiecał recepcjonista konkretnemu gościowi. Pan Kowalski dostał przepustkę na salę 214 (trzecie piętro), ale polityka zarządcy przycina to do parteru — wpis na liście istnieje, ale efektywnie nie działa tak, jak zapisano.

Jak to działa

Rozwiń: jak to działa pod spodem

Znak `+` w `ls -l`. Standardowy widok `ls -l` nie ma miejsca na wypisanie rozszerzonych wpisów ACL — pokazuje wyłącznie klasyczne dziewięć bitów. Jedynym śladem, że plik ma coś więcej, jest znak `+` doklejony na końcu kolumny uprawnień:

```
-rw-rw-r--+ 1 adm_user deweloperzy 812 wrz 17 11:00 raport.txt
```

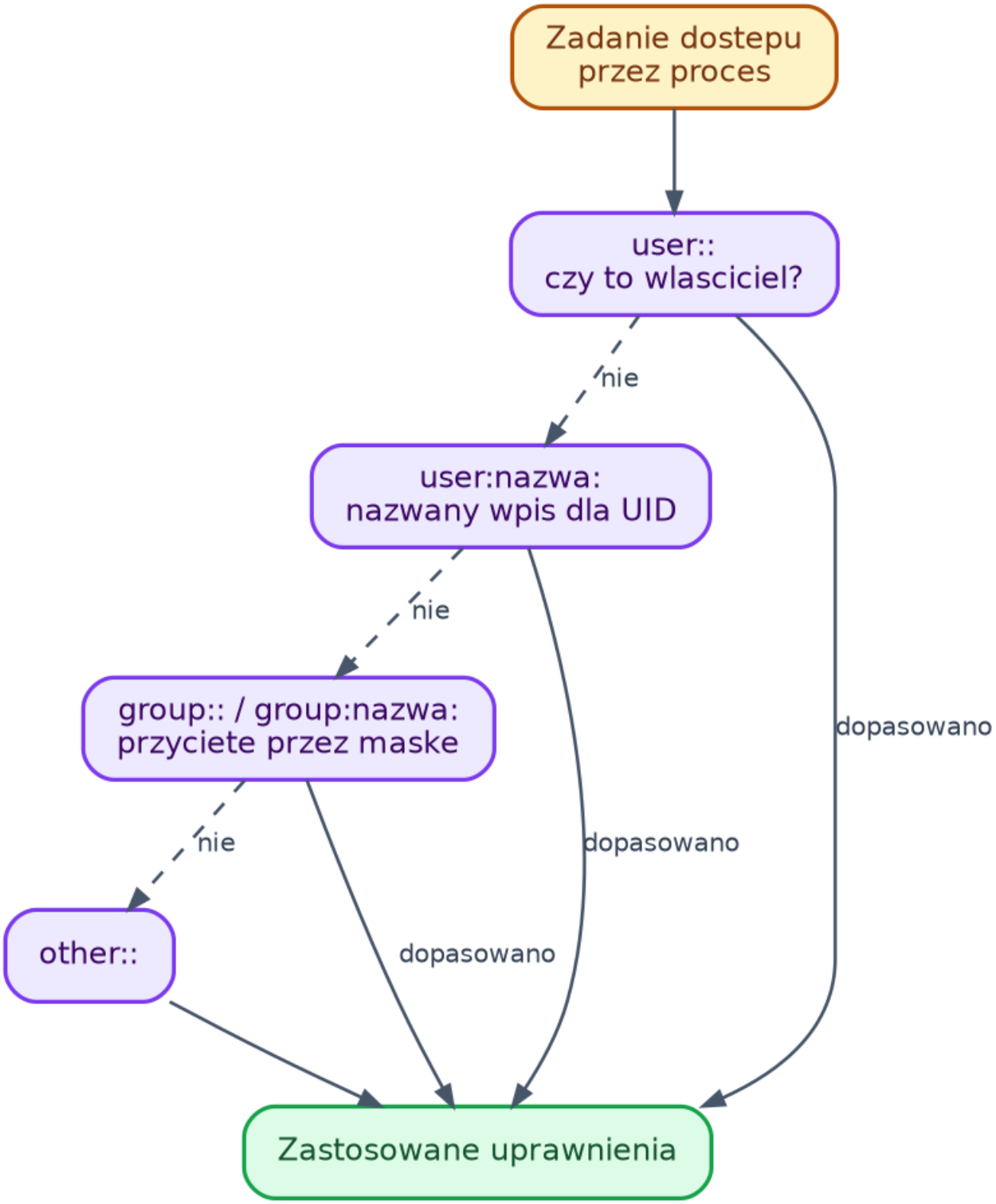
Bez tego `+` nikt patrzący tylko na `ls -l` nie dowie się, że istnieje dodatkowy, niewidoczny w tym widoku zestaw reguł. To najczęstsza przyczyna zaskoczenia przy audycie bezpieczeństwa: administrator patrzy na `chmod 600` i uznaje plik za w pełni prywatny, nie zauważając `+` i nie sprawdzając `getfacl` — a tam może czekać wpis `user:ktos:rwx` przyznający dostęp mimo restrykcyjnego trybu klasycznego.

