Hasła i wygasanie kont: chage, passwd, faillock

Poziom: średni **Dotyczy: RHEL/Rocky 9–10, różnice dla Debian/Ubuntu i SLES zaznaczone**

Czas: ~16 min

`chage` i `passwd` sterują dwiema osobnymi rzeczami, które łatwo pomylić: kiedy hasło przestaje być ważne i kiedy konto przestaje działać w ogóle. `faillock` dokłada trzeci, niezależny mechanizm — tymczasową blokadę po serii nieudanych logowań, niezwiązaną z żadnym z powyższych.

Skróty użyte na tej stronie (4)

Skrót	Co znaczy
PAM	Pluggable Authentication Modules: wymienne moduły logowania w Linuksie, decydują na przykład o sile wymaganego hasła
RHEL	Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa
SLES	SUSE Linux Enterprise Server: płatna dystrybucja Linuksa firmy SUSE, konkurent Rocky/RHEL
SSH	Secure Shell: szyfrowane zdalne logowanie do serwera

Fiszka

TYLKO ODCZYT

Sprawdzenie pełnego stanu wygasania konta: chage -l

```
sudo chage -l nazwa_usera
```

Wypisujemy komplet informacji o cyklu życia hasła danego użytkownika: kiedy było ostatnio zmienione, kiedy wygasa, kiedy konto stanie się nieaktywne i kiedy wygasa samo konto. Zwykły użytkownik może uruchomić to samo bez `sudo`, ale tylko dla samego siebie.

Przykład:

```
Ostatnia zmiana hasła                                  : wrz 02, 2026
Hasło wygasa                                           : gru 01, 2026
Hasło nieaktywne                                       : gru 08, 2026
Konto wygasa                                           : nigdy
Minimalna liczba dni pomiędzy zmianami hasła           : 1
Maksymalna liczba dni pomiędzy zmianami hasła          : 90
Liczba dni ostrzeżenia przed wygaśnięciem hasła        : 7
```

□□ TYLKO ODCZYT

Ustawienie maksymalnego wieku hasła: chage -M

```
sudo chage -M 90 nazwa_usera
```

Ustawiamy liczbę dni, po których hasło wygasa i wymusza zmianę przy następnym logowaniu. Nowa wartość liczy się od daty ostatniej zmiany hasła zapisanej w `/etc/shadow`, nie od dnia wykonania tej komendy — jeśli hasło było zmienione dawno, nowy limit może już być przekroczony.

□□ TYLKO ODCZYT

Ustawienie minimalnego wieku hasła: chage -m

```
sudo chage -m 1 nazwa_usera
```

Ustawiamy minimalną liczbę dni, jakie muszą minąć od ostatniej zmiany hasła, zanim użytkownik będzie mógł zmienić je ponownie. Bez tego ograniczenia użytkownik zmuszony do okresowej zmiany hasła mógłby obejść politykę, zmieniając je dziesięć razy z rzędu w ciągu minuty i wracając do starego.

□□ TYLKO ODCZYT

Ustawienie liczby dni ostrzeżenia: chage -W

```
sudo chage -W 7 nazwa_usera
```

Liczba dni przed wygaśnięciem hasła, w których logowanie zaczyna wyświetlać ostrzeżenie o zbliżającym się terminie zmiany — użytkownik dostaje szansę zmienić hasło spokojnie, zamiast zostać zaskoczonym wymuszoną zmianą w środku pracy.

🔒 TYLKO ODCZYT

Ustawienie daty wygaśnięcia konta: chage -E

```
sudo chage -E 2026-12-31 nazwa_usera
```

Ustawiamy sztywną datę, po której konto przestaje działać **niezależnie od stanu hasła** — nawet jeśli hasło jest świeże i ważne. Typowe zastosowanie: konto kontraktora z określonym z góry terminem współpracy. `chage -E -1` usuwa datę wygaśnięcia (konto znów nie wygasa nigdy).

🔒 TYLKO ODCZYT

Ustawienie liczby dni nieaktywności po wygaśnięciu hasła: chage -I

```
sudo chage -I 10 nazwa_usera
```

Po wygaśnięciu hasła użytkownik zwykle dostaje jeszcze okno na jego zmianę. `-I` określa, ile dni to okno trwa — po jego przekroczeniu bez zmiany hasła konto zostaje automatycznie zablokowane, tak jakby ktoś wykonał `passwd -l`.

