Użytkownicy, grupy i pliki, w których system ich trzyma

Poziom: podstawy **Dotyczy: wszystkie dystrybucje** **Czas: ~18 min**

Każdy proces w Linuksie działa jako konkretny użytkownik i konkretna grupa — od tego zależy, co wolno mu odczytać, zapisać i uruchomić. Ta strona pokazuje, gdzie system trzyma te tożsamości i jakimi komendami się nimi zarządza.

Skróty użyte na tej stronie (6)

Skrót	Co znaczy
BSD	Berkeley Software Distribution: rodzina Uniksów z uczelni Berkeley, inna gałąź niż Linux, stąd czasem inna składnia tych samych poleceń
GID	Group Identifier: numer grupy systemowej, drugi obok UID element decydujący o prawach dostępu do pliku
LDAP	Lightweight Directory Access Protocol: protokół do odpytywania katalogu użytkowników (AD, FreeIPA)
SSH	Secure Shell: szyfrowane zdalne logowanie do serwera
UID	User ID: numer użytkownika w Linuksie
WWW	World Wide Web: strony i aplikacje otwierane w przeglądarce po HTTP albo HTTPS

Fiszka

TYLKO ODCZYT

Sprawdzenie własnej tożsamości: id

```
id
```

Bez argumentów pokazujemy tożsamość bieżącej powłoki: UID, GID grupy głównej i pełną listę grup dodatkowych. Sięgamy po nią jako pierwszą komendę, gdy coś odmawia dostępu,

a nie wiadomo, czy problem leży po stronie użytkownika czy grupy.

Przykład:

```
uid=1001(adm_user) gid=1001(adm_user) groups=1001(adm_user),10(wheel),994(docker)
```

UID = User ID: numer użytkownika w Linuksie · **GID** = Group Identifier: numer grupy systemowej, drugi obok UID element decydujący o prawach dostępu do pliku · **id** = pokaż użytkownika, jego numer UID oraz grupy, do których należy

TYLKO ODCZYT

Wypisanie samych nazw grup: id -nG

```
id -nG nazwa_usera
```

`-n` każe wypisać nazwy zamiast numerów, `-G` ogranicza wynik do samych grup (pomija UID). Tego formatu używamy przy sprawdzaniu w skrypcie: `id -nG "$USER" | grep -qw wheel`.

Przykład:

```
adm_user wheel docker
```

UID = User ID: numer użytkownika w Linuksie · **id** = pokaż użytkownika, jego numer UID oraz grupy, do których należy

TYLKO ODCZYT

Sprawdzenie efektywnego użytkownika: whoami

```
whoami
```

Wypisujemy wyłącznie nazwę efektywnego użytkownika procesu. Po `sudo -u inny_user komenda` `whoami` wewnątrz tej komendy pokaże `inny_user`, nawet jeśli fizycznie zalogowany jest ktoś inny.

Przykład:

```
adm_user
```

TYLKO ODCZYT

Lista zalogowanych sesji: who

```
who
```

Wypisujemy listę aktualnie zalogowanych sesji: kto, na jakim terminalu, od kiedy i (przy logowaniu sieciowym) skąd. Dane pochodzą z `/var/run/utmp`, więc dotyczą wyłącznie sesji interaktywnych — proces demona uruchomiony przez systemd tu się nie pojawi.

Przykład:

```
adm_user pts/0        2026-09-17 08:12 (10.20.20.15)
root     pts/1        2026-09-17 07:55 (10.20.20.20)
```

□□ TYLKO ODCZYT

Sprawdzenie grup użytkownika: groups

```
groups nazwa_usera
```

Krótsza, starsza forma `id -nG` wywodząca się z BSD — bez UID-ów, sama nazwa użytkownika i lista grup.