> **Pojęcie: maska ACL.** Wpis `mask::` w wyniku `getfacl` to górny limit efektywnych uprawnień dla **wszystkich** wpisów nazwanych — konkretnych użytkowników (`user:nazwa:`) i grup (`group:nazwa:`), a także dla grupy właściciela (`group::`). Realne, egzekwowane przez jądro uprawnienia to część wspólna (bitowe AND) wpisu i maski, nie sam wpis. `setfacl -m u:jkowalski:rwx` może więc ustawić dokładnie to, o co poproszono, a `getfacl` mimo to pokaże:

```
user:jkowalski:rwx    #effective:r-x
```

`#effective:r-x` to komentarz `getfacl`, informujący, że realne uprawnienia są niższe niż zapisany wpis — bo maska w tym przykładzie wynosi `r-x`, nie `rwx`. Maska nie dotyczy właściciela pliku (`user::`) ani „pozostałych” (`other::`) — to jedyne dwa wpisy, które zawsze działają w pełni, niezależnie od maski.

Pojęcie: ACL domyślne. Zwykły wpis ACL (ustawiony przez `setfacl -m`) dotyczy istniejącego obiektu — pliku albo konkretnego katalogu, na którym go ustawiono. ACL domyślne (`setfacl -d -m`, ustawiane wyłącznie na katalogach) to osobna, dodatkowa lista, która nie wpływa bezpośrednio na sam katalog, tylko określa, jaki ACL dostanie każdy nowy plik i podkatalog utworzony wewnątrz — od momentu ustawienia, nigdy retroaktywnie. To mechanizm analogiczny do `setgid` z poprzedniej strony, tylko obejmujący pełen zestaw wpisów ACL, nie samą grupę.



Jądro sprawdza wpisy po kolei i zatrzymuje się na pierwszym dopasowaniu: właściciel, potem nazwany użytkownik, potem grupy przycięte przez maskę, na końcu reszta świata (kliknięcie otwiera pełny rozmiar)

Kolejność sprawdzania dostępu. Jądro sprawdza dopasowanie w ustalonej kolejności: najpierw, czy proces jest właścicielem (`user::`) — jeśli tak, stosuje te uprawnienia i kończy sprawdzanie. Potem, czy istnieje nazwany wpis dla dokładnie tego UID (`user:nazwa:`) — jeśli tak, stosuje go (przycięty przez maskę) i kończy. Dopiero potem sprawdza grupę właściciela i nazwane grupy (`group::`, `group:nazwa:`), zawsze przycięte przez maskę, biorąc najbardziej pasujący/najszerszy zestaw spośród dopasowanych grup. Na końcu, jeśli nic innego nie pasowało, stosuje `other::`. Ta kolejność ma praktyczną konsekwencję: nazwany wpis użytkownika **wygrywa** z uprawnieniami grupy, nawet jeśli grupa dawałaby więcej — pierwsze dopasowanie w tej hierarchii kończy sprawdzanie.