🔒 TYLKO ODCZYT

Wymuszenie zmiany hasła przy następnym logowaniu: chage -d 0

```
sudo chage -d 0 nazwa_usera
```

Ustawiamy datę ostatniej zmiany hasła na najwcześniejszą możliwą (epoka Uniksa), co w połączeniu z jakimkolwiek `-M` natychmiast czyni hasło „wygasłym”. Klasyczne zastosowanie: nowe konto z tymczasowym hasłem, które administrator zna, a użytkownik ma obowiązek zmienić przy pierwszym logowaniu.

Przykład:

```
$ ssh user1@srv01
Twoje hasło wygasło.
Musisz je teraz zmienić (wymagane przez administratora).
Bieżące hasło:
```

⚠ ZMIENIA SYSTEM

Sprawdzenie statusu hasła w jednej linii: passwd -S

```
sudo passwd -S nazwa_usera
```

Skrótowy odpowiednik `chage -l` w jednym wierszu, wygodny do skryptów. Drugie pole to kod statusu: `P` (usable password — hasło ustawione i aktywne), `L` (locked — zablokowane), `NP` (no password — brak hasła w ogóle).

Przykład:

```
user1 P 2026-09-02 1 90 7 -1
```

⚠ ZMIENIA SYSTEM

Zablokowanie hasła: passwd -l

```
sudo passwd -l nazwa_usera
```

Dopisujemy wykrzyknik na początku pola z hashem hasła w `/etc/shadow`, przez co żadne podane hasło nigdy nie dopasuje się do zapisanego wzorca — logowanie hasłem przestaje działać. Nic więcej się nie zmienia: konto nadal istnieje, powłoka i katalog domowy zostają nietknięte.

⚠ ZMIENIA SYSTEM

Odblokowanie hasła: passwd -u

```
sudo passwd -u nazwa_usera
```

Odwracamy `passwd -l`, usuwając wykrzyknik z pola hasła. **Uwaga:** jeśli hasło było puste przed zablokowaniem, odblokowanie może przywrócić konto z pustym hasłem — warto sprawdzić `passwd -S` po tej operacji.

⚠ ZMIENIA SYSTEM

Wymuszenie natychmiastowego wygaśnięcia hasła: passwd -e

```
sudo passwd -e nazwa_usera
```

Funkcjonalny odpowiednik `chage -d 0` — wymuszamy zmianę hasła przy najbliższym logowaniu. W przeciwieństwie do `passwd -l` **nie blokuje** logowania — użytkownik nadal może się zalogować, tylko od razu zostaje poproszony o nowe hasło.

□□ TYLKO ODCZYT

Sprawdzenie historii nieudanych logowań: faillock --user

```
sudo faillock --user nazwa_usera
```

Pokazujemy licznik nieudanych prób logowania danego użytkownika prowadzony przez moduł PAM `pam_faillock`, razem z czasem każdej próby i adresem/terminalem, z którego przyszła.

Przykład:

```
user1:
When                Type  Source                                           Valid
2026-09-17 09:41:02 SSH   203.0.113.23                                     V
2026-09-17 09:41:15 SSH   203.0.113.23                                     V
2026-09-17 09:41:29 SSH   203.0.113.23                                     V
```

□□ **PAM** = Pluggable Authentication Modules: wymienne moduły logowania w Linuksie, decydują na przykład o sile wymaganego hasła

□□ TYLKO ODCZYT

Wyzerowanie licznika nieudanych logowań: faillock --reset

```
sudo faillock --user nazwa_usera --reset
```

Czyścimy licznik nieudanych prób i, jeśli konto zostało tymczasowo zablokowane przez `pam_faillock`, natychmiast je odblokowujemy. To zupełnie inna blokada niż `passwd -l` — dotyczy licznika prób, nie samego hasła.

□□ TYLKO ODCZYT

Sprawdzenie domyślnych parametrów dla nowych kont: /etc/login.defs

```
grep -E '^(PASS_MAX_DAYS|PASS_MIN_DAYS|PASS_WARN_AGE)' /etc/login.defs
```

Plik z wartościami domyślnymi, które `useradd` kopiuje do `/etc/shadow` **w momencie tworzenia** nowego konta. Zmiana tego pliku nie ma wpływu na konta, które już istnieją — dotyczy wyłącznie kont utworzonych po zmianie.

