Komunikat powitalny: MOTD i banery

Poziom: podstawy **Dotyczy: RHEL/Rocky 9–10, różnice Debian/Ubuntu zaznaczone**

Czas: ~16 min

`/etc/motd` *to zwykły plik tekstowy wypisywany po zalogowaniu — bez interpretera, bez zmiennych, bez logiki. Cała „dynamiczność”, którą widujesz na cudzych serwerach, to skrypt uruchamiany przy logowaniu, który dopiero generuje treść tego pliku albo wypisuje ją bezpośrednio.*

Skróty użyte na tej stronie (7)

Skrót	Co znaczy
ANSI	American National Standards Institute: tu chodzi o kody sterujące kolorem i pozycją kursora w terminalu
BSD	Berkeley Software Distribution: rodzina Uniksów z uczelni Berkeley, inna gałąź niż Linux, stąd czasem inna składnia tych samych poleceń
LED	Light Emitting Diode: dioda świecąca, tu źródło światła w czujce optycznej i w optoizolatorze
MOTD	Message Of The Day: wiadomość pokazywana zaraz po zalogowaniu przez SSH, zanim zobaczymy znak zachęty
PAM	Pluggable Authentication Modules: wymienne moduły logowania w Linuksie, decydują na przykład o sile wymaganego hasła
RHEL	Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa
SSH	Secure Shell: szyfrowane zdalne logowanie do serwera

Fiszka

TYLKO ODCZYT

Odczyt bieżącego komunikatu: `cat /etc/motd`

```
cat /etc/motd
```

Domyślnie na świeżym RHEL/Rocky ten plik jest pusty albo nie istnieje w ogóle — treść powitalna na większości nowoczesnych dystrybucji generuje się dynamicznie (patrz `update-motd.d` niżej), a nie leży zapisana na stałe w tym pliku.

Przykład:

□□ **RHEL** = Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa · **cat** = concatenate: sklej i wypisz pliki

□□ ZMIENIA SYSTEM

Zapis statycznego komunikatu przez here-document

```
cat > /etc/motd <<'EOF'
Serwer produkcyjny.
Zmiany wyłącznie przez zgłoszenie w systemie ticketowym.
Nieautoryzowany dostęp jest monitorowany i rejestrowany.
EOF
```

Najprostszy możliwy sposób na stały, statyczny baner. Apostrofy wokół `EOF` są obowiązkowe, jeśli treść ma zawierać znaki specjalne bez rozwijania ich przez powłokę. `/etc/motd` jest zwykłym plikiem tekstowym — nie interpretuje poleceń ani zmiennych, wyświetla dosłownie to, co w nim zapisano.

□□ **cat** = concatenate: sklej i wypisz pliki

□□ TYLKO ODCZYT

Wyświetlenie banera przed logowaniem na konsoli lokalnej: `/etc/issue`

```
cat /etc/issue
```

Wyświetlany na konsoli lokalnej (fizycznej albo przez `getty`) **przed** podaniem loginu i hasła — w przeciwieństwie do `/etc/motd`, które widać dopiero **po** udanym zalogowaniu.

Przykład:

```
Rocky Linux 9.4 (Blue Onyx)
Kernel \r on an \m
```

cat = concatenate: sklej i wypisz pliki

ZMIENIA SYSTEM

Sekwencje ucieczki w `/etc/issue`: `\S \r \m \l \b`

```
cat > /etc/issue <<'EOF'
\S
Kernel \r on an \m
Login: \l  Speed: \b
EOF
```

`getty` podmienia te sekwencje na żywe dane systemu w momencie wyświetlania banera — to jedyne miejsce w tym temacie, gdzie plik statyczny potrafi pokazać coś dynamicznego, bez żadnego skryptu.

Przykład:

```
Rocky Linux 9.4 (Blue Onyx)
Kernel 5.14.0-427.el9.x86_64 on an x86_64
Login: tty1  Speed: 38400
```

Sekwencja	Podstawia
`\S`	nazwa i wersja systemu (z `/etc/os-release`)
`\r`	wersja jądra (release)
`\m`	architektura sprzętu
`\l`	nazwa linii terminala (tty)
`\b`	prędkość transmisji (baud rate)

cat = concatenate: sklej i wypisz pliki

TYLKO ODCZYT

Wyświetlenie banera przed logowaniem przez sieć: `/etc/issue.net`

```
cat /etc/issue.net
```

Odpowiednik `/etc/issue`, ale dla logowań przychodzących przez sieć — w praktyce niemal zawsze SSH. Ma własny plik, bo te dwie ścieżki logowania (konsola i sieć) są obsługiwane

przez różne mechanizmy i nie muszą pokazywać tego samego.