Przykład:

```
adm_user : adm_user wheel docker
```

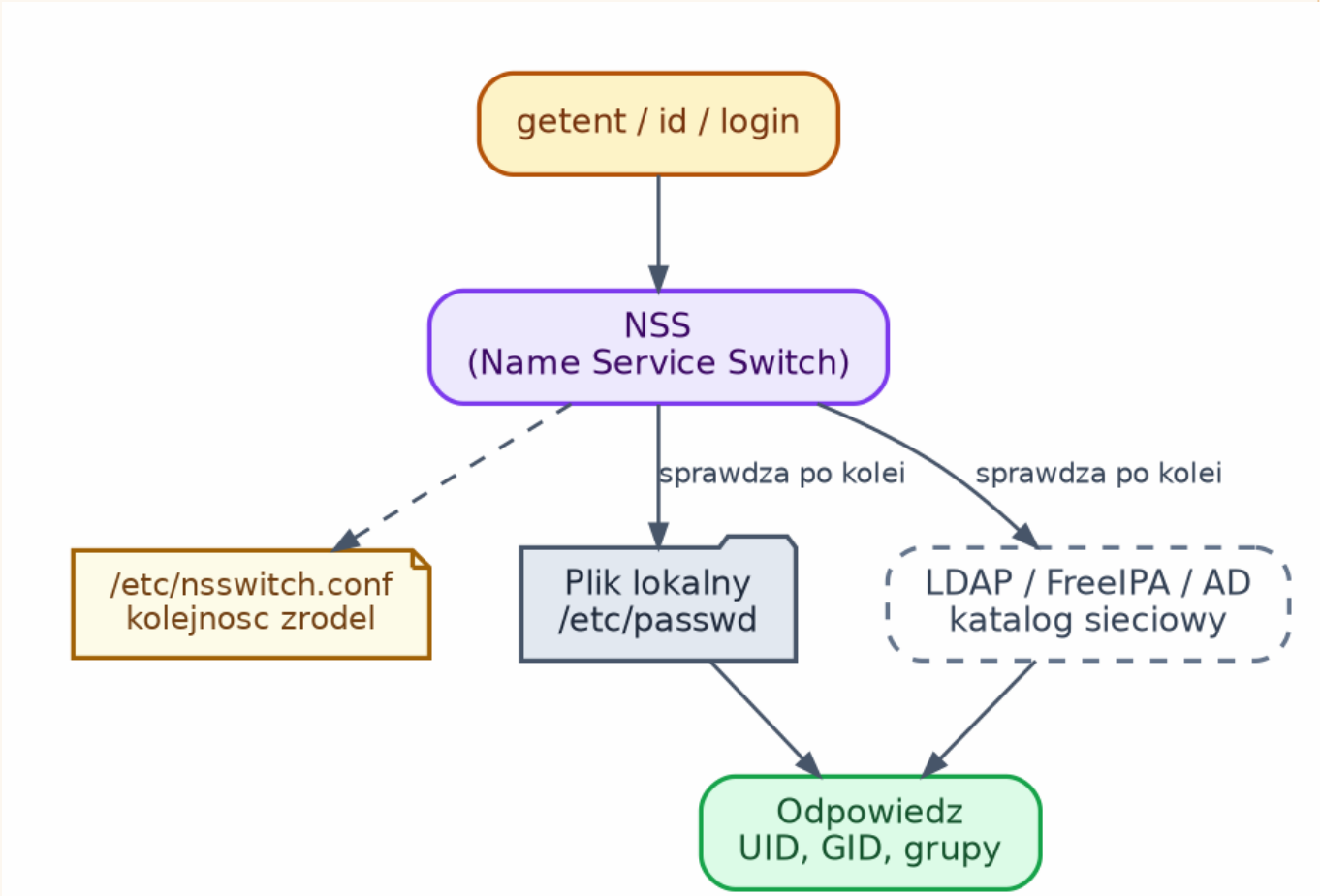
□□ **BSD** = Berkeley Software Distribution: rodzina Uniksów z uczelni Berkeley, inna gałąź niż Linux, stąd czasem inna składnia tych samych poleceń

□□ ZMIENIA SYSTEM

Odpytanie NSS o konto: getent passwd

```
getent passwd nazwa_usera
```

Odpytujemy bazę użytkowników przez NSS (Name Service Switch) — mechanizm, który pośredniczy między komendami systemowymi a źródłem danych o kontach: lokalnym plikiem, LDAP-em, FreeIPA albo Active Directory. `getent` widzi konto niezależnie od tego, skąd naprawdę pochodzi; `grep nazwa_usera /etc/passwd` widzi wyłącznie plik lokalny i przy koncie domenowym nie zwróci nic, mimo że konto realnie istnieje i działa.