Przykład:

```
PASS_MAX_DAYS   90
PASS_MIN_DAYS   1
PASS_WARN_AGE   7
```

grep = global regular expression print: szukaj wzorca w tekście

TYLKO ODCZYT

Sprawdzenie reguł złożoności hasła: pwquality.conf

```
cat /etc/security/pwquality.conf
```

Konfiguracja modułu PAM `pam_pwquality`, który sprawdza nowe hasło w momencie jego ustawiania — minimalną długość, liczbę wymaganych cyfr, wielkich liter, znaków specjalnych i tak dalej. To jest miejsce, które odpowiada za komunikat „hasło zbyt proste” przy próbie ustawienia słabego hasła.

Przykład:

```
minlen = 12
dcredit = -1
ucredit = -1
lcredit = -1
ocredit = -1
retry = 3
```

Ujemna wartość (np. `dcredit = -1`) oznacza „wymagany co najmniej jeden znak tego typu”; wartość dodatnia działałaby jako premia skracająca wymaganą długość — w praktyce na serwerach produkcyjnych używa się niemal wyłącznie wartości ujemnych.

PAM = Pluggable Authentication Modules: wymienne moduły logowania w Linuksie, decydują na przykład o sile wymaganego hasła · **cat** = concatenate: sklej i wypisz pliki

TYLKO ODCZYT

Sprawdzenie nieudanych logowań w systemie: lastb

```
sudo lastb -n 5
```

Odczytujemy `/var/log/btmp` — dziennik **nieudanych** prób logowania w całym systemie, nie tylko dla jednego użytkownika. Szerszy obraz niż `faillock`, bo obejmuje też próby na nieistniejące konta.

Przykład:

```
user1    ssh:notty   203.0.113.23     wt wrz 17 09:41 - 09:41  (00:00)
root     ssh:notty   203.0.113.45     wt wrz 17 03:12 - 03:12  (00:00)
admin    ssh:notty   203.0.113.45     wt wrz 17 03:11 - 03:11  (00:00)
```

TYLKO ODCZYT

Sprawdzenie udanych logowań w systemie: last

```
last -n 5 nazwa_usera
```

Odczytujemy `/var/log/wtmp` — historię **udanych** logowań, wylogowań i restartów systemu. Zestawienie `last` z `lastb` dla tego samego okresu szybko pokazuje, czy seria nieudanych prób poprzedzała udane włamanie, czy skończyła się na samych odrzuceniach.

Przykład:

```
user1    pts/0        10.20.20.15      wt wrz 17 08:55   still logged in
user1    pts/1        10.20.20.15      pon wrz 16 17:02 - 17:45  (00:43)
```

Uważaj

- `chage -M` liczy nowy termin od daty ostatniej zmiany hasła, nie od dziś — warto sprawdzić `chage -l` po zmianie, żeby zobaczyć rzeczywistą nową datę wygaśnięcia.

- `passwd -l` blokuje wyłącznie logowanie **hasłem** — nie rusza kluczy SSH ani ważności konta.
- `/etc/login.defs` działa tylko dla **nowo tworzonych** kont — istniejące trzeba poprawić ręcznie przez `chage`.

SSH = Secure Shell: szyfrowane zdalne logowanie do serwera · **last** = pokaż historię logowań użytkowników do systemu

Z życia wzięte

Z ŻYCIA WZIĘTE

O 16:00 w piątek odchodzi z firmy zewnętrzny kontraktor, który miał dostęp do docker01. Tomek szybko odpala `passwd -l` na jego koncie i uznaje sprawę za zamkniętą, ale zapomina, że kontraktor logował się kluczem SSH, nie hasłem. W poniedziałek Ola zauważa w logach, że tamto konto nadal się loguje, więc Tomek dokłada `chage -E 1`, które odcina dostęp niezależnie od metody uwierzytelniania.

Po co to istnieje

Pojęcie: wygasanie hasła kontra wygasanie konta. To dwa niezależne mechanizmy, które łatwo pomylić, bo oba w efekcie „blokują dostęp” — ale w różny sposób i z różnych powodów. Wygasanie hasła (`chage -M`, `-W`, `-d`) dotyczy **samego sekretu uwierzytelniającego**: po upływie ustalonego czasu system wymusza jego zmianę, ale konto jako takie pozostaje aktywne — użytkownik nadal może się zalogować, tylko od razu trafia w obowiązkową zmianę hasła. Wygasanie konta (`chage -E`) dotyczy **samego istnienia dostępu**: po ustalonej dacie konto przestaje działać w ogóle, niezależnie od tego, jak świeże i silne jest jego hasło. Pierwsze to polityka higieny haseł, drugie to twardy termin ważności dostępu — na przykład dla kontraktora zatrudnionego na trzy miesiące.