SSH = Secure Shell: szyfrowane zdalne logowanie do serwera · **cat** = concatenate: sklej i wypisz pliki

TYLKO ODCZYT

Włączenie banera SSH: `Banner` w `/etc/ssh/sshd_config`

```
grep -i '^Banner' /etc/ssh/sshd_config
```

Przykład:

```
Banner /etc/issue.net
```

> **Pojęcie: baner przed uwierzytelnieniem kontra po.** To rozróżnienie, na którym stoi cały ten temat: baner przed uwierzytelnieniem (`/etc/issue`, `/etc/issue.net`) pokazuje się każdemu, kto tylko połączy się z serwerem — także temu, kto nigdy nie poda poprawnego hasła, i dlatego zwykle zawiera ostrzeżenie prawne, nie żadne dane wewnętrzne. Baner po uwierzytelnieniu (`/etc/motd`) pokazuje się wyłącznie temu, kto już się zalogował, więc może bezpiecznie zawierać stan systemu, adresy wewnętrzne czy informacje serwisowe. Kluczowy szczegół: **sam `/etc/issue.net` nic nie zrobi dla SSH, dopóki dyrektywa `Banner` w `sshd_config` nie wskaże na niego wprost** — inaczej niż `/etc/issue`, które `getty` czyta automatycznie bez żadnej konfiguracji.

Bez ustawienia `Banner` demon `sshd` w ogóle nie zajrzy do `/etc/issue.net`, niezależnie od tego, co w nim zapiszemy.

SSH = Secure Shell: szyfrowane zdalne logowanie do serwera · **grep** = global regular expression print: szukaj wzorca w tekście

TYLKO ODCZYT

`PrintMotd` i `PrintLastLog` w `sshd_config`

```
grep -iE '^PrintMotd|^PrintLastLog' /etc/ssh/sshd_config
```

Przykład:

```
PrintMotd no
PrintLastLog yes
```

`PrintMotd` każe samemu `sshd` wypisać `/etc/motd` po zalogowaniu — na nowoczesnych dystrybucjach jest to zwykle wyłączone (`no`), bo o wypisanie MOTD dba `pam_motd` (niżej), a podwójne wypisanie dawałoby komunikat dwa razy. `PrintLastLog` pokazuje datę i adres ostatniego udanego logowania — osobna, choć sąsiadująca funkcja.

MOTD = Message Of The Day: wiadomość pokazywana zaraz po zalogowaniu przez SSH, zanim zobaczymy znak zachęty · **grep** = global regular expression print: szukaj wzorca w tekście

ZMIENIA SYSTEM

Generowanie dynamicznego MOTD na Debianie/Ubuntu: `/etc/update-motd.d/`

```
ls -la /etc/update-motd.d/
```

Katalog ze skryptami numerowanymi — numer w nazwie decyduje o kolejności wykonania, tak jak w `/etc/rc.d/init.d` w dawnych czasach SysV init.

Przykład:

```
-rwxr-xr-x 1 root root  982 lut 12 10:03 00-header
-rwxr-xr-x 1 root root  604 lut 12 10:03 10-uname
-rwxr-xr-x 1 root root  399 lut 12 10:03 50-landscape-sysinfo
-rwxr-xr-x 1 root root  217 lut 12 10:03 90-updates-available
```

MOTD = Message Of The Day: wiadomość pokazywana zaraz po zalogowaniu przez SSH, zanim zobaczymy znak zachęty · **ls** = list: wypisz zawartość katalogu

ZMIENIA SYSTEM

Własny skrypt w `update-motd.d` i `chmod +x`

```
cat > /etc/update-motd.d/80-stan-systemu <<'EOF'
#!/bin/bash
echo "Uptime: $(uptime -p)"
echo "Obciążenie: $(cut -d ' ' -f1-3 /proc/loadavg)"
echo "Wolne miejsce na /: $(df -h / | awk 'NR==2{print $4}')"
EOF
chmod +x /etc/update-motd.d/80-stan-systemu
```

Skrypt bez prawa wykonania (`chmod +x`) zostanie po cichu pominięty przez `run-parts` — bez błędu, po prostu nie wykona się nigdy, co jest jedną z najczęstszych przyczyn „mój skrypt MOTD nie działa”.

MOTD = Message Of The Day: wiadomość pokazywana zaraz po zalogowaniu przez SSH, zanim zobaczymy znak zachęty · **cat** = concatenate: sklej i wypisz pliki · **chmod** = change mode: zmień prawa dostępu

ZMIENIA SYSTEM

Uruchomienie mechanizmu generującego MOTD: `run-parts /etc/update-motd.d/`

```
run-parts /etc/update-motd.d/ > /run/motd.dynamic
```

To dokładnie to polecenie (wywoływane automatycznie, nie ręcznie na co dzień) stoi za dynamicznym MOTD na Debianie/Ubuntu: uruchamia po kolei każdy wykonywalny skrypt z katalogu, w porządku alfabetyczno-numerycznym, i zbiera ich standardowe wyjście do jednego pliku, który dopiero wtedy trafia na ekran użytkownika.