NSS odpytuje po kolei źródła wskazane w konfiguracji: najpierw zwykle plik lokalny, potem katalog sieciowy, i zwraca pierwszą pasującą odpowiedź (kliknięcie otwiera pełny rozmiar)

Przykład:

```
adm_user:x:1001:1001:Jan Kowalski,,,:/home/adm_user:/bin/bash
```

getent = get entries: odpytaj bazę systemową (użytkowników, grup, hostów) niezależnie od tego, czy to plik lokalny czy AD

TYLKO ODCZYT

Odpytanie NSS o grupę: getent group

```
getent group wheel
```

To samo dla grup — pokazujemy GID i listę członków dodatkowych (nie licząc tych, dla których to jest grupa główna, bo ci w polu listy członków się nie pojawiają).

Przykład:

```
wheel:x:10:adm_user,drugi_user
```

GID = Group Identifier: numer grupy systemowej, drugi obok UID element decydujący o prawach dostępu do pliku · **getent** = get entries: odpytaj bazę systemową (użytkowników, grup, hostów) niezależnie od tego, czy to plik lokalny czy AD

ZMIENIA SYSTEM

Założenie konta: useradd

```
useradd -m -s /bin/bash -c "Jan Kowalski" nazwa_usera
```

Zakładamy nowe konto. `-m` tworzy katalog domowy i kopiuje do niego zawartość `/etc/skel`, `-s` ustawia powłokę logowania, `-c` to pole komentarza (GECOS), zwyczajowo pełne imię i nazwisko. Sama komenda niczego nie wypisuje przy powodzeniu — efekt od razu weryfikujemy przez `getent passwd nazwa_usera`. To, czy `useradd` bez `-m` utworzy katalog domowy, zależy od ustawienia `CREATE_HOME` w `/etc/login.defs` danej instalacji — dlatego `-m` podajemy zawsze jawnie, zamiast polegać na domyślnym zachowaniu dystrybucji.

useradd = user add: tworzy nowe konto systemowe w Linuksie

ZMIENIA SYSTEM

Założenie konta usługowego: useradd -r

```
useradd -r -s /sbin/nologin -d /var/lib/NazwaUslugi NazwaUslugi
```

`-r` (system account) bierze UID z osobnej, niższej puli — poniżej `UID_MIN` z `/etc/login.defs`, zwykle < 1000 — i nie zakłada hasła. W parze z powłoką `/sbin/nologin` powstaje typowe konto do uruchamiania demona: ma swój UID do rozliczania plików i procesów, ale nikt nie zaloguje się nim interaktywnie.

UID = User ID: numer użytkownika w Linuksie · **useradd** = user add: tworzy nowe konto systemowe w Linuksie

ZMIENIA SYSTEM

Dopisanie do grupy dodatkowej: usermod -aG

```
usermod -aG docker nazwa_usera
```

Dopisujemy grupę `docker` do listy grup dodatkowych użytkownika, nie ruszając pozostałych wpisów. Flaga `-a` (append) jest tu kluczowa — jej pominięcie to jeden z najczęstszych błędów administracyjnych, opisany niżej w *Częste błędy*.

usermod = zmień ustawienia istniejącego konta systemowego, na przykład przynależność do grupy

OSTROŻNIE · NIEODWRACALNE

Usunięcie konta razem z danymi: userdel -r

```
userdel -r nazwa_usera
```

Usuwamy konto razem z katalogiem domowym i skrzynką pocztową (`-r` = remove). Bez `-r` wpis znika z `/etc/passwd`, ale pliki zostają na dysku, przypisane do UID-u, który już niczemu nie odpowiada — `ls -l` zacznie pokazywać przy nich gołą liczbę zamiast nazwy.

TYLKO ODCZYT

Utworzenie grupy: groupadd

```
groupadd NazwaGrupy
```

Tworzymy nową grupę. Kolejny wolny GID system bierze automatycznie z puli `>= GID_MIN`, zdefiniowanej w `/etc/login.defs`; jawny numer można wymusić przez `-g`.

GID = Group Identifier: numer grupy systemowej, drugi obok UID element decydujący o prawach dostępu do pliku · **groupadd** = group add: zakłada grupę w systemie, z opcją --gid o wybranym numerze, co ułatwia takie same numery na wielu maszynach

TYLKO ODCZYT

Dopisanie użytkownika do grupy: gpasswd -a

```
gpasswd -a nazwa_usera NazwaGrupy
```

Alternatywa dla `usermod -aG`, wygodniejsza, gdy dopisujemy jednego użytkownika do jednej grupy: `gpasswd` operuje na tej jednej grupie, więc nie ma tu ryzyka nadpisania całej listy — pułapka braku `-a` z `usermod` po prostu nie istnieje.