Pojęcie: blokada konta. Blokada konta to stan, w którym logowanie zostaje odrzucone celowo, niezależnie od poprawności podanych danych — ale sposobów zablokowania jest kilka i nie są one równoważne. `passwd -l` (identyczne działanie ma `usermod -L`) blokuje wyłącznie ścieżkę uwierzytelniania hasłem, dopisując nieprawidłowy znak na początku zapisanego hasha. `chage -E` w przeszłości blokuje konto jako całość poprzez mechanizm wygasania, sprawdzany niezależnie od metody logowania. `faillock` blokuje tymczasowo,

> automatycznie, w reakcji na serię nieudanych prób — i sam się odblokowuje po upływie skonfigurowanego czasu albo po ręcznym `faillock --reset`. Rozróżnienie tych trzech mechanizmów ma bezpośrednie znaczenie bezpieczeństwa — patrz akapit niżej o kluczach SSH.

Ten zestaw narzędzi istnieje, bo hasło to sekret, który z czasem traci wartość: im dłużej jest niezmienione, tym większa szansa, że wyciekło (baza danych innej usługi, keylogger, podpatrzenie) bez wiedzy właściciela. Wymuszanie okresowej zmiany, minimalnego czasu między zmianami (żeby nie dało się „przeklikać" z powrotem do starego hasła) i automatycznej blokady po nieudanych próbach to warstwy tej samej polityki: ograniczyć okno, w którym skradzione albo odgadnięte hasło jest użyteczne.

Analogia

Wyobraź sobie **przepustkę na budowę z terminem ważności** i osobno **furtkę z kodem, który zmienia się co miesiąc**.

Kod do furtki (hasło) ma swój własny cykl życia: co jakiś czas brygadzista wymienia go na nowy i informuje wszystkich pracowników — to jest wygasanie hasła. Ktoś, kto zna stary kod, po prostu przestaje wejść, bo kod się zmienił, ale jego przepustka jako taka wciąż jest ważna — wystarczy, że dowie się nowego kodu. Przepustka natomiast ma wydrukowaną datę ważności — dzień, po którym ochrona nie wpuści tej osoby na teren budowy, niezależnie od tego, czy zna aktualny kod do furtki, czy nie. To jest wygasanie konta: twardy, niezależny od hasła termin.

Jest jeszcze trzecia rzecz, o której żaden z tych dwóch dokumentów nic nie mówi: ochrona przy bramie ma zasadę, że po trzech błędnych próbach wpisania kodu w ciągu minuty **tymczasowo** nie wpuszcza nikogo tą furtką przez najbliższe piętnaście minut, niezależnie od tego, czy kolejna osoba zna poprawny kod, czy nie — to reakcja na podejrzaną serię prób, a nie ocena czyjejkolwiek przepustki. Po piętnastu minutach blokada sama znika. To jest `faillock`: osobny, automatyczny mechanizm reagujący na wzorzec zachowania, nie na ważność konkretnego dokumentu.