MOTD = Message Of The Day: wiadomość pokazywana zaraz po zalogowaniu przez SSH, zanim zobaczymy znak zachęty

TYLKO ODCZYT

`pam_motd` w `/etc/pam.d/sshd`

```
grep motd /etc/pam.d/sshd
```

Przykład:

```
session    optional     pam_motd.so motd=/run/motd.dynamic
session    optional     pam_motd.so noupdate
```

> **Pojęcie: dynamiczny MOTD.** Komunikat powitalny, który nie jest zapisany na stałe w jednym pliku, tylko generowany od nowa przy każdym logowaniu przez zestaw skryptów odczytujących bieżący stan systemu. Mechanizm PAM (`pam_motd`) jest tym, co faktycznie wypisuje wynik działania `run-parts` (albo, na RHEL 9+, zawartość `/etc/motd.d/`) w trakcie procesu logowania — bez wpisu `pam_motd.so` w `/etc/pam.d/sshd` (albo `/etc/pam.d/login` dla konsoli) żaden dynamiczny mechanizm się nie uruchomi, niezależnie od tego, ile skryptów przygotujemy w `update-motd.d`.

□□ **MOTD** = Message Of The Day: wiadomość pokazywana zaraz po zalogowaniu przez SSH, zanim zobaczymy znak zachęty · **PAM** = Pluggable Authentication Modules: wymienne moduły logowania w Linuksie, decydują na przykład o sile wymaganego hasła · **RHEL** = Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa · **grep** = global regular expression print: szukaj wzorca w tekście

□□ TYLKO ODCZYT

Generowanie dynamicznego MOTD na RHEL 9+: `/etc/motd.d/`

```
ls /etc/motd.d/
```

Odpowiednik `update-motd.d`, ale prostszy: pliki w tym katalogu nie są skryptami do uruchomienia, tylko gotowymi fragmentami tekstu, które `pam_motd` skleja w jedną całość i wypisuje. Nie trzeba nadpisywać jednego wspólnego `/etc/motd` — każdy pakiet czy administrator dokłada własny plik.

Przykład:

```
10-banner
50-nagroda-za-uptime
```

```
cat > /etc/motd.d/50-info-serwisu <<'EOF'
Kontakt serwisowy: zgloszenia@zielona.example
EOF
```

Dopisanie własnej treści to po prostu nowy plik w katalogu — bez ryzyka nadpisania tego, co zostawił tam instalator systemu albo inny zespół.

□□ **MOTD** = Message Of The Day: wiadomość pokazywana zaraz po zalogowaniu przez SSH, zanim zobaczymy znak zachęty · **RHEL** = Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa · **ls** = list: wypisz zawartość katalogu · **cat** = concatenate: sklej i wypisz pliki

□□ ZMIENIA SYSTEM

Generowanie dynamicznego MOTD ze stanem systemu — pełny przykład

```
cat > /etc/update-motd.d/80-stan-systemu <<'SKRYPT'
#!/bin/bash
RED='\033[31m'; GREEN='\033[32m'; YELLOW='\033[33m'; RESET='\033[0m'
```

```
echo -e "${YELLOW}Uptime:${RESET} $(uptime -p)"
echo -e "${YELLOW}Obciążenie (1/5/15 min):${RESET} $(cut -d ' ' -f1-3 /proc/loadavg)"
echo -e "${YELLOW}Pamięć wolna:${RESET} $(free -h | awk '/Mem:/{print $7}')"
echo -e "${YELLOW}Wolne miejsce na /:${RESET} $(df -h / | awk 'NR==2{print $4}')"

count_process() {
  pgrep -c "$1" 2>/dev/null || echo 0
}

for usluga in nginx postgresql sshd; do
  n=$(count_process "$usluga")
  if [ "$n" -gt 0 ]; then
    echo -e "${usluga}: ${GREEN}RUNNING (${n})${RESET}"
  else
    echo -e "${usluga}: ${RED}NOT RUNNING${RESET}"
  fi
done
SKRYPT
chmod +x /etc/update-motd.d/80-stan-systemu
```

Struktura funkcyjna (zmienne kolorów ANSI, funkcja licząca procesy, pętla po liście usług) — praktyczny wzorzec z produkcji: zamiast trzymać osobne linie dla każdej usługi, pętla po tablicy nazw skaluje się bez kopiowania kodu. Na Linuksie licznik procesów i obciążenie czyta się z `/proc`, nie przez `sysctl` (który jest mechanizmem BSD/macOS) — to jedna z tych rzeczy, które trzeba dostosować, przenosząc gotowy skrypt z innego uniksa.