⚠ ZMIENIA SYSTEM

Zablokowanie hasła: passwd -l

```
passwd -l nazwa_usera
```

Blokujemy logowanie hasłem, wstawiając `!` przed hashem w `/etc/shadow`. To nie jest to samo co wyłączenie konta — logowanie kluczem SSH albo przez inny mechanizm uwierzytelniania, który nie pyta o hasło, nadal działa.

Przykład:

```
Zablokowanie hasła użytkownika jkowalski.
passwd: operacja powiodła się
```

⚠ **SSH** = Secure Shell: szyfrowane zdalne logowanie do serwera

⚠ ZMIENIA SYSTEM

Sprawdzenie stanu konta: passwd -S

```
passwd -S nazwa_usera
```

Status konta w jednej linii: nazwa, stan hasła (`L` zablokowane, `P` używalne, `NP` brak hasła), data ostatniej zmiany, minimalny i maksymalny wiek hasła, okres ostrzeżenia, okres nieaktywności po wygaśnięciu.

Przykład:

```
jkowalski L 09/10/2026 0 99999 7 -1
```

⚠ TYLKO ODCZYT

Pełna sesja logowania a samo przełączenie UID: su - / su

```
su - nazwa_usera
```

`-` (albo `-l`) uruchamia pełną sesję logowania: czyta pliki startowe powłoki docelowego użytkownika, ustawia jego `HOME`, `PWD` i zmienne środowiskowe tak, jakby zalogował się od nowa.

```
su nazwa_usera
```

Bez `-` zmienia tylko UID/GID bieżącego procesu, zachowując dotychczasowe środowisko — `$HOME` i `$PATH` nadal wskazują na poprzedniego użytkownika, co bywa źródłem mylących błędów typu „mam prawa jkowalskiego, ale otwiera mi pliki configi poprzedniego konta”.

UID = User ID: numer użytkownika w Linuksie · **su** = substitute user: przełącz się na innego użytkownika

TYLKO ODCZYT

Tymczasowa zmiana grupy głównej: newgrp

```
newgrp NazwaGrupy
```

Startujemy nową powłokę, w której podana grupa staje się grupą główną bieżącej sesji, bez wylogowania i ponownego logowania. Efekt jest tymczasowy i kończy się razem z tą powłoką (`exit`).

newgrp = przełącz się na inną grupę bez wylogowania z bieżącej sesji

TYLKO ODCZYT

Zbiorczy raport o kontach: lslogins

```
lslogins -u
```

Zestawiamy w jednej tabeli dane z `/etc/passwd`, `/etc/shadow`, `/var/log/lastlog` i listy grup — szybki przegląd wszystkich kont zwykłych (`-u`) bez ręcznego łączenia kilku plików.

Przykład:

```
UID USER     PWD-LOCK PWD-DENY LAST-LOGIN  GECOS
1001 adm_user 0        0        2026-09-17  Jan Kowalski
1002 aplikacja 1        1        [nigdy]
```

TYLKO ODCZYT

Wszystkie komendy

Komenda	Do czego służy
`id` / `id -nG`	tożsamość bieżącego (albo wskazanego) procesu
`whoami`	sama nazwa efektywnego użytkownika
`who`	kto jest zalogowany w sesjach interaktywnych
`groups`	grupy danego użytkownika, format BSD
`getent passwd` / `getent group`	zapytanie do NSS — działa też dla kont domenowych
`useradd`, `useradd -r`	założenie konta zwykłego / usługowego
`usermod -aG`	dopisanie do grupy dodatkowej (z `-a`!)
`userdel -r`	usunięcie konta razem z danymi
`groupadd`, `gpasswd -a`	utworzenie grupy, dopisanie do niej członka
`passwd -l`, `passwd -S`	zablokowanie hasła, sprawdzenie stanu konta
`su -` / `su`	pełna sesja logowania kontra samo przełączenie UID
`newgrp`	tymczasowa zmiana grupy głównej sesji
`lslogins`	zbiorczy raport o kontach

Uważaj

- `usermod -G` (bez `-a`) nadpisuje całą listę grup dodatkowych — patrz *Częste błędy*.
- `grep nazwa_usera /etc/passwd` nie zobaczy konta domenowego z LDAP /FreeIPA/AD; do tego służy `getent`.
- `passwd -l` nie blokuje logowania kluczem SSH.