Krok po kroku

1. Sprawdź, czy system plików w ogóle obsługuje ACL.

```
findmnt -no OPTIONS /srv | tr ',' '\n' | grep -w acl
tune2fs -l /dev/mapper/vgPostgres-lvPostgres 2>/dev/null | grep -i 'default mount options'
```

Brak wyniku w obu miejscach nie jest wyrokiem — spróbuj wprost:

```
touch /srv/test_acl
setfacl -m u:nobody:r /srv/test_acl && echo "ACL działa"
rm /srv/test_acl
```

2. Zbuduj katalog projektowy: grupa, gość spoza grupy, domyślne ACL na przyszłość.

```
mkdir /srv/projekt
chgrp deweloperzy /srv/projekt
chmod 2770 /srv/projekt
setfacl -m u:audytor_zewnetrzny:r-x /srv/projekt
setfacl -d -m g:deweloperzy:rwx /srv/projekt
setfacl -d -m u:audytor_zewnetrzny:r-x /srv/projekt
```

3. Zweryfikuj cały efekt jedną komendą.

```
getfacl /srv/projekt
```

```
# file: projekt
# owner: adm_user
# group: deweloperzy
```

```
# flags: -s-
user::rwx
user:audytor_zewnetrzny:r-x
group::rwx
mask::rwx
other::---
default:user::rwx
default:group::rwx
default:group:deweloperzy:rwx
default:user:audytor_zewnetrzny:r-x
default:mask::rwx
default:other::---
```

4. Zrób kopię zapasową konfiguracji ACL przed dalszymi zmianami.

```
getfacl -R /srv/projekt > /root/acl-projekt-backup.txt
```

Częste błędy

`chmod` na pliku z ACL zmienia maskę, nie „zwykłą” grupę.

```
setfacl -m u:jkowalski:rwx raport.txt
chmod g-w raport.txt
getfacl raport.txt
```

```
user::rw-
user:jkowalski:rwx    #effective:r--
group::r--
mask::r--
other::---
```

Administrator chciał odebrać zapis grupie właścicielskiej pliku, ale gdy plik ma jakikolwiek rozszerzony ACL, `chmod` na polu grupy **nie modyfikuje** `group::` — zmienia maskę (`mask::`). Efekt uboczny: uprawnienia `jkowalski`, ustawione wcześniej na `rwx`, zostały po cichu przycięte do `r--`, mimo że nikt nie dotykał wpisu `jkowalski` wprost. Naprawa: na plikach z ACL uprawnienia grupowe reguluje się przez `setfacl -m g::prawa`, nie przez `chmod g`, a po każdej zmianie warto sprawdzić `getfacl` pod kątem komentarzy `#effective:`.

`cp` bez `-a` gubi cały ACL.

```
cp raport.txt kopia.txt
getfacl kopia.txt
```

```
# file: kopia.txt
user::rw-
group::r--
other::r--
```

Zwykłe `cp` tworzy nowy plik z uprawnieniami wynikającymi z `umask` procesu kopiującego (ewentualnie z ACL domyślnego katalogu docelowego) — nie kopiuje ACL pliku źródłowego, nawet jeśli plik docelowy leży w tym samym katalogu. Nazwany wpis `user:jkowalski:rwx` po prostu znika, bez żadnego ostrzeżenia. Naprawa: `cp -a raport.txt kopia.txt` (albo `cp --preserve=all`) zachowuje ACL razem z resztą metadanych, o ile system plików docelowy je obsługuje.