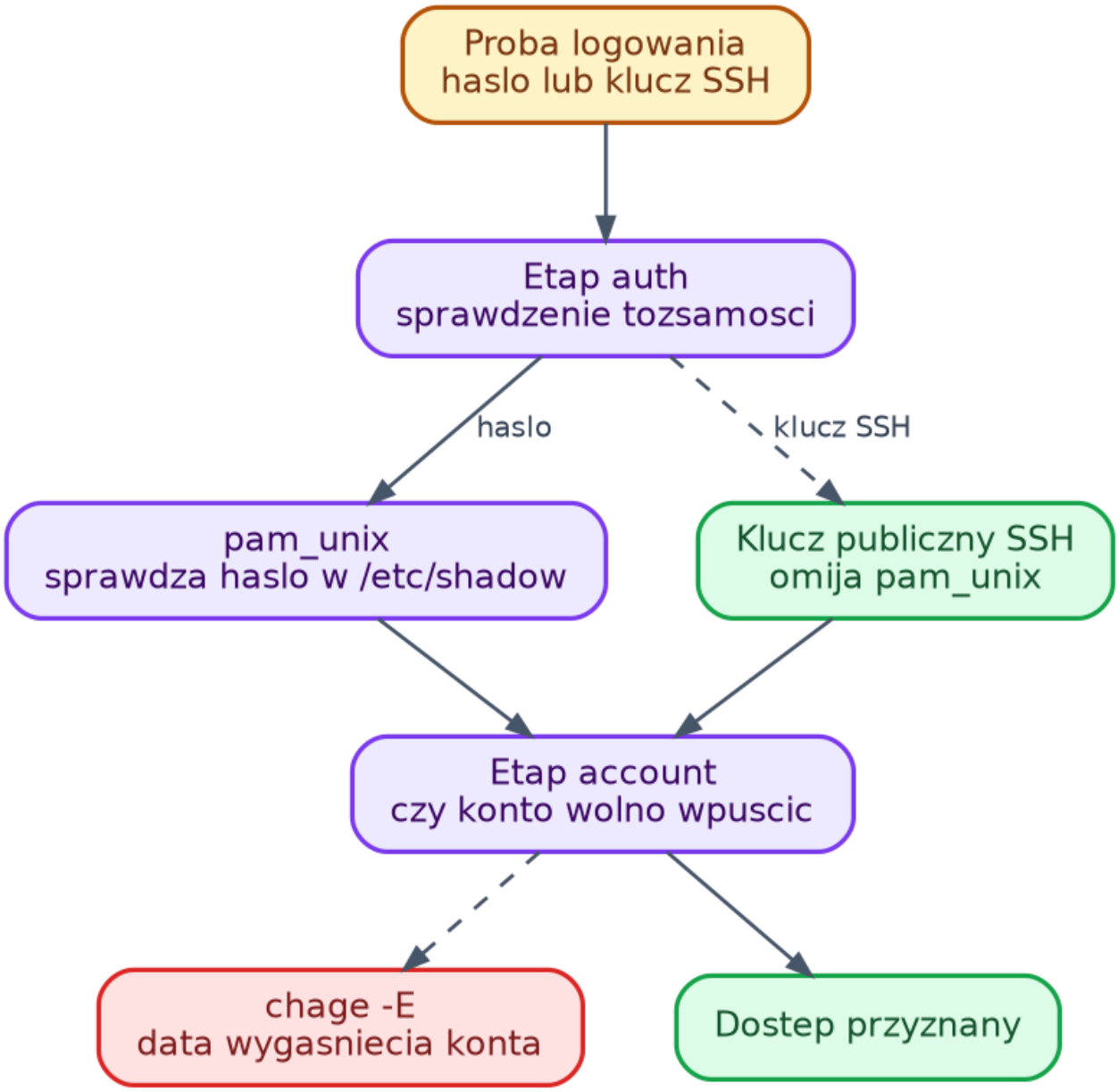
Jak to działa

▯▯ Rozwiń: jak to działa pod spodem

Wszystkie te wartości — data ostatniej zmiany hasła, maksymalny i minimalny wiek, data wygaśnięcia konta — są zapisane jako pola liczbowe (liczba dni od 1 stycznia 1970) w pliku `/etc/shadow`, czytelnym tylko dla roota. `chage` to w praktyce wygodna nakładka do edycji tych

pól bez ręcznego liczenia dni od epoki Uniksa; `passwd -S` odczytuje te same pola i tłumaczy je na daty kalendarzowe. `chage -d 0` ustawia pole „data ostatniej zmiany” wprost na zero — najwcześniejszą możliwą wartość — co w połączeniu z jakimkolwiek limitem `-M` czyni hasło automatycznie przeterminowanym od razu przy najbliższym logowaniu.

`passwd -l` i `usermod -L` operują na zupełnie innym polu — na samym hashu hasła. Dopisują na jego początku wykrzyknik (`!`), przez co żaden ciąg wpisany przez użytkownika, po przepuszczeniu przez funkcję skrótu, nigdy nie dopasuje się do zapisanej wartości — logowanie hasłem staje się matematycznie niemożliwe do spełnienia, nie tylko „zablokowane administracyjnie” w jakimś osobnym polu flag.



Etap auth sprawdza tożsamość przez pam_unix (hasło) albo klucz SSH, który go omija; oba prowadzą do etapu account, gdzie dopiero data wygaśnięcia konta blokuje dostęp niezależnie od metody logowania (kliknięcie otwiera pełny rozmiar)