Przykład:

```
Uptime: up 14 days, 3 hours, 22 minutes
Obciążenie (1/5/15 min): 0.42 0.38 0.31
Pamięć wolna: 2.1G
Wolne miejsce na /: 38G
nginx: RUNNING (2)
postgresql: RUNNING (5)
sshd: NOT RUNNING
```

MOTD = Message Of The Day: wiadomość pokazywana zaraz po zalogowaniu przez SSH, zanim zobaczymy znak zachęty · **ANSI** = American National Standards Institute: tu chodzi o kody sterujące kolorem i pozycją kursora w terminalu · **BSD** = Berkeley Software Distribution: rodzina Uniksów z uczelni Berkeley, inna gałąź niż Linux, stąd czasem inna składnia tych samych poleceń · **cat** = concatenate: sklej i wypisz pliki · **pgrep** = process grep: znajdź numer procesu po nazwie bez przeglądania pełnej listy ps · **chmod** = change mode: zmień prawa dostępu

◫◫ ZMIENIA SYSTEM

Liczba dostępnych aktualizacji w MOTD

```
echo "Dostępne aktualizacje: $(dnf check-update --quiet 2>/dev/null | grep -c '^[a-zA-Z]')"
```

Na Debianie/Ubuntu odpowiednik to `apt list --upgradable 2>/dev/null | grep -c upgradable` — obie wersje bywają wolne (odpytują bazę pakietów), więc w praktyce lepiej cache'ować wynik w pliku odświeżanym raz dziennie przez `cron`, a nie liczyć go od nowa przy każdym logowaniu.

◫◫ **MOTD** = Message Of The Day: wiadomość pokazywana zaraz po zalogowaniu przez SSH, zanim zobaczymy znak zachęty

◫◫ TYLKO ODCZYT

Wstawienie dużego napisu ASCII: `figlet` / `toilet`

```
figlet "SRV01"
```

Przykład:

```
 ____  ____  __     __   ___
/ ___||  _ \ \ \   / /  / _ \_ _
\___ \| |_) | \ \ / /  | | | | |
 ___) |  _ <   \ V /   | |_| |_|
|____/|_| \_\   \_/     \___/(_)
```

`toilet` to nowszy zamiennik z obsługą kolorów ANSI i większym zestawem czcionek (`toilet -f mono12 --gay "SRV01"` dla kolorowego gradientu). Obie komendy trzeba doinstalować — nie ma ich domyślnie w minimalnej instalacji.

◫◫ **ASCII** = American Standard Code for Information Interchange: podstawowy zestaw znaków (litery, cyfry) używany w komputerach · **ANSI** = American National Standards Institute: tu chodzi o kody sterujące kolorem i pozycją kursora w terminalu

◫◫ TYLKO ODCZYT

Wszystkie pliki i mechanizmy

Element	Kiedy się pokazuje	Interpretuje polecenia?
`/etc/issue`	przed logowaniem, konsola lokalna	nie, tylko sekwencje `\X`
`/etc/issue.net`	przed logowaniem, przez sieć (SSH, wymaga `Banner`)	nie
`/etc/motd`	po zalogowaniu	nie
`/etc/update-motd.d/*` (Debian/Ubuntu)	po zalogowaniu, generowane na bieżąco	tak — to są skrypty
`/etc/motd.d/*` (RHEL 9+)	po zalogowaniu, sklejane na bieżąco	nie — to gotowe fragmenty tekstu
`pam_motd` w `/etc/pam.d/sshd` lub `/etc/pam.d/login`	mechanizm wypisujący powyższe	—

Uważaj

- Baner ostrzegawczy prawny musi być w `/etc/issue` i `/etc/issue.net` (widoczny przed logowaniem, więc dotyczy też nieudanych prób) — sam `/etc/motd` nie wystarcza jako podstawa prawna, bo widzi go tylko ten, kto już wszedł.
- `/etc/issue.net` bez ustawionej dyrektywy `Banner` w `sshd_config` jest po prostu ignorowany przez `sshd`.
- Skrypt w `/etc/update-motd.d/` bez `chmod +x` zostaje pominięty bez żadnego komunikatu błędu.

SSH = Secure Shell: szyfrowane zdalne logowanie do serwera · **RHEL** = Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa

Z życia wzięte

Z ŻYCIA WZIĘTE

Poniedziałek, 8:00: pan Wojtek podczas dorocznego audytu pyta, czy logowanie do serwerów pokazuje ostrzeżenie prawne, zanim ktokolwiek poda hasło. Sprawdzamy `/etc/issue.net` na docker01 i widzimy własny tekst ostrzeżenia, ale `ssh` i tak nie pokazuje żadnego banera. Dopiero brakująca linia `Banner /etc/issue.net` w `sshd_config` tłumaczy, dlaczego plik istniał od miesięcy, a nikt go nigdy nie widział.