Maska przycina prawa, o które ktoś dopiero co poprosił.

```
setfacl -m u:jkowalski:rwx raport.txt
setfacl -m m::r-x raport.txt
getfacl raport.txt | grep jkowalski
```

```
user:jkowalski:rwx    #effective:r-x
```

Ustawienie maski jawnie (`m::r-x`) po dodaniu wpisu użytkownika przycina wszystkie nazwane wpisy do tego poziomu, nawet jeśli sam wpis nadal formalnie mówi `rwx`. `jkowalski` faktycznie nie zapisze pliku, mimo że polecenie `setfacl` dla niego zakończyło się sukcesem i `getfacl` nadal pokazuje `rwx` przy jego nazwie — trzeba doczytać komentarz `#effective:`, żeby to zauważyć. Domyślnie (bez ręcznego ustawiania maski przez `-m m::`) `setfacl` sam przelicza maskę jako sumę wszystkich wpisów po każdej zmianie — problem pojawia się dopiero, gdy ktoś ustawi maskę ręcznie albo użyje `setfacl -n`, które wyłącza automatyczne przeliczanie.

Poleganie na `chmod 600` bez sprawdzenia, czy plik ma ACL.

```
chmod 600 tajne.txt
ls -l tajne.txt
```

```
-rw-------+ 1 adm_user adm_user 214 wrz 17 11:40 tajne.txt
```

Plusik na końcu łatwo przeoczyć, zwłaszcza w skrypcie, który parsuje tylko pierwsze dziesięć znaków uprawnień. `chmod 600` ustawia klasyczne bity poprawnie, ale zostawia nietknięty ewentualny wcześniejszy wpis `user:ktos:rwx` — plik wygląda na w pełni prywatny, a ktoś inny nadal ma do niego pełny dostęp przez ACL. Naprawa: przy plikach z danymi wrażliwymi sprawdzać `getfacl`, nie tylko `ls -l`; `setfacl -b` czyści cały rozszerzony ACL, jeśli ma nie być tam żadnych wyjątków.

ACL domyślne nie obejmuje plików sprzed jego ustawienia.

```
ls /srv/projekt
setfacl -d -m g:deweloperzy:rwx /srv/projekt
ls -l /srv/projekt/stary_raport.txt
```

```
-rw-r--r-- 1 adm_user adm_user 812 wrz 10 08:00 stary_raport.txt
```

Brak `+` przy starym pliku — ACL domyślne zadziała dopiero na plikach utworzonych **po** tej komendzie, dokładnie jak przy `setgid` na poprzedniej stronie. Pliki, które już tam leżały, trzeba dociągnąć ręcznie: `setfacl -R -m g:deweloperzy:rwx /srv/projekt`.

□□ Ćwiczenia

> Wszystko poniżej wykonujesz na własnej maszynie, w katalogu `/tmp`. Ćwiczenia zakładają, że `/tmp` leży na systemie plików z obsługą ACL (typowe dla ext4/xfs na RHEL/Rocky, Debian/Ubuntu i SLES) — jeśli `setfacl` zwróci błąd „Operation not supported”, wykonaj ćwiczenia na innym, lokalnym katalogu.

Ćwiczenie 1 — maska przycina uprawnienia po cichu

```
cd /tmp
touch plik_acl.txt
chmod 640 plik_acl.txt
setfacl -m u:nobody:rwx plik_acl.txt
getfacl plik_acl.txt
setfacl -m m::r-- plik_acl.txt
getfacl plik_acl.txt
```

Porównaj oba wywołania `getfacl`. Co się zmieniło w linii dotyczącej `nobody`, mimo że nikt tego wpisu bezpośrednio nie ruszył?

Odpowiedź

Po pierwszym `getfacl` wpis `user:nobody:rwx` jest w pełni efektywny (maska automatycznie przeliczona na `rwx`). Po ręcznym ustawieniu maski na `r--` ten sam wpis pokazuje `user:nobody:rwx  #effective:r--` — sam zapis wpisu się nie zmienił, ale efektywne uprawnienia spadły do samego odczytu. To dokładnie mechanizm opisany w *Częste błędy*: maska działa jak sufit nałożony na wszystkie nazwane wpisy naraz.