To prowadzi do najważniejszego zastrzeżenia tej strony: **logowanie kluczem SSH w ogóle nie sprawdza pola hasła**. Proces uwierzytelniania w PAM dzieli się na etap `auth` (sprawdzenie tożsamości — hasło, klucz, token) i etap `account` (sprawdzenie, czy konto jako takie wolno wpuścić — czy nie wygasło, czy nie jest zablokowane przez system). Uwierzytelnienie kluczem publicznym SSH odbywa się poza modułem `pam_unix`, który sprawdza hasło — więc zablokowanie hasła przez `passwd -l` nie ma żadnego wpływu na tę ścieżkę. Etap `account` w PAM **owszem** sprawdza datę wygaśnięcia konta ustawioną przez `chage -E` (i tu

logowanie kluczem zostanie odrzucone tak samo jak logowanie hasłem), ale nie sprawdza samego faktu zablokowania hasła — to nie jest stan konta, tylko stan konkretnej metody uwierzytelniania. Skutek praktyczny: administrator, który „wyłącza” odchodzącego pracownika przez `passwd -l`, może być przekonany, że dostęp został odcięty, podczas gdy dostęp przez klucz SSH — jeśli pracownik go kiedykolwiek skonfigurował — nadal działa bez żadnego ostrzeżenia. Pełne odcięcie wymaga albo `chage -E 1` (twarde wygaśnięcie konta, sprawdzane niezależnie od metody logowania), albo dodatkowego usunięcia/zablokowania kluczy w `~/.ssh/authorized_keys`.

`faillock` działa jeszcze inaczej — to moduł PAM (`pam_faillock`) wpięty w etap `auth`, prowadzący własny, oddzielny licznik nieudanych prób w katalogu `/var/run/faillock/` (per użytkownik). Po przekroczeniu progu (konfigurowanego w `/etc/security/faillock.conf`, domyślnie kilka prób) `pam_faillock` sam zaczyna odrzucać próby logowania na określony czas, niezależnie od tego, czy podane hasło byłoby akurat poprawne. To mechanizm ochrony przed atakiem brute-force, nie mechanizm trwałej blokady konta — stąd `faillock --reset` odblokowuje natychmiast, bez ruszania hasła ani dat wygaśnięcia.

Krok po kroku

1. Sprawdzenie bieżącej polityki dla konta.

```
sudo chage -l user1
```

2. Ustawienie pełnej polityki wygaśnięcia hasła naraz.

```
sudo chage -M 90 -m 1 -W 7 user1
sudo chage -l user1
```

```
Maksymalna liczba dni pomiędzy zmianami hasła          : 90
Minimalna liczba dni pomiędzy zmianami hasła           : 1
Liczba dni ostrzeżenia przed wygaśnięciem hasła        : 7
```

3. Wymuszenie zmiany hasła przy następnym logowaniu — bez blokowania konta.

```
sudo chage -d 0 user1
sudo passwd -S user1
```

```
user1 P 1970-01-01 1 90 7 -1
```

Data ostatniej zmiany cofnięta do epoki — hasło jest już formalnie przeterminowane, ale status `P` (usable) potwierdza, że logowanie samo w sobie nadal działa; tylko od razu poprosi o nowe hasło.

4. Ustawienie twardego terminu wygaśnięcia konta kontraktora.

```
sudo chage -E 2026-12-31 kontraktor1
sudo chage -l kontraktor1 | grep "Konto wygasa"
```

```
Konto wygasa                                          : gru 31, 2026
```

5. Diagnoza po serii nieudanych logowań.

```
sudo faillock --user user1
```

Jeśli licznik przekroczył próg, konto jest tymczasowo zablokowane przez `pam_faillock` — nie przez `passwd -l` i nie przez `chage`.

```
sudo faillock --user user1 --reset
```

6. Zestawienie prób nieudanych z udanymi logowaniami.

```
sudo lastb -n 10 | grep user1
last -n 10 user1
```

Pozwala ocenić, czy seria nieudanych prób poprzedziła udane logowanie z tego samego adresu (podejrzenie skutecznego brute-force) czy skończyła się samą blokadą.

Częste błędy

Zablokowanie hasła w przekonaniu, że to odcina cały dostęp.

```
$ sudo passwd -l bylly_pracownik
passwd: hasło zmienione.
```

Komenda wykonuje się bez błędu i wygląda na skuteczną — `passwd -S` potwierdzi status `L`. Problem pojawia się później, często podczas audytu bezpieczeństwa: jeśli `bylly_pracownik` miał kiedykolwiek dodany własny klucz do `~/.ssh/authorized_keys`, logowanie tą drogą nadal działa, bez żadnego komunikatu ostrzegawczego po stronie klienta — połączenie po prostu się udaje. Pełne odcięcie wymaga dodatkowo `sudo chage -E 1 bylly_pracownik` (wygaśnięcie konta, sprawdzane niezależnie od metody logowania) oraz przeglądu `~/.ssh/authorized_keys`.