Po co to istnieje

> **Pojęcie: komunikat dnia (MOTD).** MOTD (Message Of The Day) to tekst wyświetlany użytkownikowi zaraz po udanym zalogowaniu do systemu — zwyczajowo powitanie, informacje o systemie albo ostrzeżenie dotyczące zasad korzystania z maszyny. W najprostszej postaci to jeden statyczny plik, `/etc/motd`, który jest zwykłym plikiem tekstowym: nie interpretuje poleceń ani zmiennych powłoki, wyświetla dosłownie to, co w nim zapisano. Jeśli chcesz, żeby komunikat pokazywał aktualny stan systemu — obciążenie, wolne miejsce, status usług — potrzebujesz osobnego mechanizmu, który wygeneruje treść na nowo przy każdym logowaniu. Tym mechanizmem jest dynamiczny MOTD, opisany niżej.

Sens istnienia MOTD nie jest wyłącznie kosmetyczny. Na serwerze produkcyjnym pierwsza rzecz, którą widzi administrator po zalogowaniu, to potencjalnie najważniejsza rzecz, jaką powinien zobaczyć: czy dysk się nie zapełnia, czy kluczowa usługa działa, czy ktoś inny jest właśnie zalogowany i coś robi. MOTD to tani, zawsze widoczny kanał informacyjny, który nie wymaga uruchamiania żadnej dodatkowej komendy — pojawia się sam, zanim administrator zdąży cokolwiek wpisać.

Analogia

Wyobraź sobie **tablicę ogłoszeń przy wejściu do zakładu produkcyjnego** — i osobno, **budkę strażnika przy bramie**.

Budka strażnika stoi przed bramą. Widzi ją każdy, kto tylko podjedzie pod zakład — dostawca, kontroler, ktoś, kto się pomylił adresem, i pracownik, który zaraz przyłoży kartę. Napis na budce mówi „teren monitorowany, wstęp za okazaniem przepustki” — to komunikat dla każdego, niezależnie od tego, czy wejdzie, czy zawróci. To jest dokładnie `/etc/issue` i `/etc/issue.net`: widoczne przed jakąkolwiek weryfikacją tożsamości, więc treścią nadaje się do ostrzeżeń prawnych, nie do informacji poufnych.

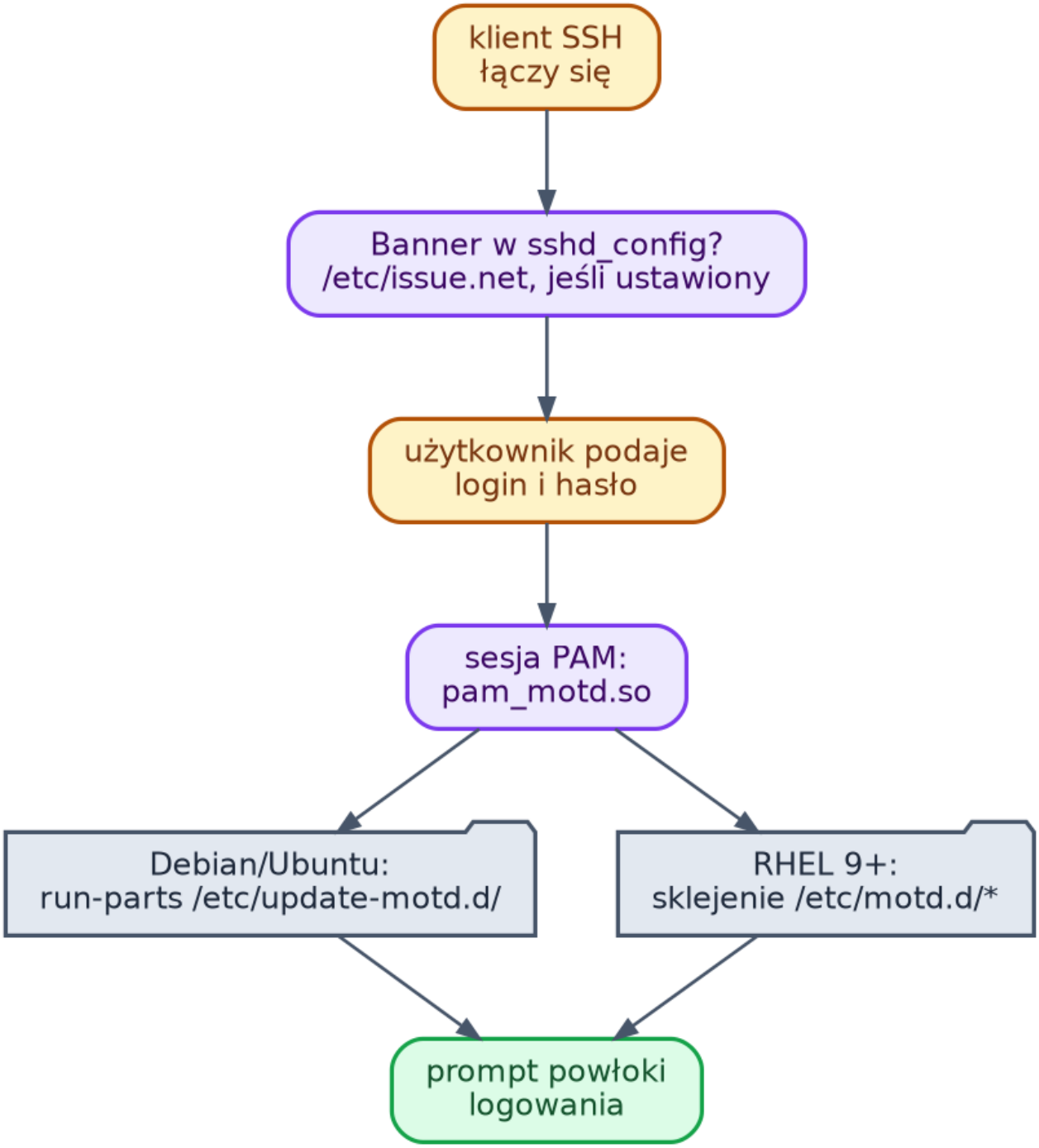
Tablica ogłoszeń stoi **za** bramą, w holu, widoczna dopiero dla tych, którzy już przyłożyli kartę i weszli. Może więc wisieć na niej harmonogram zmian, informacja o awarii windy towarowej, przypomnienie o audycie w przyszłym tygodniu — rzeczy, które nie są tajne, ale też nie mają sensu dla przypadkowego przechodnia pod bramą. To jest `/etc/motd`: dostępne wyłącznie dla kogoś, kto już się uwierzytelnił.