LDAP = Lightweight Directory Access Protocol: protokół do odpytywania katalogu użytkowników (AD, FreeIPA) · **SSH** = Secure Shell: szyfrowane zdalne logowanie do serwera

Z życia wzięte

Z ŻYCIA WZIĘTE

O 9:15 Tomek dopisuje nowego programistę do grupy docker na docker01 komendą `usermod -G docker`, nie zauważając że nadpisuje nią całą listę grup dodatkowych. Po południu Ola zgłasza, że kolega stracił dostęp do sudo, bo zniknął z grupy wheel bez żadnego komunikatu o błędzie. Tomek naprawia to jednym `usermod -aG wheel` i od tej pory sprawdza `id -nG` przed

> i po każdej takiej zmianie.

Po co to istnieje

> **Pojęcie: użytkownik systemowy kontra zwykły.** Linux nie rozróżnia kont technicznie inaczej niż numerem UID, ale konwencyjnie dzieli je na dwie kategorie. Użytkownik zwykły ma UID z górnej puli (zwykle od 1000 wzwyż), hasło, powłokę logowania i katalog domowy pod `/home` — to konto dla człowieka. Użytkownik systemowy (service account) ma UID z dolnej puli (poniżej `UID_MIN`, historycznie < 500 albo < 1000, zależnie od dystrybucji), zwykle bez hasła i z powłoką blokującą logowanie interaktywne — istnieje wyłącznie po to, żeby jakiś demon miał własną tożsamość do rozliczania plików i procesów, oddzieloną od roota i od kont ludzi.

Bez tego rozróżnienia każdy proces w systemie musiałby działać jako root albo jako przypadkowy człowiek — obie opcje są złe z punktu widzenia bezpieczeństwa. Baza danych, serwer WWW czy kolejka komunikatów dostają własne konto systemowe, żeby awaria albo włamanie na ten proces nie dawało od razu pełnych praw roota ani dostępu do plików prawdziwego pracownika.

Analogia

System kont użytkowników działa jak **recepcja w dużym biurowcu z identyfikatorami dostępu**.

Każda osoba wchodząca do budynku dostaje kartę z unikalnym numerem — to jej UID. Karta otwiera konkretne drzwi, bo jest przypisana do zespołów, do których ta osoba należy — to grupy. Jedna karta może otwierać drzwi kilku działów naraz (grupy dodatkowe), ale ma dokładnie jeden dział wypisany na przodzie identyfikatora jako główny (grupa główna) — ten, który recepcjonista poda pierwszy, gdy ktoś zapyta „z jakiego pan jest zespołu”.

Są też karty serwisowe — dla firmy sprzątającej, dla dostawcy automatów z kawą. Mają dostęp tylko tam, gdzie muszą, i nikt nie próbuje nimi „zalogować się” jako pracownik na stałe — to odpowiednik kont systemowych z powłoką `/sbin/nologin`. Karta zablokowana (`passwd -l`) przestaje działać na czytniku przy drzwiach głównych, ale zapasowy klucz mechaniczny do bocznego wejścia (klucz SSH) nadal otwiera drzwi — blokada karty go nie dotyczy.

Jak to działa

Rozwiń: jak to działa pod spodem

> **Pojęcie: UID i GID.** UID (user identifier) to liczba, która jednoznacznie identyfikuje użytkownika dla jądra systemu — nazwa `adm_user` jest wyłącznie etykietą dla ludzi, jądro operuje na liczbie. GID (group identifier) to analogiczna liczba dla grupy. Każdy plik na dysku ma zapisany numeryczny UID właściciela i numeryczny GID grupy, nie nazwy — dlatego usunięcie konta bez usunięcia jego plików (`userdel` bez `-r`) zostawia pliki „przypisane” do liczby, która nagle niczego nie znaczy.
>
> **Pojęcie: grupa główna kontra dodatkowa.** Każdy użytkownik ma dokładnie jedną grupę główną (primary group) — jej GID trafia do nowo tworzonych przez niego plików, jeśli nic innego tego nie zmieni (patrz kolejna strona o `umask`). Poza tym użytkownik może należeć do dowolnej liczby grup dodatkowych (supplementary groups), które nie wpływają na domyślnego właściciela nowych plików, ale dają dostęp do wszystkiego, co jest udostępnione tej grupie — stąd dopisywanie do grupy `docker` czy `wheel` przez `usermod -aG`, a nie zmiana grupy głównej.