Ćwiczenie 2 — `cp` gubi ACL, `cp -a` go zachowuje

```
cd /tmp
touch oryginal.txt
setfacl -m u:nobody:rw oryginal.txt
cp oryginal.txt kopia_zwykla.txt
cp -a oryginal.txt kopia_z_a.txt
getfacl kopia_zwykla.txt
getfacl kopia_z_a.txt
```

Który plik zachował wpis dla `nobody`, a który nie?

Odpowiedź

`kopia_zwykla.txt` nie ma znaku `+` ani wpisu `user:nobody:` — zwykłe `cp` nie kopiuje rozszerzonego ACL. `kopia_z_a.txt` zachowuje pełny ACL źródła, łącznie z wpisem dla `nobody`, bo `-a` (archive) każe `cp` zachować jak najwięcej metadanych, w tym ACL, o ile system plików docelowy je obsługuje.

Pytania kontrolne

1. Po co w ogóle sprawdzać `+` w `ls -l`, skoro można po prostu zawsze uruchamiać `getfacl`?

Odpowiedź

`ls -l` uruchamiasz w praktyce przy każdym przeglądaniu katalogu, `getfacl` — tylko gdy podejrzewasz, że coś jest nietypowe. Znak `+` jest tanim, natychmiastowym sygnałem „tu jest coś więcej, sprawdź to dokładniej” bez konieczności uruchamiania osobnej komendy na każdym

pliku w katalogu.

2. `setfacl -m u:jkowalski:rwx plik` zwraca sukces, ale `jkowalski` nadal nie może zapisać pliku. Jakie są dwa najbardziej prawdopodobne wytłumaczenia poza literówką w nazwie użytkownika?

Odpowiedź

Po pierwsze — maska ACL (`mask::`) jest ustawiona niżej niż `rwx` i przycina efektywne uprawnienia; sprawdź komentarz `#effective:` w `getfacl`. Po drugie — jeden z katalogów nadrzędnych po drodze do pliku nie ma dla `jkowalski` prawa `x` (wejścia), co blokuje dostęp niezależnie od uprawnień samego pliku — dokładnie mechanizm opisany na stronie o `chmod`/`chown`, sprawdzany komendą `namei -l`.

3. Skopiowałeś katalog przez `rsync -a` zamiast `cp -a`. Czy ACL przetrwa?

Odpowiedź

Samo `-a` w `rsync` (archive) nie obejmuje ACL — trzeba dodatkowo podać `-A` (`rsync -aA`), żeby ACL było przenoszone razem z resztą metadanych. To częsta pomyłka wynikająca z podobieństwa flagi `-a` w `cp` i `rsync`, które w tym akurat zakresie nie działają identycznie.

4. Katalog ma ustawione ACL domyślne z prawem `rwx` dla grupy `deweloperzy`. Ktoś tworzy w nim podkatalog. Czy ten nowy podkatalog też będzie miał ACL domyślne dla `deweloperzy`, czy tylko zwykły ACL?

Odpowiedź

Oba naraz. Nowy podkatalog dziedziczy zwykły ACL (z wpisem `group:deweloperzy:rwx` jako normalnym prawem dostępu) oraz kopię samego ACL domyślnego z katalogu nadrzędnego — dzięki temu mechanizm dziedziczenia propaguje się dalej, do plików i podkatalogów tworzonych już wewnątrz tego nowego podkatalogu, bez konieczności ponownego wywoływania `setfacl -d` na każdym poziomie z osobna.

Powiązane: chmod, chown, chgrp — uprawnienia klasyczne · umask, setuid, setgid, sticky bit · Użytkownicy, grupy i pliki, w których system ich trzyma

Źródła: `man 1 getfacl` · `man 1 setfacl` · `man 5 acl` · `man 8 tune2fs`

Revision #16
Created 2026-09-17 21:42:51 UTC by Tomasz Konieczny
Updated 2026-10-03 19:08:31 UTC by Tomasz Konieczny