Różnica między tablicą statyczną a dynamiczną to różnica między kartką przypiętą pinezką raz na tydzień a monitorem LED, który dobierze wyświetlaną treść w chwili, gdy ktoś przejdzie obok — pokazując akurat to, co jest aktualne teraz, a nie to, co ktoś zapisał trzy dni temu.

Jak to działa

Rozwiń: jak to działa pod spodem

Kolejność zdarzeń przy logowaniu przez SSH wygląda tak:



Kolejność zdarzeń przy logowaniu SSH: baner przed uwierzytelnieniem, potem sesja PAM generuje MOTD osobnym mechanizmem na Debianie/Ubuntu i na RHEL 9+ (kliknięcie otwiera pełny rozmiar)

1. Zanim padnie prośba o login i hasło, klient SSH może otrzymać zawartość `/etc/issue.net` — ale tylko jeśli `sshd_config` ma ustawioną dyrektywę `Banner /etc/issue.net`. Bez tego wpisu `sshd` nigdy nie zajrzy do tego pliku.
2. Użytkownik podaje dane uwierzytelniające. Jeśli logowanie się powiedzie, `sshd` uruchamia sesję PAM.
3. W trakcie sesji PAM moduł `pam_motd.so`, wpisany w `/etc/pam.d/sshd`, wypisuje treść MOTD. Na RHEL/Rocky domyślna konfiguracja ma dwie linie `pam_motd`: jedna wskazuje na `/run/motd.dynamic` (plik generowany przez mechanizm dynamiczny), druga (`noupdate`) dopisuje statyczną resztę z `/etc/motd.d/` bez ponownego przeliczania dynamicznej części.
4. Na Debianie/Ubuntu ten sam etap wykonuje `run-parts /etc/update-motd.d/`, zbierając standardowe wyjście wszystkich wykonywalnych skryptów z tego katalogu, w kolejności wynikającej z numeru na początku nazwy pliku, i zapisując zbiorczy wynik do `/run/motd.dynamic`.
5. Na RHEL 9 i nowszych ten sam mechanizm jest prostszy: `/etc/motd.d/` zawiera gotowe fragmenty tekstu (nie skrypty), które `pam_motd` po prostu skleja w jedną całość — nie trzeba pisać ani uruchamiać niczego, żeby dodać własną sekcję.
6. Dopiero po tym wszystkim powłoka logowania rusza i użytkownik widzi prompt.

Kluczowy punkt architektury: **sam plik `/etc/motd` nie jest już głównym źródłem prawdy na nowoczesnych dystrybucjach** — jest raczej celem, do którego coś inne (skrypt, `pam_motd`, `run-parts`) zapisuje wynik tuż przed wyświetleniem. Ręczna edycja `/etc/motd` na systemie z aktywnym dynamicznym MOTD zwykle kończy się tym, że własna treść zostaje nadpisana przy następnym logowaniu przez ten sam mechanizm, który wygenerował poprzednią wersję.

Krok po kroku

1. Sprawdź, co obecnie generuje MOTD na Twoim systemie.

```
grep motd /etc/pam.d/sshd
ls /etc/update-motd.d/ 2>/dev/null
ls /etc/motd.d/ 2>/dev/null
```

Obecność `update-motd.d` wskazuje na mechanizm Debian/Ubuntu, obecność `motd.d` — na RHEL 9+.