Budowa `/etc/passwd`, plik po polu, na przykładzie wpisu:

```
adm_user:x:1001:1001:Jan Kowalski,,,:/home/adm_user:/bin/bash
```

1. `adm_user` — login, nazwa którą wpisujesz przy logowaniu.
2. `x` — historyczne miejsce na hash hasła; dziś zawsze `x`, prawdziwy hash mieszka w `/etc/shadow`.
3. `1001` — UID.
4. `1001` — GID grupy głównej.
5. `Jan Kowalski,,,` — pole GECOS: pełne imię i nazwisko, opcjonalnie numer pokoju, telefon służbowy, telefon domowy, rozdzielone przecinkami.
6. `/home/adm_user` — katalog domowy.
7. `/bin/bash` — powłoka uruchamiana po zalogowaniu.

Budowa `/etc/shadow` dla tego samego konta:

```
adm_user:$6$rounds=656000$abc...:19980:0:99999:7:::
```

1. `adm_user` — login, klucz łączący z `/etc/passwd`.
2. `$6$rounds=656000$abc...` — hash hasła; prefiks `$6$` oznacza algorytm SHA-512.
3. `19980` — dzień ostatniej zmiany hasła, liczony od 1 stycznia 1970 (epoka uniksowa).
4. `0` — minimalny wiek hasła w dniach (0 = można zmienić w każdej chwili).

5. `99999` — maksymalny wiek hasła w dniach (99999 w praktyce oznacza „nigdy nie wygasa”).
6. `7` — ile dni przed wygaśnięciem system zacznie ostrzegać.
7. (puste) — ile dni po wygaśnięciu konto jest jeszcze nieaktywne, zanim zostanie zablokowane.
8. (puste) — data wygaśnięcia całego konta (nie tylko hasła), w dniach od epoki.
9. (puste, zarezerwowane) — pole na przyszłe rozszerzenia formatu.

Dlaczego `/etc/shadow` to osobny plik. `/etc/passwd` musi być czytelny dla wszystkich (`-rw-r--r--`), bo dziesiątki komend systemowych — od `ls -l` po `id` — muszą umieć zamienić UID na nazwę i odwrotnie, dla dowolnego procesu. Gdyby hash hasła siedział w tym samym pliku, każdy zalogowany użytkownik mógłby go odczytać i próbować złamać offline. Rozdzielenie danych w 1988 roku (shadow password suite) rozwiązało ten problem: `/etc/shadow` ma prawa `-rw-------` (albo `-r--------` z grupą `shadow` na niektórych dystrybucjach Debian/Ubuntu) i jest czytelny wyłącznie przez roota — albo procesy z odpowiednim setuid, jak `passwd` (mechanizm setuid opisuje kolejna strona rozdziału).

Powłoki blokujące logowanie: `/sbin/nologin` i `/bin/false`. Obie odmawiają uruchomienia interaktywnej sesji, ale różnią się zachowaniem. `/sbin/nologin` wypisuje komunikat w stylu „This account is currently not available.” i loguje próbę logowania, po czym kończy proces z kodem 1 — jaśniejsza informacja dla kogoś, kto próbował się zalogować przez pomyłkę. `/bin/false` po prostu kończy działanie z kodem 1, bez żadnego komunikatu — bardziej surowe, historycznie starsze rozwiązanie. Żadna z tych powłok nie blokuje dostępu przez mechanizmy, które nie uruchamiają powłoki logowania wprost — np. `rsync` czy wymuszone polecenie (`ForceCommand`) w konfiguracji SSH nadal mogą działać na takim koncie.

Krok po kroku

1. Załóż grupę projektową i konto usługowe.

```
groupadd projektxyz
useradd -r -s /sbin/nologin -g projektxyz -d /var/lib/projektxyz projektxyz
```

2. Sprawdź, co naprawdę powstało.

```
getent passwd projektxyz
getent group projektxyz
```

```
projektxyz:x:987:987::/var/lib/projektxyz:/sbin/nologin
projektxyz:x:987:
```

UID/GID 987 — z dolnej puli, jak przystało na konto systemowe.