`chage -M` nie działa „od dziś”, tylko od ostatniej zmiany hasła.

```
$ sudo chage -M 30 user1
$ ssh user1@srv01
Twoje hasło wygasło.
Musisz je teraz zmienić (wymagane przez administratora).
```

Administrator ustawia limit 30 dni, spodziewając się, że hasło wygaśnie za miesiąc — a użytkownik dostaje wymuszoną zmianę już przy najbliższym logowaniu. Powód: hasło było ostatnio zmienione np. 45 dni temu, więc nowy, krótszy limit został już przekroczony w chwili jego ustawienia. Sprawdzenie `chage -l` od razu po zmianie `-M` pokazuje rzeczywistą nową datę wygaśnięcia, zamiast zakładać ją na podstawie samej liczby dni.

Zmiana `/etc/login.defs` bez efektu na istniejące konta.

```
$ sudo vim /etc/login.defs   # PASS_MAX_DAYS 90 -> 30
$ sudo chage -l user1 | grep Maksymalna
Maksymalna liczba dni pomiędzy zmianami hasła           : 90
```

`login.defs` jest odczytywany wyłącznie przez `useradd` w momencie tworzenia nowego konta — kopiuje wartości do `/etc/shadow` tego jednego konta i na tym kończy swoją rolę. Zmiana pliku nie propaguje się do kont, które już istnieją. Naprawa dla istniejących kont wymaga jawnego `chage -M` na każdym z nich (albo pętli po wszystkich kontach, analogicznie do skryptu audytującego sudo z poprzedniej strony).

Powtarzające się blokady logowania mylone z problemem hasła.

```
$ ssh user1@srv01
user1@srv01's password:
Permission denied, please try again.
user1@srv01's password:
Permission denied (publickey,password).
```

Po kilku nieudanych próbach (np. literówka w haśle powtórzona kilka razy) `pam_faillock` zaczyna odrzucać **każdą** kolejną próbę, także z poprawnym hasłem, przez skonfigurowany czas. Administrator, który w tym momencie resetuje hasło przez `passwd`, nie rozwiązuje problemu — blokada dotyczy licznika prób, nie samego hasła. Właściwa diagnoza i naprawa:

```
sudo faillock --user user1
sudo faillock --user user1 --reset
```

Mylenie `passwd -e` z `passwd -l`.

Administrator chce zapobiec logowaniu użytkownika, więc uruchamia `sudo passwd -e user1`, spodziewając się blokady — ale `-e` tylko wymusza zmianę hasła przy najbliższym logowaniu, nie

blokuje niczego. Użytkownik loguje się bez przeszkód, tylko od razu trafia na monit o ustawienie nowego hasła — dokładnie odwrotność zamierzonego efektu, jeśli celem było odcięcie dostępu.

□□ Ćwiczenia

> Ćwiczenie 1 działa na Twoim własnym koncie bez `sudo`. Ćwiczenia 2 i 3 wymagają uprawnień administratora i operują na koncie testowym — wykonaj je na maszynie testowej. Wyniki pośrednie zapisuj w `/tmp`.

Ćwiczenie 1 — odczyt własnej polityki hasła

```
chage -l $(whoami) | tee /tmp/moja_polityka.txt
```

Znajdź wartość „Maksymalna liczba dni pomiędzy zmianami hasła”. Czy jest ustawiona, czy widnieje jako brak limitu, i co to mówi o polityce Twojej maszyny?

Odpowiedź

Wartość zależy od konfiguracji konkretnej maszyny testowej — na świeżo zainstalowanym systemie deweloperskim najczęściej nie ma ustawionego limitu (widnieje jako brak ograniczenia albo bardzo duża liczba dni), bo domyślne `login.defs` na obrazach do nauki rzadko wymusza restrykcyjną politykę haseł, w przeciwieństwie do środowisk produkcyjnych, gdzie `PASS_MAX_DAYS` jest zwykle jawnie skonfigurowane.

Ćwiczenie 2 — pełny cykl wygasania hasła na koncie testowym (wymaga sudo)

```
sudo useradd -m cwiczenie_chage
sudo passwd cwiczenie_chage
sudo chage -M 30 -W 5 cwiczenie_chage
sudo chage -l cwiczenie_chage
sudo chage -d 0 cwiczenie_chage
sudo passwd -S cwiczenie_chage
```

Dlaczego status w ostatniej komendzie nadal pokazuje `P` (usable), skoro hasło jest już formalnie wygasłe?