2. Ustaw prawny baner ostrzegawczy przed logowaniem.

```
cat > /etc/issue.net <<'EOF'
UWAGA: dostęp wyłącznie dla autoryzowanych użytkowników.
Wszystkie działania są monitorowane i rejestrowane.
Nieautoryzowany dostęp podlega odpowiedzialności prawnej.
EOF
cp /etc/issue.net /etc/issue
```

3. Włącz baner w konfiguracji SSH.

```
sed -i 's|^#\?Banner.*|Banner /etc/issue.net|' /etc/ssh/sshd_config
sshd -t && systemctl reload sshd
```

`sshd -t` sprawdza poprawność składni pliku przed przeładowaniem usługi — pomija ryzyko zerwania dostępu przez literówkę w konfiguracji.

```
[bez wyniku = konfiguracja poprawna]
```

4. Sprawdź efekt z zewnątrz, bez logowania.

```
ssh -o PreferredAuthentications=none adm_user@app01
```

```
UWAGA: dostęp wyłącznie dla autoryzowanych użytkowników.
Wszystkie działania są monitorowane i rejestrowane.
Nieautoryzowany dostęp podlega odpowiedzialności prawnej.
adm_user@app01: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
```

Baner pojawia się mimo odrzuconego logowania — dokładnie zgodnie z regułą „przed uwierzytelnieniem, widoczny dla każdego”.

5. Dodaj dynamiczną informację o stanie systemu, widoczną tylko po zalogowaniu.

Na RHEL/Rocky 9+:

```
cat > /etc/motd.d/80-stan <<'EOF'
Ostatnia aktualizacja pakietów sprawdzona: patrz /var/log/dnf.log
EOF
```

Na Debianie/Ubuntu — skrypt z sekcji Fiszki, z `chmod +x`.

Częste błędy

Ustawienie `/etc/issue.net`, ale bez `Banner` w `sshd_config`.

```
$ cat /etc/issue.net
UWAGA: dostęp monitorowany.
$ ssh adm_user@app01
[żaden baner się nie pokazuje]
```

Zero komunikatu błędu — po prostu nic się nie dzieje, bo `sshd` bez dyrektywy `Banner` nigdy nie sięga po ten plik. To jedyny z plików banerów, który wymaga jawnej konfiguracji, żeby zacząć działać — `/etc/issue` czyta `getty` automatycznie, bez żadnego przełącznika.

Skrypt w `update-motd.d` bez prawa wykonania.

```
$ ls -l /etc/update-motd.d/80-stan-systemu
-rw-r--r-- 1 root root 240 wrz 18 09:00 /etc/update-motd.d/80-stan-systemu
$ ssh adm_user@app01
[dynamiczna sekcja po prostu brakuje, reszta MOTD działa normalnie]
```

`run-parts` po cichu pomija pliki bez uprawnienia do wykonania — nie ma logu, nie ma ostrzeżenia. Naprawa: `chmod +x /etc/update-motd.d/80-stan-systemu`.

Nadpisanie `/etc/motd` ręcznie na systemie z aktywnym `pam_motd` i `motd.dynamic`.

```
cat > /etc/motd <<'EOF'
Moja własna treść
EOF
```

Po następnym zalogowaniu treść znika albo miesza się z wygenerowaną częścią — bo `pam_motd` w wielu konfiguracjach wypisuje `/run/motd.dynamic` zamiast (albo obok) `/etc/motd`, a mechanizm generujący ten plik nie wie nic o ręcznej zmianie i przy kolejnym uruchomieniu nadpisze ją swoim wynikiem. Zanim edytujesz `/etc/motd` bezpośrednio, sprawdź krokiem 1 wyżej, czy coś już nim nie zarządza.

Przeniesienie skryptu MOTD z innego uniksa bez zmiany komend na Linuksowe.

```
#!/bin/bash
LOAD=$(sysctl -n vm.loadavg | awk '{print $2, $3, $4}')
```

```
sysctl: cannot stat /proc/sys/vm/loadavg: No such file or directory
```

`sysctl -n vm.loadavg` i `hw.physmem` to identyfikatory z BSD/macOS — na Linuksie `sysctl` operuje na zupełnie innej przestrzeni nazw (`/proc/sys/...`), więc te konkretne OID-y nie istnieją i komenda kończy się błędem. Linuksowe odpowiedniki: `cat /proc/loadavg` albo `uptime` dla obciążenia, `free -`

`h` dla pamięci. Gotowy skrypt MOTD znaleziony w internecie albo przeniesiony z innego środowiska trzeba zawsze zweryfikować pod kątem tego, na jakim systemie faktycznie ma działać.

□□ Ćwiczenia

> Wszystko poniżej wykonujesz na własnej maszynie, jako `root` albo przez `sudo`, bo pliki w `/etc/` wymagają uprawnień administracyjnych.

Ćwiczenie 1 — trzy pliki, trzy momenty

```
echo "TRESC-ISSUE" | sudo tee /etc/issue > /dev/null
echo "TRESC-MOTD" | sudo tee /etc/motd > /dev/null
```

Zaloguj się na konsolę lokalną (albo `su - $(whoami)` w tym samym terminalu, jeśli nie masz dostępu do konsoli fizycznej) i zaobserwuj, w którym momencie widzisz każdy z napisów. Które pojawia się przed podaniem hasła, a które po?