3. Dopisz swoje konto administracyjne do grupy, żeby móc zarządzać danymi usługi.

```
usermod -aG projektxyz adm_user
id -nG adm_user
```

```
adm_user wheel docker projektxyz
```

4. Zablokuj hasło (i tak go nie ma, ale dla porządku i jasnego stanu w `passwd -S`).

```
passwd -l projektxyz
passwd -S projektxyz
```

```
projektxyz L 09/17/2026 0 99999 7 -1
```

Częste błędy

Nadpisanie grup dodatkowych przez `usermod -G` bez `-a`.

```
usermod -G docker adm_user
id -nG adm_user
```

```
adm_user docker
```

Brak błędu na ekranie — to najgroźniejszy typ pomyłki. `adm_user` był w grupach `wheel` i `docker`, a po tej komendzie zostaje wyłącznie w `docker`; przynależność do `wheel` (a z nią prawo do `sudo`) znika po cichu, dopóki ktoś nie spróbuje użyć `sudo` i nie dostanie odmowy. Naprawa: zawsze `-aG`, nigdy samo `-G`, chyba że naprawdę chcesz zastąpić całą listę.

Mylenie `su` z `su -`.

```
su jkowalski
echo $HOME
```

```
/home/adm_user
```

Powłoka działa już jako `jkowalski`, ale `$HOME` nadal wskazuje na poprzedniego użytkownika, bo `su` bez `-` nie czyta plików startowych docelowego konta. Skutek uboczny: skrypty odwołujące się do `$HOME` czytają albo zapisują nie tam, gdzie powinny. Naprawa: `su - jkowalski`.

`userdel` bez `-r` zostawia osierocone pliki.

```
userdel jkowalski
ls -l /home/jkowalski
```

```
drwx------ 15 1005 1005 4096 wrz 10 09:00 /home/jkowalski
```

UID `1005` już nikomu nie odpowiada — `ls -l` pokazuje gołą liczbę zamiast nazwy. Pliki fizycznie zostają, zajmują miejsce i nikt formalnie nie jest ich właścicielem. Naprawa: `userdel -r`, albo ręczne dociągnięcie `rm -rf /home/jkowalski`, jeśli konto już usunięto bez tej flagi.

Usunięcie konta, którego procesy wciąż działają.

```
userdel aplikacja
```

```
userdel: user aplikacja is currently used by process 4821
```

`userdel` odmawia usunięcia konta, dopóki jakiś proces działa na jego prawach. Naprawa: najpierw zatrzymać usługę (`systemctl stop aplikacja`), dopiero potem usuwać konto.

Przekonanie, że `passwd -l` w pełni wyłącza konto.

```
passwd -l projektxyz
ssh -i klucz_projektxyz.pem projektxyz@srv01
```

Logowanie kluczem SSH się udaje, mimo zablokowanego hasła — `passwd -l` blokuje wyłącznie ścieżkę uwierzytelniania hasłem, nie samo konto. Do pełnego wyłączenia logowania służy ustawienie powłoki na `/sbin/nologin` (`usermod -s /sbin/nologin`) albo data wygaśnięcia konta przez `chage -E 0`.

□□ Ćwiczenia

> Wszystko poniżej wykonujesz na własnej maszynie, w katalogu `/tmp`. Odczyt `/etc/passwd` nie wymaga uprawnień administratora; próba odczytu `/etc/shadow` w Ćwiczeniu 1 celowo kończy się odmową, jeśli nie jesteś rootem — to jest właśnie zamierzony efekt.

Ćwiczenie 1 — dlaczego `/etc/shadow` jest niewidoczny

```
cd /tmp
id -u
ls -l /etc/passwd /etc/shadow
cat /etc/shadow 2>&1 | tee shadow_probe.txt
```

Porównaj uprawnienia obu plików w wyniku `ls -l` i sprawdź, co trafiło do `shadow_probe.txt`.