Odpowiedź

`P` oznacza, że logowanie **jest możliwe** — konto nie jest zablokowane. Wygasłe hasło nie blokuje logowania samo w sobie: system pozwala się zalogować, ale natychmiast przerywa sesję żądaniem ustawienia nowego hasła, zanim użytkownik dostanie działającą powłokę. Status `L` pojawiłby się dopiero, gdyby ktoś dodatkowo użył `passwd -l`, albo gdyby minął okres nieaktywności ustawiony przez `chage -I` bez zmiany hasła.

Ćwiczenie 3 — porównanie trzech rodzajów blokady (wymaga sudo)

```
sudo passwd -l cwiczenie_chage
sudo passwd -S cwiczenie_chage
sudo passwd -u cwiczenie_chage
sudo chage -E 1 cwiczenie_chage
sudo chage -l cwiczenie_chage | grep "Konto wygasa"
```

Posprzątaj:

```
sudo userdel -r cwiczenie_chage
```

Który z tych dwóch stanów (`passwd -l` czy `chage -E 1`) zablokowałby też logowanie kluczem SSH, gdyby konto go miało skonfigurowanego?

Odpowiedź

`chage -E 1` — wygaśnięcie konta jest sprawdzane na etapie `account` w PAM, niezależnie od metody uwierzytelniania, więc blokuje też logowanie kluczem. `passwd -l` blokuje wyłącznie pole hasła sprawdzane przez `pam_unix` na etapie `auth` — logowanie kluczem SSH w ogóle nie przechodzi przez ten etap, więc `passwd -l` go nie dotyczy.

Pytania kontrolne

1. Czym różni się wygasanie hasła od wygasania konta?

Odpowiedź

Wygasanie hasła (`chage -M`, `-d`) dotyczy ważności samego sekretu — po jego upływie użytkownik nadal może się zalogować, ale musi od razu ustawić nowe hasło. Wygasanie konta (`chage -E`) to twardy termin, po którym konto przestaje działać w ogóle, niezależnie od tego, jak świeże jest hasło.

2. Zablokowałeś konto przez `passwd -l`. Czy to wystarczy, żeby uniemożliwić dostęp przez klucz SSH?

Odpowiedź

Nie. `passwd -l` blokuje wyłącznie logowanie hasłem, zmieniając pole hasła w `/etc/shadow` tak, by żadne hasło się nie dopasowało. Uwierzytelnianie kluczem SSH nie sprawdza tego pola w ogóle, więc pozostaje w pełni funkcjonalne. Pełne odcięcie dostępu wymaga `chage -E` z datą w przeszłości albo usunięcia kluczy z `authorized_keys`.

3. Do czego służy `faillock --reset` i czym różni się od `passwd -u`?

Odpowiedź

`faillock --reset` zeruje licznik nieudanych prób logowania prowadzony przez `pam_faillock` i znosi wynikającą z niego tymczasową blokadę — nie rusza przy tym samego hasła. `passwd -u` odblokowuje hasło zablokowane wcześniej przez `passwd -l`, zdejmując wykrzyknik z pola hasha w `/etc/shadow` — to zupełnie inny mechanizm, niezwiązany z licznikiem prób.

4. Dlaczego zmiana `PASS_MAX_DAYS` w `/etc/login.defs` nie wpływa na konta utworzone przed tą zmianą?

Odpowiedź

`login.defs` jest odczytywany wyłącznie w momencie tworzenia konta przez `useradd`, które kopiuje jego wartości do pól `/etc/shadow` tego jednego, nowego konta. Po utworzeniu konta te pola żyją własnym życiem i nie są już powiązane z plikiem `login.defs` — jego dalsza zmiana dotyczy tylko kolejnych, przyszłych kont. Istniejące konta trzeba poprawić bezpośrednio przez `chage`.

Powiązane: sudo i /etc/sudoers: konfiguracja i audyt · SSH: klucze, ssh-agent, config, hartowanie · Zarządzanie użytkownikami i grupami: useradd, usermod, userdel · PAM: stos uwierzytelniania

Źródła: `man 1 chage` · `man 1 passwd` · `man 5 shadow` · `man 8 pam_faillock` · `man 5 login.defs` · `man 5 pwquality.conf` · stare BookStack: Sprawdzenie daty wygasania hasła użytkownika · OneNote „Ogolne" (Sprawdzenie czasu wygasania hasła)

Revision #16
Created 2026-09-17 21:42:56 UTC by Tomasz Konieczny
Updated 2026-10-03 19:08:36 UTC by Tomasz Konieczny