Odpowiedź

`TRESC-ISSUE` pojawia się przed monitem o login/hasło — to `/etc/issue`, czytane przez `getty` zawsze, bez dodatkowej konfiguracji. `TRESC-MOTD` pojawia się dopiero po udanym zalogowaniu — to `/etc/motd`, wypisywane przez mechanizm PAM w trakcie otwierania sesji. `su -` symuluje pełne logowanie (z odczytem plików startowych), więc `TRESC-MOTD` powinno się pojawić także tą drogą.

Ćwiczenie 2 — własny dynamiczny fragment

Na RHEL/Rocky:

```
sudo bash -c 'cat > /etc/motd.d/90-cwiczenie <<EOF
Wolne miejsce na /tmp: $(df -h /tmp | awk "NR==2{print \$4}")
EOF'
cat /etc/motd.d/90-cwiczenie
```

Na Debianie/Ubuntu, zamiast tego:

```
sudo tee /etc/update-motd.d/90-cwiczenie > /dev/null <<'EOF'
#!/bin/bash
echo "Wolne miejsce na /tmp: $(df -h /tmp | awk 'NR==2{print $4}')"
EOF
sudo chmod +x /etc/update-motd.d/90-cwiczenie
sudo run-parts /etc/update-motd.d/
```

Dlaczego wersja z RHEL wstawia wynik `df` **raz, na stałe**, w momencie tworzenia pliku, a wersja z Debiana przelicza go **przy każdym logowaniu**?

Odpowiedź

Plik w `/etc/motd.d/` to gotowy tekst — polecenie `$(df -h ...)` w heredoku bez apostrofów wokół `EOF` zostaje rozwinięte przez powłokę **raz**, w chwili tworzenia pliku, i od tej pory plik zawiera już tylko zastygły wynik, tak jak każdy statyczny tekst. Plik w `/etc/update-motd.d/` to skrypt z `#!/bin/bash` na początku, uruchamiany przez `run-parts` na nowo przy każdym logowaniu — polecenie `df` w jego treści wykonuje się dopiero wtedy, więc wynik jest zawsze aktualny na moment logowania, nie na moment pisania skryptu.

Pytania kontrolne

1. Administrator chce ostrzeżenie prawne widoczne nawet dla kogoś, kto poda złe hasło i nigdy się nie zaloguje. W którym pliku (plikach) je umieści i dlaczego nie wystarczy sam `/etc/motd`?

Odpowiedź

W `/etc/issue` (konsola lokalna) i `/etc/issue.net` (SSH, wymaga też ustawienia `Banner` w `sshd_config`) — oba są wyświetlane przed uwierzytelnieniem, więc widzi je każdy, kto tylko połączy się z systemem, niezależnie od wyniku logowania. `/etc/motd` nie wystarczy, bo pokazuje się wyłącznie po udanym zalogowaniu — ktoś, kto nigdy się nie zaloguje, nigdy go nie zobaczy.

2. Skrypt w `/etc/update-motd.d/50-info` ma poprawną treść, ale nie pojawia się w MOTD po zalogowaniu. Jaka jest najbardziej prawdopodobna przyczyna do sprawdzenia w pierwszej kolejności?

Odpowiedź

Brak prawa wykonania na pliku. `run-parts` pomija pliki bez `chmod +x` całkowicie po cichu, bez żadnego komunikatu — pierwsze, co warto sprawdzić, to `ls -l` na tym skrypcie.

3. Czym różni się mechanizm `/etc/update-motd.d/` od `/etc/motd.d/`?

Odpowiedź

`/etc/update-motd.d/` (Debian/Ubuntu) zawiera skrypty wykonywalne uruchamiane przez `run-parts` — każdy generuje fragment treści od nowa przy logowaniu. `/etc/motd.d/` (RHEL 9+) zawiera gotowe, statyczne fragmenty tekstu, które `pam_motd` po prostu skleja w jedną całość, bez uruchamiania czegokolwiek — prostszy mechanizm, ale bez możliwości wstawienia świeżo obliczonej wartości bez zewnętrznego odświeżania pliku.

Powiązane: PAM: moduły, stosy i sesje logowania · sshd_config: najważniejsze dyrektywy · Pliki startowe powłoki i zmienne środowiskowe

Źródła: `man 5 motd` · `man 5 issue` · `man 8 sshd_config` (sekcja `Banner`, `PrintMotd`) · `man 8 pam_motd` · materiały wewnętrzne — skrypt dynamicznego MOTD

Revision #15
Created 2026-09-18 06:57:19 UTC by Tomasz Konieczny
Updated 2026-10-03 19:09:59 UTC by Tomasz Konieczny