Odpowiedź

```
-rw-r--r--. 1 root root  2847 wrz 10 09:00 /etc/passwd
-rw-------. 1 root root  1523 wrz 10 09:00 /etc/shadow
cat: /etc/shadow: Permission denied
```

`/etc/passwd` jest czytelny dla wszystkich, `/etc/shadow` — tylko dla roota. Jeśli uruchamiasz to jako zwykły użytkownik (`id -u` różne od `0`), plik `shadow_probe.txt` zawiera właśnie ten komunikat o odmowie dostępu, a nie żadną treść pliku. To jest dokładnie mechanizm opisany w sekcji *Jak to działa* — rozdzielenie danych publicznych od hashy haseł.

Ćwiczenie 2 — kto może się zalogować interaktywnie

```
cd /tmp
awk -F: '$7 !~ /(nologin|false)/ {print $1, $3, $7}' /etc/passwd > logowalni.txt
cat logowalni.txt
wc -l logowalni.txt
```

Ile kont z Twojego `/etc/passwd` ma powłokę inną niż `/sbin/nologin` i `/bin/false`? Czy to naprawdę tyle kont, które da się zalogować, ile jest w Twoim systemie ludzi?

Odpowiedź

Zwykle więcej niż liczba prawdziwych osób — część kont systemowych (np. konta demonów tworzonych przez pakiety) dostaje przy instalacji powłokę `/bin/sync` albo w ogóle nie ma jej ustawionej jawnie, a filtr `awk` łapie tylko dwie najpopularniejsze powłoki blokujące. Pełny audyt wymaga też sprawdzenia `passwd -S` dla każdego konta (`L`/`NP` vs `P`), bo powłoka i stan hasła to dwie niezależne blokady logowania.

Pytania kontrolne

1. Czym różni się grupa główna od grupy dodatkowej?

Odpowiedź

Grupa główna jest dokładnie jedna na użytkownika i jej GID trafia domyślnie do każdego nowego pliku, który ten użytkownik utworzy (o ile nic tego nie zmieni — patrz `umask` i `setgid` na kolejnych stronach). Grup dodatkowych może być dowolnie wiele; dają dostęp do zasobów udostępnionych tym grupom, ale nie wpływają na właściciela nowo tworzonych plików.

2. Dlaczego `useradd -r` nie ustawia hasła użytkownikowi?

Odpowiedź

Konto usługowe (`-r`, system account) jest przeznaczone do uruchamiania procesów, nie do logowania interaktywnego — nikt nie ma go wpisywać przy ekranie logowania. Brak hasła (pole w `/etc/shadow` ustawione na `!` albo `*`) uniemożliwia zalogowanie tą drogą, a powłoka `/sbin/nologin` dodatkowo blokuje próbę uruchomienia sesji, gdyby ktoś jednak próbował.

3. Masz plik należący do UID `1005`, ale `getent passwd 1005` nic nie zwraca. Co się prawdopodobnie stało?

Odpowiedź

Konto o tym UID zostało usunięte — najpewniej przez `userdel` bez flagi `-r` — więc plik pozostał na dysku z numerem, który nie odpowiada już żadnemu wpisowi w `/etc/passwd` (ani w żadnym innym źródle NSS). `ls -l` pokaże przy takim pliku gołą liczbę zamiast nazwy.

4. Zalogowałeś się jako `adm_user`, wykonałeś `su jkowalski`, a potem `cat plik_z_configiem`. Plik otworzył się z innej lokalizacji niż oczekiwałeś. Dlaczego?

Odpowiedź

`su` bez `-` nie czyta plików startowych docelowego użytkownika i nie zmienia `$HOME` ani `$PWD` — proces nadal ma środowisko `adm_user`, tylko efektywny UID/GID zmienił się na `jkowalski`. Ścieżka względna w komendzie została więc rozwiązana względem starego katalogu roboczego. Rozwiązanie: `su - jkowalski`, żeby dostać pełną sesję logowania z poprawnym środowiskiem.

Powiązane: chmod, chown, chgrp — uprawnienia klasyczne · umask, setuid, setgid, sticky bit · ACL — uprawnienia ponad model właściciel-grupa-inni

Źródła: `man 5 passwd` · `man 5 shadow` · `man 8 useradd` · `man 1 su` · `man 1 lslogins`

Revision #16
Created 2026-09-17 21:42:46 UTC by Tomasz Konieczny
Updated 2026-10-03 19:08:26 UTC by Tomasz Konieczny